

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Веб-технології»)

на тему:

**«Моделювання та візуалізація адаптивних
алгоритмів регулювання дорожнього руху»**

Виконав студент групи ВТ-21-1
БАБУШКО Андрій Сергійович

Керівник роботи:
ВЛАСЕНКО Олег Васильович

Рецензент:
ФАНТ Микола Олександрович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна Вакалюк

«26» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **БАБУШКО Андрій Сергійович**

Керівник роботи: **ВЛАСЕНКО Олег Васильович**

Тема роботи: **«Моделювання та візуалізація адаптивних алгоритмів регулювання дорожнього руху»**,

затверджена Наказом закладу вищої освіти від «14» лютого 2025 р., №61/с

Вихідні дані для роботи: **процес управління міським транспортним рухом за допомогою адаптивної системи керування світлофорами, що реалізує централізоване управління перехрестями з урахуванням реального навантаження на дорожню мережу. Система використовує алгоритми оптимізації та механізми зворотного зв'язку для формування ефективних графіків роботи світлофорів з метою підвищення пропускну здатності та безпеки дорожнього руху.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Власенко О.В.	27.02.2025р.	27.02.2025р.
2	Власенко О.В.	27.02.2025р.	27.02.2025р.
3	Власенко О.В.	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Записка до випускної кваліфікаційної роботи бакалавра складається зі вступу, трьох розділів, висновків та списку використаних джерел.

Текстова частина викладена на 45 сторінках друкованого тексту. Список використаних джерел містить 27 найменувань і займає 3 сторінки. У роботі наведено 26 рисунків. Загальний обсяг роботи – 240 сторінок.

У першому розділі виконано постановку задачі, проаналізовано існуючі програмні рішення у сфері адаптивного керування дорожнім рухом та обґрунтовано вибір архітектури, алгоритмів і технологій реалізації.

Другий розділ присвячено розробці архітектури системи, опису об'єктно-орієнтованої моделі, діаграм кооперації та послідовності, а також логіці реалізації алгоритмів керування на основі стратегій «зеленої хвилі».

У третьому розділі продемонстровано процес використання системи, запуск симуляцій, аналіз результатів за ключовими метриками та тестування компонентів. Наведено рисунки, графіки та витяги з логів, що підтверджують ефективність реалізованого підходу.

Висновки містять узагальнення отриманих результатів, а також підтвердження виконання усіх поставлених завдань.

Ключові слова: ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА, C++, SUMO, КЕРУВАННЯ СВІТЛОФОРАМИ, АДАПТИВНИЙ АЛГОРИТМ, СИМУЛЯЦІЯ, МОДЕЛЬ, ЗЕЛЕНА ХВИЛЯ.

					ІПЗ.КР.Б.-121-25-ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Бабушко А.С.			Моделювання та візуалізація адаптивних алгоритмів дорожнього руху			Літ.	Арк.	Аркушів	
										3	240
Керівник		Власенко О.В.		ФІКТ Гр. ВТ-21-1							
Н. контр.		Тичина О.П.									
Зав. каф.		Вакалюк Т.А.									

ABSTRACT

The explanatory note for the final qualification project consists of an introduction, three chapters, a conclusion, and a list of references.

The textual part comprises 45 pages of printed text. The list of references includes 27 sources and takes up 3 pages. The work contains 26 figures. The total volume of the project is 240 pages.

The first chapter outlines the problem statement, reviews existing adaptive traffic control systems, and provides a rationale for the chosen architecture, algorithms, and development tools.

The second chapter presents the object-oriented structure of the system, describes the interaction between its components via UML diagrams, and details the logic behind adaptive strategies such as green-wave control.

The third chapter demonstrates the use of the system through simulation scenarios, output analysis, and automated testing. It includes performance metrics, visual results, and examples of system logs.

The conclusion summarizes the results achieved and confirms the successful completion of the project's objectives.

Keywords: FINAL QUALIFICATION PROJECT, C++, SUMO, TRAFFIC CONTROL, ADAPTIVE ALGORITHM, SIMULATION, MODEL, GREEN WAVE.

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІПЗ	Арк.
		Власенко О.В.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	6
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ, ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ	10
1.1. Визначення та постановка задачі	10
1.2. Аналіз існуючого ПЗ	11
1.3. Визначення архітектури ПЗ	16
1.4. Обґрунтування вибору інструментальних засобів та вимоги до ПЗ	17
Висновки до першого розділу	20
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	21
2.1. Аналіз вимог до ПЗ	21
2.2. Визначення об'єктно-орієнтованої структури системи	25
2.3. Комунікації і послідовність взаємодії об'єктів системи.....	29
2.4. Алгоритм роботи та стани програмної системи	31
Висновки до другого розділу	36
РОЗДІЛ 3. ДЕМОНСТРАЦІЯ РОБОТИ ТА ТЕСТУВАННЯ СИСТЕМИ.....	37
3.1. Інструкція з використання програми.....	37
3.2. Результати тестування і перевірка алгоритму	44
Висновки до третього розділу	46
ВИСНОВКИ.....	48
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	50
ДОДАТКИ.....	53
Додаток А.....	54
Додаток В.....	54
Додаток С.....	55

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

СКС – Система керування світлофорами
ПЗ – Програмне забезпечення
ООП – Об'єктно-орієнтоване програмування
API – Application Programming Interface
CI/CD – Continuous Integration/Continuous Delivery
GUI – Graphical User Interface
CSV – Comma-Separated Values
UML – Unified Modeling Language
PID – Proportional–Integral–Derivative
CLI – Command Line Interface
XML – Extensible Markup Language
SIMD – Single Instruction Multiple Data
AVX – Advanced Vector Extensions
SSE – Streaming SIMD Extensions
C++ – Мова програмування «C++»
C++20 – Стандарт мови програмування C++ 2020 року
CL – CLion
SUMO – Simulation of Urban Mobility
libsumo – Бібліотека керування симуляцією SUMO з C++
spdlog – Бібліотека для логування в C++
yaml-cpp – Бібліотека для роботи з YAML у C++
PugiXML – Бібліотека для обробки XML у C++
Matplot++ – Бібліотека побудови графіків на C++
Doxygen – Система генерації документації з коду
Conan – Менеджер пакетів для C++
Docker – Платформа для контейнеризації додатків
clang-format – Інструмент для форматування коду на основі Clang
Pandoc – Конвертер документів між форматами

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

PlantUML – Інструмент для створення UML-діаграм з тексту

GitLab – Платформа для розробки програмного забезпечення з підтримкою CI/CD

GitHub Pages – Сервіс публікації веб-документації

GoogleTest – Фреймворк для написання модульних тестів у C++

Graphviz – Інструмент для візуалізації графів

Homebrew – Менеджер пакунків для macOS

AUR – Arch User Repository

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

У випускній кваліфікаційній роботі бакалавра наведено процес розробки та впровадження адаптивної системи керування світлофорами й дорожнім рухом. Така система є програмним продуктом, що дає змогу організовувати й оптимізувати роботу світлофорів, налаштовувати пріоритети, відстежувати інтенсивність руху та взаємодіяти з різними дорожньо-транспортними службами. Ці можливості дозволяють покращити пропускну здатність доріг, зменшити тривалість заторів, підвищити рівень безпеки на перехрестях і сприяти раціональному використанню транспортної інфраструктури.

Актуальність теми зумовлена стрімким зростанням кількості транспортних засобів у містах та необхідністю мінімізувати затори й підвищувати безпеку дорожнього руху. Сучасні технології дозволяють упровадити адаптивні алгоритми, які динамічно реагують на зміни трафіку та пріоритети різних учасників руху, що сприяє зниженню кількості дорожньо-транспортних пригод і підвищенню ефективності використання вулично-дорожньої мережі.

Об'єктом дослідження є процес проектування, розробки та експлуатації адаптивної системи керування світлофорами з централізованим управлінням перехрестями та динамічним коригуванням робочих циклів залежно від реального навантаження на дорожню мережу.

Предметом дослідження є методи й технології збору та аналізу даних про транспортні потоки й пріоритети проїзду, а також алгоритми формування оптимальних графіків роботи світлофорів для підвищення пропускну здатності та безпеки дорожнього руху.

Метою роботи є спроектувати та реалізувати адаптивну систему автоматизованого керування світлофорами, здатну збирати та аналізувати дані про транспортні потоки й формувати оптимальні режими роботи з метою зменшення заторів, підвищення швидкості руху та забезпечення пріоритету екстрених служб.

Для досягнення поставленої мети необхідно виконати такі взаємопов'язані завдання: сформувати концепцію адаптивної системи керування світлофорами й

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІПЗ	Арк.
		Власенко О.В.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

визначити її функціональні можливості; проаналізувати та порівняти наявні рішення, виокремивши їх переваги й недоліки; обґрунтувати вибір архітектури та технологічного стеку з огляду на вимоги до продуктивності, масштабованості та інтеграції; розробити об'єктно-орієнтовану модель і необхідні UML-діаграми; реалізувати й протестувати адаптивні алгоритми разом із всіма розробленими модулями; налаштувати обмін даними із симулятором SUMO; створити модулі візуалізації та збору статистики для оцінювання ефективності; забезпечити unit та інтеграційне тестування й CI/CD-конвеєр із генерацією документації; підготувати аналітичний звіт з графічним порівнянням отриманих метрик.

Практичним значенням є можливість використання результатів роботи для створення прототипів інтелектуальних систем регулювання дорожнього руху з подальшою інтеграцією у реальну міську інфраструктуру. Реалізована система підтримує масштабування, має відкриту модульну архітектуру та адаптована для взаємодії з сенсорами, симуляторами та зовнішніми API. За тематикою дипломної роботи були опубліковані тези [1].

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ, ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1. Визначення та постановка задачі

У межах випускної кваліфікаційної роботи бакалавра ставиться мета підвищити ефективність та безпеку дорожнього руху, використовуючи систему автоматизованого управління світлофорами. Система має збирати дані про транспортний потік, аналізувати їх і автоматично визначати оптимальний графік перемикавання світлофорів. Для реалізації цієї мети доцільно сформулювати концепцію, визначити функціональні можливості та передбачити способи отримання необхідної інформації про трафік, а також розробити ефективні методи прийняття рішень щодо налаштування світлофорів.

Одним із найважливіших етапів є проектування архітектури системи. Потрібно визначити основні компоненти, продумати їхню взаємодію та інформаційні потоки між ними. Особливу увагу слід звернути на модуль збору даних, що може включати камери, датчики або зовнішні джерела, та на модуль аналізу, який оброблятиме одержані дані й формуватиме графіки роботи світлофорів. Далі постає завдання знайти або розробити алгоритми управління, які можуть адаптувати тривалість світлофорних циклів залежно від поточної завантаженості доріг. Це можуть бути адаптивні або прогнозні алгоритми, що дають змогу оперативно коригувати час перемикавання світлофорів.

Щоб оператори системи могли керувати налаштуваннями та отримувати необхідну інформацію, слід приділити увагу візуалізації та адмініструванню. Це передбачає реалізацію інтерфейсу, де відображається поточний стан перехресть у реальному часі, є можливість налаштувати роботу всієї системи та контролювати її ефективність, переглядаючи статистичні дані й показники роботи. Окрім цього, потрібно розробити план тестування та продумати способи впровадження системи, врахувати перевірку ключових показників ефективності: середнього часу затримки на перехресті, довжини черг автомобілів та інших критеріїв, що впливають на якість дорожнього руху.

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Передбачається комплексне рішення з централізованим управлінням світлофорами на різних перехрестях міста або мережі доріг. Така система охоплює джерела збору даних і модуль обробки, який перетворює отриману інформацію на структуровані показники щодо завантаженості, швидкості руху та історичних тенденцій. Далі модуль прийняття рішень, спираючись на результати аналізу, має визначити оптимальний або наближено оптимальний розклад перемикання сигналів. Для відображення реальної моделі дорожнього руху передбачається візуалізація, що дає змогу оцінити якість і результативність прийнятих рішень, переглянути аналітику ефективності, а також виконувати певні дії з адміністрування. Важливо передбачити систему зворотного зв'язку та оповіщень, яка вчасно повідомлятиме про поламки світлофорів, критичні затори чи надзвичайні ситуації.

Таким чином, визначено основні аспекти проекту, його мету та загальну структуру, що необхідні для створення й успішної реалізації адаптивної системи керування світлофорами.

1.2. Аналіз існуючого ПЗ

У випускній кваліфікаційній роботі бакалавра розглядається можливість використання адаптивного керування світлофорами, що на сьогодні залишається актуальним у містах різного масштабу. Вже існують відкриті та комерційні системи, які дають змогу зменшити затримки, підвищити пропускну здатність і забезпечити вищу безпеку на дорогах.

Відомим прикладом є SCOOT, розроблена організацією «Transport Research Laboratory» у Великій Британії. Вона використовує безперервний аналіз даних від детекторів, динамічно коригує час «зеленого» світла й загальні цикли, щоб зменшувати середній час затримки та довжину черги автомобілів. На рис. 1.1 зображено сайт компанії «Transport Research Laboratory», яка розробила SCOOT.

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				11
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис. 1.1. Сайт компанії «Transport Research Laboratory», яка розробила «SCOOT»

Аналогічно SCATS, створена Департаментом транспорту Нового Південного Вельсу, реалізує коригування фаз світлофорів на базі поточних та історичних даних про рух у межах визначених зон. Ця система також орієнтована на поліпшення пропускної здатності та мінімізацію зупинок автомобілів. На Рис. 1.2 наведено сайт транспортної системи SCATS.



Рис. 1.2. Сайт транспортної системи «SCATS»

В обох випадках перевагою є перевірені алгоритми, широкий досвід впровадження та можливість інтеграції з різними дорожніми контролерами. Серед

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

недоліків зазвичай згадують високу вартість упровадження та обслуговування, складність інтеграції в наявну інфраструктуру, необхідність постійного зв'язку з центральною системою та обов'язкове використання специфічного обладнання.

Окрім того, сьогодні активно досліджуються системи на базі машинного навчання та великих даних. Вони навчаються на історії трафіку й можуть прогнозувати майбутні затори. Перевагою є здатність самостійно вдосконалюватися під час накопичення даних та ефективніше обирати тривалість «зелених» фаз на випередження. Проте є високі вимоги до обчислювальних ресурсів, потрібна велика й різноманітна база даних, а пояснити конкретне рішення такої «чорної скриньки» часом складно.

Прикладом системи на основі ML/AI є Surtrac, створена в Університеті Карнегі-Меллона. Вона працює за децентралізованим принципом: кожне перехрестя самостійно обробляє вхідні дані й приймає рішення щодо фаз світлофора. Алгоритм враховує інтенсивність потоку, прогноз на найближчі хвилини та адаптує сигнали відповідно до змін. Система Surtrac успішно протестована в Піттсбурзі, де зафіксовано зниження середнього часу поїздки на 25% і зменшення заторів у центрі міста.



Рис. 1.3. Карта застосування системи Surtrac в Піттсбурзі

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

Сьогодні також поширені хмарні сервіси за моделлю SaaS. Такі платформи дають змогу швидко впровадити систему керування світлофорами, оскільки основна інфраструктура перебуває у хмарному середовищі. Серед переваг – централізоване керування, масштабованість і простота для користувачів. Недоліки – залежність від якісного доступу до інтернету та питання безпеки, оскільки дані про рух надсилаються до хмари.

Яскравим прикладом є EcoTrafIX від компанії Kapsch. Система забезпечує збір і аналіз даних з міських сенсорів у хмарному середовищі, централізовано адаптує сигнали світлофорів та формує звіти для міських операторів.

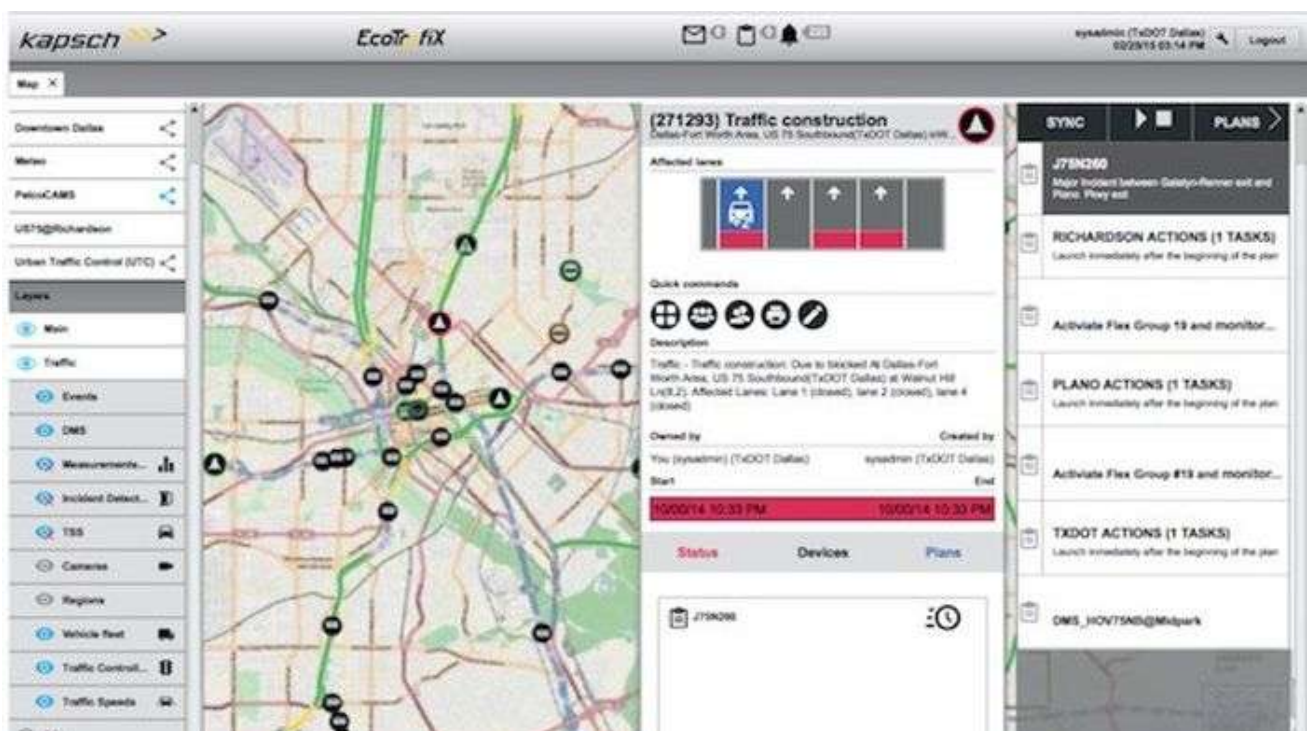


Рис. 1.4. Інтерфейс системи EcoTrafIX

Окрім базових відомостей про системи SCOOT та SCATS, доцільно зіставити конкретний набір функцій кожного рішення з тими, що реалізовані у власній системі. Це дозволяє чіткіше побачити, які можливості вже покриті, а які потребують подальшої розробки чи інтеграції.

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 1.1.

Порівняльна таблиця наявних рішень на ринку

Назва / Тип системи	Основний функціонал	Переваги	Недоліки	Приклади застосування
SCOOT (TRL, Велика Британія)	Безперервне вимірювання потоків, онлайн-коригування фаз, синхронізація «зеленої хвилі»	Перевірені алгоритми, широке впровадження	Висока вартість розгортання, складна архітектура, тривале налаштування	Лондон, Манчестер
SCATS (NSW Transport)	Розподілена оптимізація за зонами, адаптивні цикли та спліти	Глибока статистика по перехрестях, стабільність при великих мережах, гнучке зонування	Високі вимоги до каналу зв'язку, висока ціна ліцензії	Сідней, Шанхай
Системи на базі ML/AI	Самонавчання на потокових даних, адаптація до нетипових подій, прогноз заторів	Найкраща реакція на аномалії, можливість роботи без жорстких правил	Необхідність значної бази даних, потреба у потужних обчисленнях, складність пояснення алгоритмів	Пілотні міські проекти, наукові дослідження
Хмарні сервіси	Веб-моніторинг у реальному часі, ОТА-оновлення контролерів, централізована аналітика	Швидке розгортання, централізоване керування, масштабованість	Залежність від Інтернету, ризики конфіденційності даних, можливі довгострокові витрати	Середні міста, приватні рішення

Проаналізувавши наявні продукти, можна дійти висновку, що необхідно звертати увагу на гнучкість налаштувань і пристосованість системи під специфіку конкретного перехрестя чи міста. Високі початкові витрати комерційних рішень можуть бути виправдані, якщо вони забезпечують швидке та надійне функціонування, але іноді вигідніше розробити власний гнучкий аналог з урахуванням довгострокового обслуговування. Не менш важливо правильно організувати роботу з даними, використовуючи їх з різних джерел – від детекторів руху до камер спостереження, інтегрувати прогностичні алгоритми й забезпечити масштабованість. Система має бути зрозумілою та зручною для операторів, а також давати змогу швидко додавати нові перехрестя й адаптуватися до специфічних потреб міста.

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

1.3. Визначення архітектури ПЗ

У межах випускної кваліфікаційної роботи бакалавра після аналізу існуючих систем та врахування потреб власного проекту постає завдання обрати відповідну архітектуру та технологічний стек для розробки. Рішення на кшталт SCOOT чи SCATS, а також комерційні хмарні платформи, можуть виявитися надмірно дорогими або недостатньо гнучкими. У зв'язку з цим доцільно створити власне рішення на C++, яке забезпечить тонке налаштування алгоритмів і їх оптимізацію під специфіку дорожньої мережі.

Система має працювати в реальному часі або максимально наближено до нього. Це означає, що програма повинна оперативно реагувати на зміни у дорожньому потоці, коригувати тривалість «зелених» фаз, а також оптимізувати роботу світлофорів для зменшення заторів. Висока продуктивність і багатопоточна обробка даних будуть важливими, щоб уникнути запізнень та відставань під час аналізу інформації від датчиків чи камер.

Іншою ключовою вимогою є гнучкість і розширюваність. Система має дозволяти додавати нові алгоритми й змінювати параметри залежно від типу перехрестя чи стану доріг, не роблячи суттєвих змін у початковій структурі коду. Це особливо важливо, коли потрібно інтегрувати методи машинного навчання чи математики оптимізації. Також варто, щоб рішення було масштабованим, тобто зберігало ефективність і керованість при збільшенні кількості перехресть та обсягів даних. Для взаємодії із зовнішніми датчиками та контролерами слід передбачити просту інтеграцію, використовуючи модульний підхід і чіткі інтерфейси.

Архітектуру можна будувати в монолітному вигляді, коли всі складники поєднані в одному виконуваному файлі. Це спрощує розгортання, але ускладнює оновлення й масштабування. Можна розглядати і багатошаровий підхід, який поділяє систему на кілька логічних модулів. Така модель дає змогу рознести складні частини в окремі процеси й забезпечити чітке розділення функцій, хоча повноцінна масштабованість все одно може вимагати додаткових зусиль.

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Найдоцільнішим варіантом вважається *мікросервісна архітектура*. Завдяки поділу на невеликі незалежні сервіси можна поступово розширювати систему й адаптувати кожний сервіс до актуальних умов роботи без кардинальних змін у всій програмі. Таке розділення полегшує внесення нових алгоритмів машинного навчання, підключення додаткових вузлів для обробки великих обсягів інформації та спрощує взаємодію зі сторонніми компонентами завдяки відкритим мережевим API. Хоча мікросервісний підхід потребує ретельнішої розробки інфраструктури та продуманої взаємодії між сервісами, він надає найбільшу гнучкість, масштабованість і стабільність, що є визначальними факторами для сучасної адаптивної системи керування світлофорами.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до ПЗ

У процесі розробки адаптивної системи керування світлофорами було прийнято рішення використовувати сучасний, ефективний і гнучкий технологічний стек, що забезпечить високу продуктивність і масштабованість програмного забезпечення. Основною мовою реалізації стала C++ у стандарті C++20, оскільки вона дозволяє ефективно працювати з пам'яттю, багатопоточністю та надає великий контроль над ресурсами системи, що особливо важливо для задач реального часу та симуляцій дорожнього руху.

Для моделювання поведінки транспортних потоків використовується SUMO(Simulation of Urban Mobility) – це один із найвідоміших і перевірених симуляторів дорожнього руху з відкритим кодом, що активно використовується у наукових дослідженнях, комерційних розробках та муніципальних проектах. Його алгоритми базуються на перевірених мікроскопічних і макроскопічних моделях руху, таких як «Krauss car-following model», яка забезпечує реалістичну поведінку транспортних засобів з урахуванням їхніх швидкостей, відстаней і реакцій на події на дорозі.

SUMO дозволяє моделювати як поодинокі перехрестя, так і великі мережі з тисячами елементів інфраструктури. Система підтримує реалістичне моделювання зміни фаз світлофорів, черг, затримок, зупинок, середньої швидкості руху та інших

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

критичних показників. Алгоритми симуляції проходять численні валідації в академічних дослідженнях і широко цитуються в науковій літературі. Завдяки цьому, дані, що генеруються в процесі моделювання, є коректними і достовірними для аналізу адаптивних алгоритмів керування дорожнім рухом.

SUMO також забезпечує точний часовий контроль з фіксованою позначкою часу, детальні XML-виходи і можливість управління в реальному часі через API *libsumo*. Це робить його ідеальним середовищем для тестування й порівняння алгоритмів у повторюваних сценаріях. Через API система отримує інформацію про стан трафіку, передає команди керування світлофорами та аналізує результати в реальному часі. Це дає змогу гнучко змінювати алгоритми, враховуючи поточне навантаження на мережу.

Журналювання та ведення діагностики здійснюється за допомогою бібліотеки *spdlog*, яка дозволяє реалізувати як просте консольне логування, так і вивід у файл з контролем форматів і рівнів важливості. Це дозволяє зручно відслідковувати поведінку алгоритму під час симуляції та аналізувати можливі аномалії або нестандартні ситуації.

Для конфігурації системи використовується *yaml-cpp*. Вона дозволяє зручно описувати налаштування у форматі YAML – від початкових параметрів симуляції до конфігурацій перехресть, циклів керування та граничних значень. Завдяки цьому система легко адаптується до різних сценаріїв без потреби перекомпіляції коду.

Візуалізація аналітичних результатів і метрик виконується за допомогою бібліотеки *Matplot++*, яка забезпечує побудову графіків, гістограм і часових рядів. Це дозволяє дослідити ефективність реалізованих стратегій керування та порівнювати їх між собою на основі симуляційних даних.

Для роботи з XML-файлами, які активно використовуються в SUMO для опису дорожньої мережі, конфігурацій маршрутів та результатів симуляції, обрана бібліотека *Rug/XML*. Вона дозволяє зручно та швидко парсити великі XML-документи, витягувати необхідну інформацію і передавати її до модулів системи.

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

З технічної точки зору, система повинна бути здатна запускатися на Linux-серверах. Під час виконання алгоритмів особливу увагу приділено стабільності багатопотокової роботи, безпеці пам'яті та ізоляції відповідальності модулів. Базова архітектура побудована у вигляді наближеного до мікросервісного підходу, що дозволяє незалежно розвивати логіку симуляції, обробки статистики та взаємодії з зовнішніми інтерфейсами.

Програмне забезпечення повинно масштабуватися відповідно до кількості перехресть та підтримувати оновлення без зупинки роботи всієї системи. Це досягається завдяки модульній побудові та чіткому розділенню відповідальностей між основними компонентами, такими як менеджер даних, алгоритм керування, статистичний збирач і контролер взаємодії із SUMO.

Для кращої візуалізації було створено таблицю 2, яка описує та порівнює характеристики інструментальних засобів.

Таблиця 1.2.

Порівняльна характеристика використаних інструментальних засобів

Інструмент	Призначення	Основні переваги
C++20	Основна мова програмування	Висока продуктивність, контроль над ресурсами, підтримка багатопоточності
libsumo	Інтеграція з SUMO	Доступ до даних трафіку в реальному часі, підтримка керування світлофорами
spdlog	Логування	Швидка робота, форматування, логування у файл, гнучка конфігурація
yaml-cpp	Завантаження конфігурацій	Зручний синтаксис YAML, гнучка структура параметрів
PugiXML	Обробка XML	Легка й швидка бібліотека, зручна робота з SUMO-файлами
Matplot++	Побудова графіків	Статистика, візуалізація результатів, підтримка інтеграції з C++

У результаті обраний набір інструментів дозволяє забезпечити ефективну розробку, налагодження, тестування та розширення системи адаптивного керування дорожнім рухом, що дає змогу гнучко реагувати на реальні транспортні умови та досягати стабільного зменшення заторів у модельованих сценаріях.

Висновки до першого розділу

У першому розділі окреслено загальну ідею проекту та його актуальність. Проаналізовано наявні рішення, зокрема SCOOT, SCATS, ML і AI, а також хмарні сервіси, визначено їхні переваги та недоліки. На основі зробленого аналізу сформульовано вимоги до власної системи і визначено оптимальну архітектуру мікросервісного типу, яка здатна забезпечити гнучкість, масштабованість і спростити інтеграцію з іншими програмними рішеннями. Розглянуто основні інструменти та вимоги до розроблюваного програмного забезпечення, що дають змогу ефективно втілити адаптивне керування світлофорами в реальному часі.

Таким чином, сформовано концептуальне бачення проекту, проведено детальний аналіз конкурентних систем та обґрунтовано технології й інструменти, які забезпечать надійну і високопродуктивну роботу майбутнього продукту.

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Аналіз вимог до ПЗ

Зростання кількості автомобілів у містах, збільшення навантаження на вулично-дорожню мережу та поява нових форм транспорту створюють серйозні виклики для ефективного керування дорожнім рухом. Однією з найгостріших проблем є не адаптивність традиційних світлофорних систем, які працюють за заздалегідь визначеними графіками, не враховуючи поточну ситуацію на дорогах. У результаті цього виникають затори, зростає час у дорозі та ризики дорожньо-транспортних пригод.

Класичні методи керування фазами світлофорів – жорстко задані цикли або статистичні налаштування вже не здатні відповідати динаміці сучасного трафіку. Вони не враховують коливання навантаження протягом дня, не реагують на неочікувані ситуації та не можуть забезпечити пріоритетний проїзд екстреного транспорту. Крім того, відсутність можливості реального моніторингу або ручного втручання ускладнює роботу операторів і знижує ефективність реагування у разі позаштатних подій.

У межах цього проекту реалізовано адаптивну систему керування світлофорами (далі – СКС), яка враховує реальний стан дорожнього руху, аналізує поточні дані та формує графіки перемикання фаз динамічно. СКС використовує алгоритм «зеленої хвилі» для синхронізації світлофорів уздовж напрямів із високим потоком, забезпечує можливість пріоритетного проїзду для екстрених служб і надає оператору інструменти ручного втручання. Для імітації умов роботи системи використовується SUMO – симулятор міського руху, який дозволяє тестувати алгоритми в різноманітних сценаріях.

Серед основних функцій – автоматичне й ручне керування світлофорами, формування статистики, візуалізація ефективності, інтеграція з сенсорами та контролерами. Користувачами системи виступають оператори, адміністративний персонал, служби обслуговування, зовнішні системи, муніципалітети, а також

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

транспортні й аварійні служби. Усі вони взаємодіють із підсистемами, які забезпечують збір, обробку, аналіз і реагування на події.

На рисунку 2.1 представлено діаграму варіантів використання, яка демонструє взаємодію між різними користувачами та функціями системи.

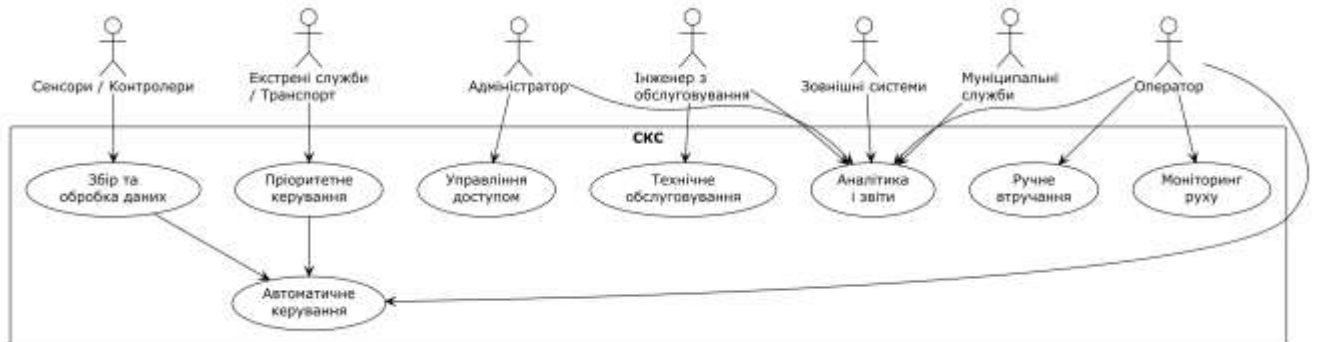


Рис. 2.1. Діаграма варіантів використання системи керування світлофорами

Опис функціональних сценаріїв представлено в таблиці нижче:

Таблиця. 2.1.

Реєстр варіантів використання

Код	Основний актор	Найменування	Формулювання
UC1	Оператор	Моніторинг руху	Спостереження за станом перехресть у реальному часі, отримання повідомлень про затори та аномальні події
UC2	Оператор, Сенсори/Контроллери	Автоматичне керування	Коригування тривалості фаз сигналів на підставі поточних даних, реалізація алгоритму «зеленої хвилі»
UC3	Оператор	Ручне втручання	Термінове переключення світлофорів для оперативного вирішення нестандартних ситуацій
UC4	Адміністратор	Управління доступом	Налаштування ролей користувачів і контроль їхніх привілеїв, аудит критичних дій
UC5	Екстрені служби / Транспорт	Пріоритетне керування	Зміна фаз на користь спецтранспорту, забезпечення «зеленого коридору» і мінімальної кількості зупинок

Код	Основний актор	Найменування	Формулювання
UC6	Оператор, Адміністратор, Зовнішні системи, Муніципальні служби, Інженер	Аналітика й звіти	Формування статистичних звітів, побудова графіків, аналіз роботи «зеленої хвилі» і відстеження середніх затримок
UC7	Сенсори/Контроллери	Збір та обробка даних	Автоматичне надсилання інформації про кількість і швидкість руху авто, фільтрація шумів і аномалій
UC8	Інженер з обслуговування	Технічне обслуговування	Перевірка стану світлофорного обладнання, планування заміни ламп, налаштування регламенту профілактики

Окрім функціональних, СКС має також нефункціональні вимоги. Вона повинна забезпечувати безперервну роботу, стабільність і надійність навіть у складних транспортних умовах. Дані, що передаються системою, мають бути захищені через шифрування та контроль доступу. СКС повинна бути здатною масштабуватись як на малі райони, так і на великі міські мережі з численними сенсорами та вузлами.

Щоб оцінити ефективність роботи адаптивного алгоритму, в системі впроваджено ряд метрик, які формуються автоматично під час моделювання. Зокрема, вимірюється середній час затримки транспортних засобів, що дозволяє оцінити рівень завантаженості перехресть. Додатково відстежується середня довжина черги, яка сигналізує про потенційні вузькі місця у мережі. Кількість зупинок на один маршрут дає уявлення про ступінь безперервності руху та реалізацію «зеленої хвилі». Також враховується пропускна здатність перехресть – тобто кількість транспортних засобів, що успішно перетнули контрольні точки протягом певного періоду. Ще одним критичним параметром є частка часу, протягом якого активна фаза світлофора була доступною для руху, що допомагає зрозуміти, наскільки ефективно використано доступний час циклу.

Зазначені метрики базуються на даних симуляцій, результати яких зберігаються у форматі CSV. До кожного запуску сценарію записується повна

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				23
Змн.	Арк.	№ докум.	Підпис	Дата		

інформація про навантаження, середню швидкість руху, загальний час у дорозі, час очікування, кількість зупинок, довжину черги, а також значення щільності руху в пік та втрату часу на маршруті. Цей формат дозволяє не лише збирати статистику, але й будувати порівняння між різними конфігураціями системи й режимами керування для подальшого аналізу.

Для ілюстрації деяких важливих сценаріїв було створено додаткові діаграми розширених варіантів використання. На рисунку 2.2 відображено логіку пріоритетного керування, коли екстрені служби або громадський транспорт ініціюють запит на забезпечення «зеленого коридору». СКС отримує сигнал, виділяє найважливіші напрями руху, вмикає послідовно зелений сигнал і, за необхідності, автоматично блокує конфліктні фази. Після завершення проїзду спеціалізованого транспорту СКС повертається до звичайного адаптивного режиму.



Рис. 2.2. Діаграма сценарію «Пріоритетне керування»

Ще один важливий сценарій – ручне втручання оператора, зображений на рисунку 2.3. Оператор проводить моніторинг руху та, якщо вважає за потрібне,

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				24
Змн.	Арк.	№ докум.	Підпис	Дата		

примусово змінює сигнали на конкретному перехресті. Це може відбуватись у разі аварії, перекриття частини дороги або тимчасової події, коли алгоритми потрібно скоригувати миттєво, без очікування автоматичної реакції.

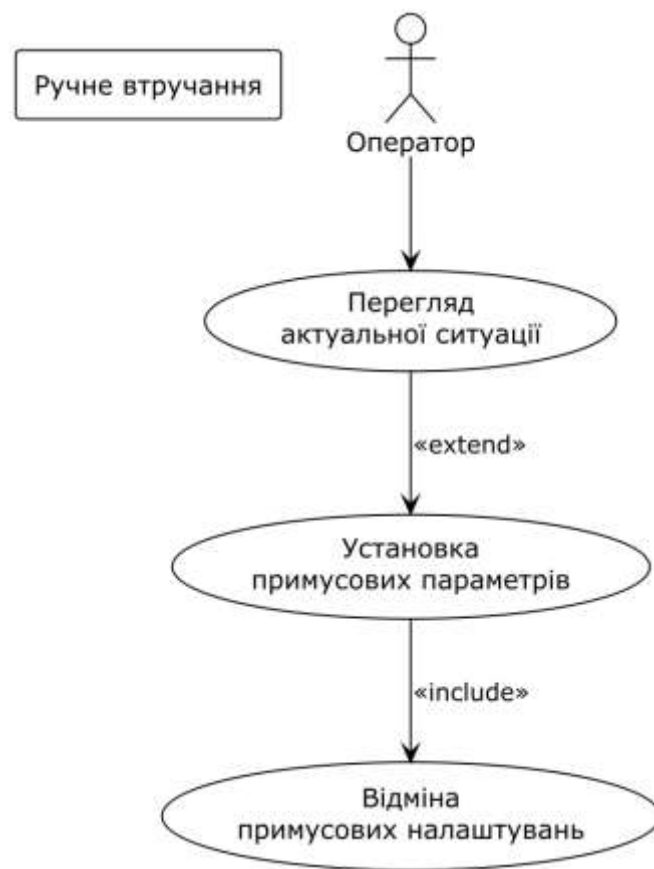


Рис. 2.3. Діаграма сценарію «Ручне втручання»

Загалом поєднання всіх описаних вимог, а також визначення ключових варіантів використання системи дає чітке уявлення про те, що має робити СКС і якими параметрами вона повинна характеризуватися. Отримана модель створює основу для об'єктно-орієнтованої структури і допомагає визначити, які саме модулі й компоненти забезпечать повний цикл керування – від отримання даних із сенсорів до аналізу ефективності «зеленої хвилі».

2.2. Визначення об'єктно-орієнтованої структури системи

Система адаптивного керування світлофорами реалізована з використанням об'єктно-орієнтованого підходу. Архітектура спроектована так, щоб забезпечити гнучкість, модульність та можливість масштабування. Основу структури становлять класи, що моделюють об'єкти дорожньої інфраструктури, такі як

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

перехрестя, світлофори та транспортні засоби. Вони представлені у вигляді класів `PulseIntersection`, `PulseTrafficLight` і `PulseVehicle`. Для опису зв'язків між перехрестями використовується клас `PulseRoadConnection`, який містить інформацію про відстань і асоційований світлофор.

Усі об'єкти симуляції зберігаються та синхронізуються через єдиний керівний клас `PulseDataManager`, який реалізовано як `Singleton`. Його завданням є централізоване управління всіма сутностями системи, а також обробка оновлень, отриманих із зовнішнього симулятора. Цей симулятор – система SUMO, яка інтегрується через клас `SumoIntegration`, який, в свою чергу, виконує роль проксі для взаємодії з реальним середовищем, зокрема для крокування симуляції, зчитування станів та керування світлофорами.

Алгоритм адаптивного управління, зокрема реалізація зеленої хвилі, знаходиться у класі `PulseTrafficAlgo`. Він аналізує поточну ситуацію на дорогах, використовуючи дані з `PulseDataManager` і `SumoIntegration`, та визначає оптимальні фази світлофорів. Центральною частиною алгоритму є `MetaController`, який обирає відповідну стратегію з-поміж реалізацій інтерфейсу `IGreenWaveStrategy`, зокрема `BaselineWaveStrategy`, `HybridWaveStrategy` та `MaxPressureWaveStrategy`. Вибір режиму базується на рівні трафіку та довжині черг на перехрестях, які визначає `TrafficModeController`. У випадку використання гібридної стратегії додатково залучається `PIDController`, параметри якого динамічно адаптуються за допомогою модуля `AutoTuner`.

Глобальний керівний клас `TrafficSystem` координує взаємодію всіх підсистем, забезпечує запуск симуляції, виконує її поетапне оновлення та публікує відповідні події через централізовану систему подій `EventBus`, яка реалізує патерн `Observer`. Таким чином, взаємодія між модулями організована через модель орієнтовану на події, що гарантує слабку зв'язаність і модульність усіх компонентів.

Нижче наведено UML-діаграму, яка демонструє структуру основних класів, їхню відповідальність і зв'язки між ними в рамках архітектури системи.

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				26
Змн.	Арк.	№ докум.	Підпис	Дата		

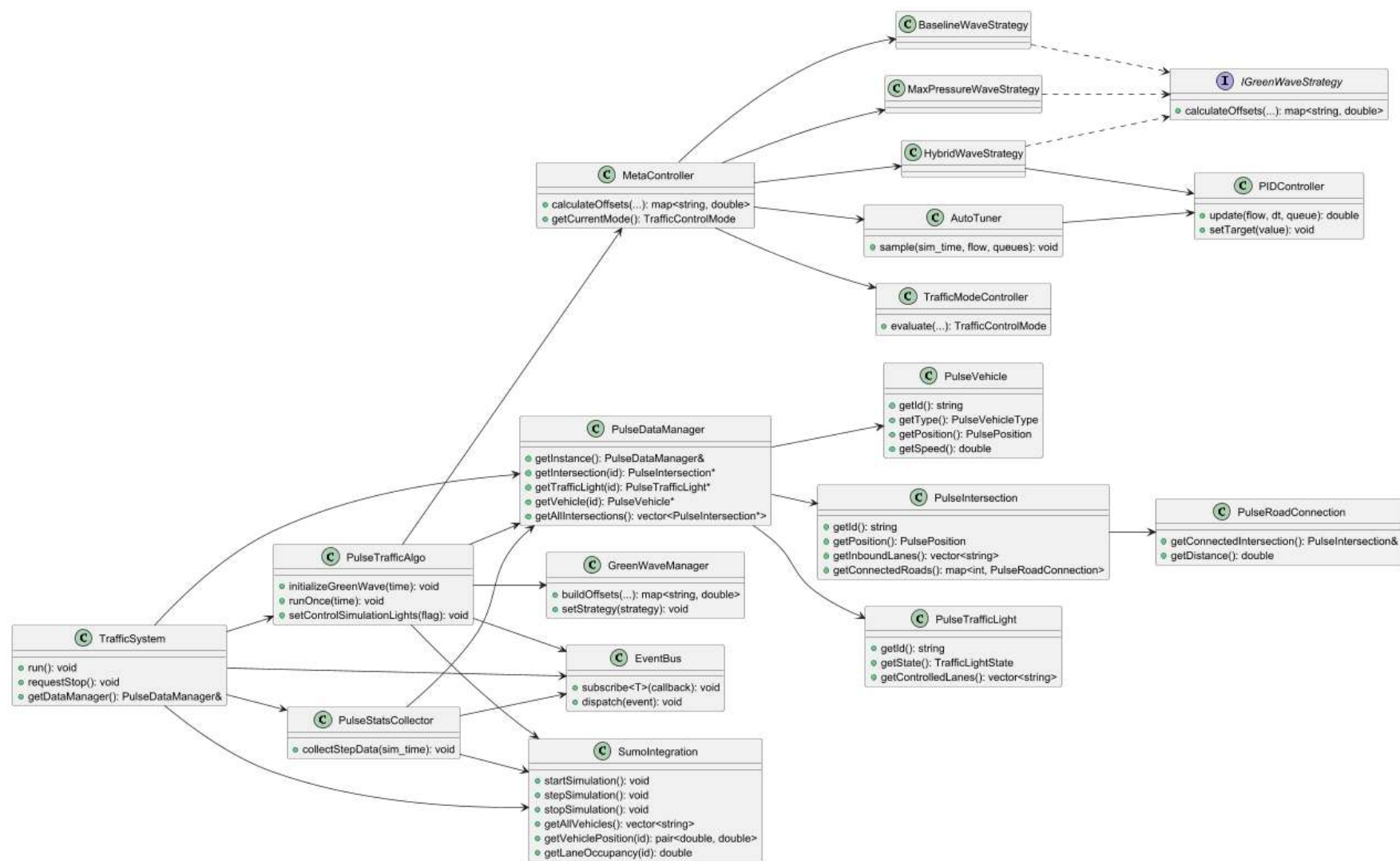


Рис. 2.4. UML-діаграма ключових класів і зв'язків у системі

Окрім загальної архітектури класів, важливо окреслити й структуру даних, які циркулюють у системі під час роботи симуляції. Нижче представлено діаграму, яка ілюструє формат і вміст ключових об'єктів, що беруть участь у процесах зчитування, обробки та накопичення інформації під час роботи алгоритмів.

Дані, які надходять від симулятора, зберігаються в об'єктах типу `PulseVehicle`, `PulseIntersection`, `PulseTrafficLight` та використовуються для формування статистичних звітів. Кожен транспортний засіб має унікальний ідентифікатор, тип і координати. Світлофори зберігають поточний стан (зелений, червоний, жовтий), а перехрестя — список входів, виходів, накопичені показники (кількість авто, час очікування, кількість зупинок). Додатково ведуться агреговані метрики типу `IntersectionStats`, `LaneStats`, `TrafficMetrics`.

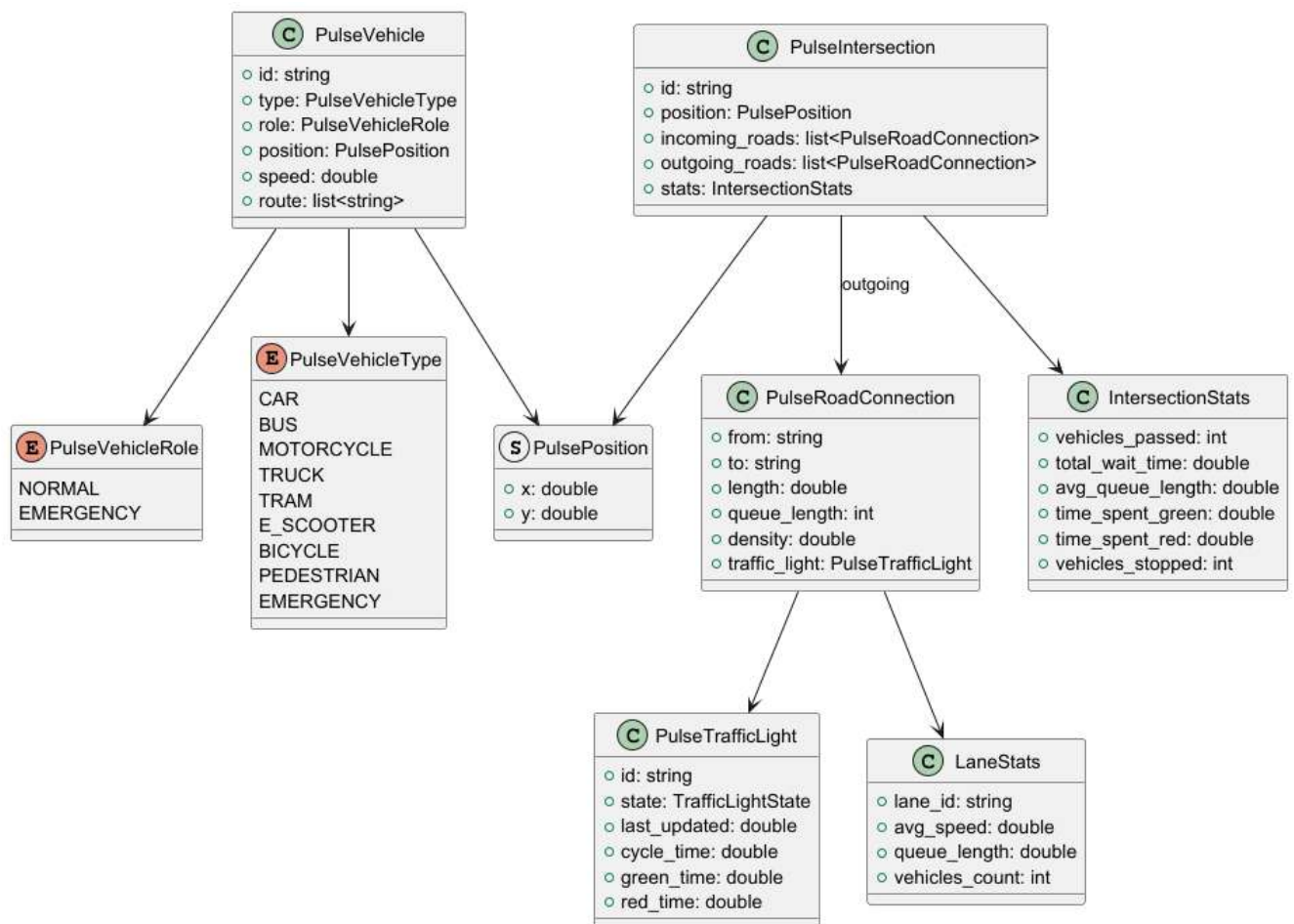


Рис. 2.5. UML-діаграма даних, що використовуються в системі під час симуляції

Такий рівень деталізації дає повне уявлення про те, з якими даними працює система на кожному кроці симуляції. Ці об'єкти постійно оновлюються, що

забезпечує алгоритм актуальними показниками — для прийняття рішень щодо адаптації фаз світлофорів. Дані також використовуються для генерації графіків і метрик після завершення сценарію, дозволяючи дослідити ефективність застосованих стратегій.

Окрім внутрішніх структур системи, у симуляції активно використовуються дані, які надходять із зовнішнього середовища — симулятора дорожнього руху SUMO. Під час кожного симуляційного кроку до системи надходить інформація про транспортні засоби: координати, швидкість, тип, смуги: довжина, кількість авто, середня швидкість, світлофори: активна фаза, стан та перехрестя: входи, виходи, топологія. Ці дані зчитуються через API libsumo і трансформуються у відповідні структури системи: PulseVehicle, LaneStats, PulseTrafficLight, PulseIntersection та PulseRoadConnection.

Такий підхід забезпечує узгоджену інтеграцію між симулятором та внутрішньою логікою системи, дозволяючи використовувати реалістичні вхідні дані під час адаптації алгоритмів. Усі отримані значення оновлюються кожного кроку, накопичуються в менеджері даних і використовуються як основа для розрахунків метрик та прийняття рішень щодо зміни фаз світлофорів.

2.3. Комунікації і послідовність взаємодії об'єктів системи

Взаємодія між компонентами системи побудована на принципах розподілу відповідальності та подіях, що забезпечує ізолюваність модулів, їхню тестованість і підтримку розширень. Кожен з компонентів реалізує свою частину логіки, а комунікація між ними відбувається через глобальну шину подій EventBus, яка підтримує підписку на події відповідно до патерну Observer.

Запуск симуляції відбувається через зовнішній CLI або API-інтерфейс. Клас TrafficSystem приймає команди та запускає симуляцію, створюючи екземпляри відповідних модулів. Через SumoIntegration запускається SUMO у фоновому режимі, зчитуються мережеві об'єкти, транспортні засоби та інші сутності. Після ініціалізації, всі об'єкти зберігаються в PulseDataManager, де з ними надалі працюють алгоритмічні модулі.

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

Клас PulseTrafficAlgo отримує актуальні дані з PulseDataManager, а потім аналізує їх відповідно до обраної стратегії управління. Залежно від сценарію, використовуються різні підходи до побудови зеленої хвилі. Усі обчислення виконуються у внутрішній логіці, після чого нові фази світлофорів застосовуються до симуляції через SumoIntegration.

Збір статистики про ефективність роботи системи виконується поза межами бібліотеки зовнішнім скриптом ScenarioRunner, який запускає симуляцію та обробляє вихідні XML-файли SUMO: tripinfo.xml, summary.xml, queue.xml тощо. Ці метрики зберігаються у CSV-таблицях і візуалізуються для порівняння продуктивності базового та адаптивного режимів.

Нижче наведено UML-діаграму кооперації, яка демонструє, як компоненти системи взаємодіють між собою. До взаємодії додано ролі оператора та адміністратора, що ініціюють відповідні запити через інтерфейс команд.

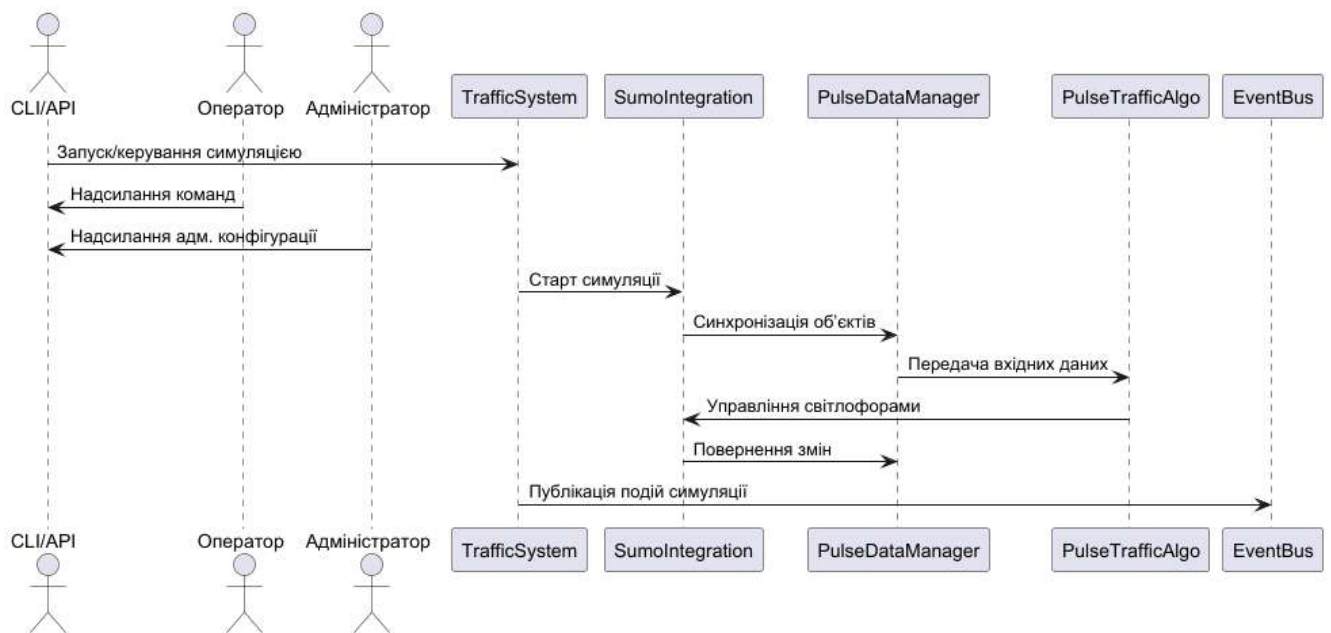


Рис. 2.6. UML-діаграма кооперації компонентів системи

У наступному фрагменті представлено діаграму послідовності, яка ілюструє типовий сценарій запуску симуляції: від отримання команд користувачем, через налаштування конфігурації, до запуску симуляції, обробки станів світлофорів і публікації подій для сторонніх модулів або сервісів.

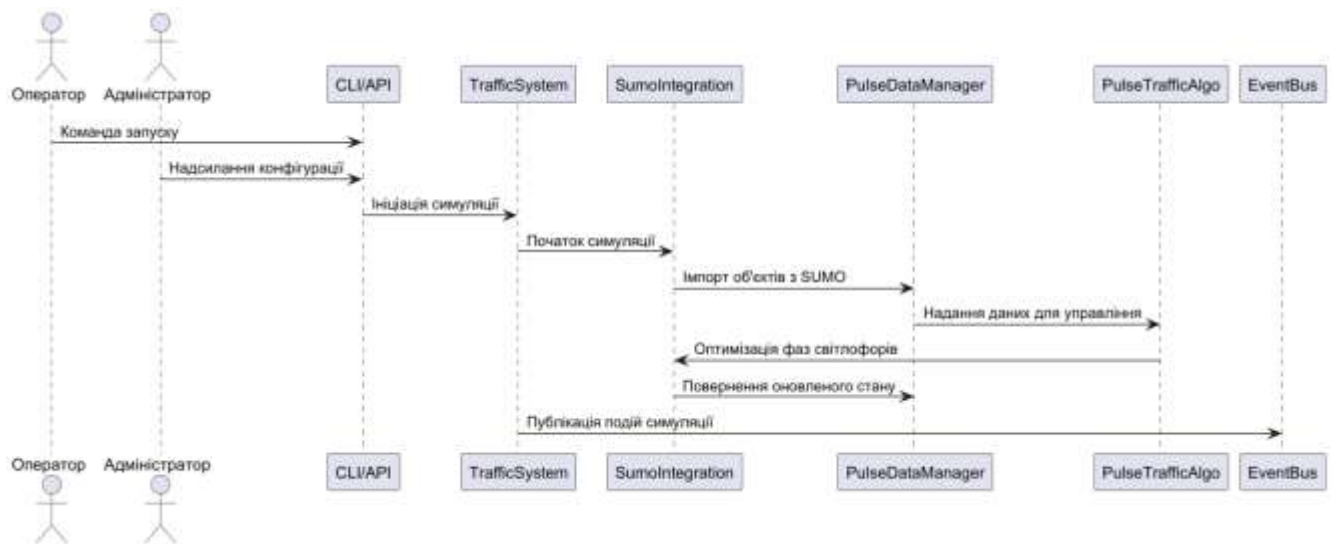


Рис. 2.7. UML-діаграма послідовності системи

Таким чином, система реалізує повністю автономну модель управління симуляцією трафіку, з чітким поділом обов'язків між модулями, підтримкою динамічного вибору алгоритмів і можливістю збору статистики без втручання в основний код. Це дозволяє використовувати її як дослідницький інструмент для порівняння алгоритмів у різних сценаріях, а також як ядро для побудови реального адаптивного контролера.

2.4. Алгоритм роботи та стани програмної системи

Система адаптивного керування світлофорами побудована як багаторівнева структура, в якій модулі взаємодіють на основі подій, конфігурацій та реакції на зовнішні зміни трафіку. Основним координатором є клас TrafficSystem, який ініціалізує компоненти, запускає симуляцію через модуль SumoIntegration, керує обробкою подій через EventBus та викликає кроки алгоритму адаптації.

У типовому сценарії після запуску система переходить у стан Running, у якому виконується основний цикл симуляції. Цей цикл складається з послідовних кроків. На кожному кроці викликається метод stepSimulation() з модуля SumoIntegration, який відповідає за оновлення стану трафіку: позицій транспортних засобів, довжини черг, швидкостей руху та інших параметрів. Дані синхронізуються з об'єктом PulseDataManager, що зберігає актуальні об'єкти

перехресть, транспортних засобів і світлофорів. Оновлення стану мережі є основою для подальшого прийняття рішень в алгоритмі адаптації.

Алгоритм адаптації реалізований у класі PulseTrafficAlgo, який аналізує ситуацію на дорогах і визначає необхідний зсув фаз світлофорів у кожному перехресті. Обрана стратегія залежить від навантаження мережі, яке визначається через модуль TrafficModeController. За логіку перемикання стратегій відповідає MetaController, який реагує на динамічні зміни в трафіку. Залежно від ситуації обирається одна з трьох реалізацій стратегії: BaselineWaveStrategy, HybridWaveStrategy або MaxPressureWaveStrategy.

Нижче наведено UML-діаграму алгоритму роботи системи, яка демонструє послідовність етапів: запуск, симуляція, оновлення даних, адаптація режимів і поширення подій:

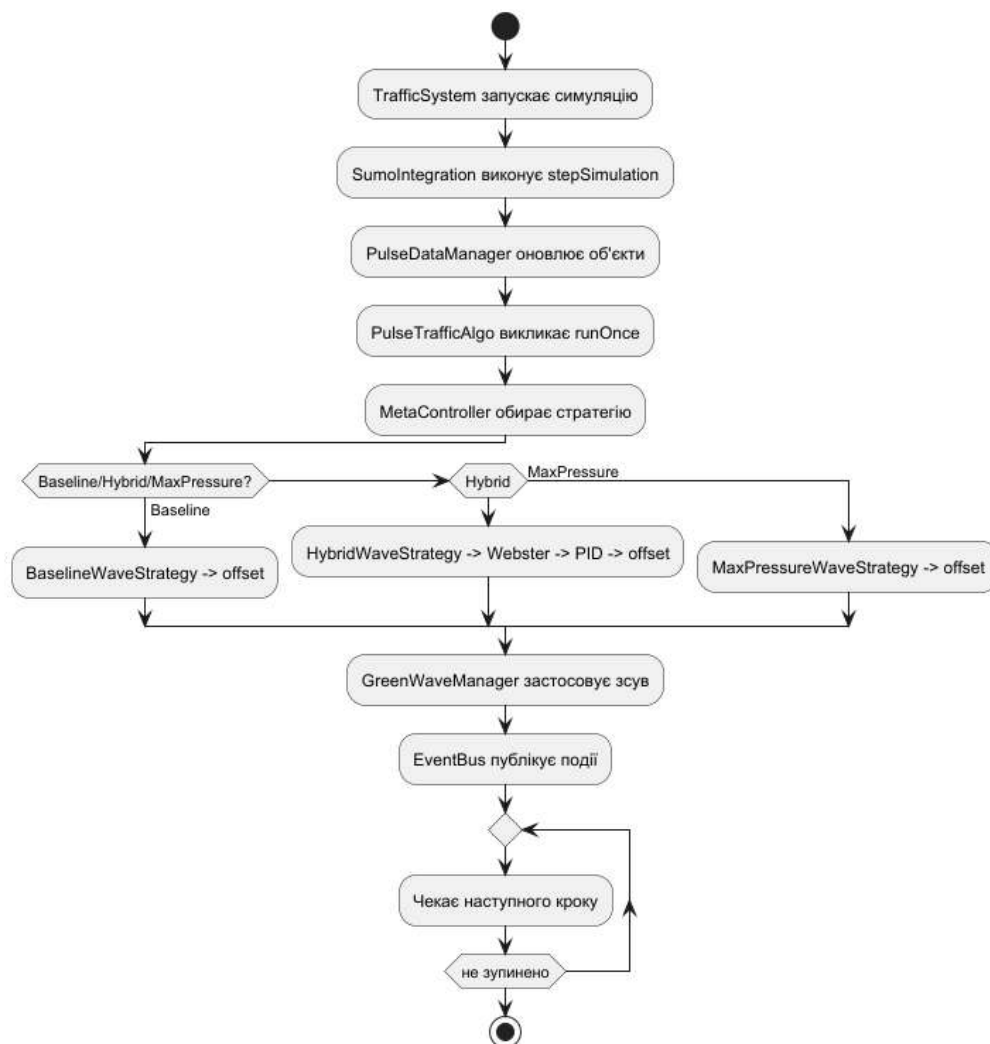


Рис. 2.8. UML-діаграма алгоритму роботи системи

У разі низького навантаження застосовується стратегія BaselineWaveStrategy, яка синхронізує світлофори відповідно до цільової швидкості руху. Зсув фази для кожного перехрестя i обчислюється за формулою:

$$offset_i = \frac{\sum_{j=0}^{i-1} distance(j, j + 1)}{v_{target}}$$

де $distance(j, j + 1)$ – відстань між суміжними перехрестями, а v_{target} – цільова швидкість потоку в метрах за секунду.

Відповідний фрагмент реалізації в коді:

```
double offset_i = 0.0;
for (size_t j = 0; j < i; ++j) {
    offset_i += getDistance(j, j + 1);
}
offset_i /= target_speed_m_s;
```

При середньому навантаженні система переходить у режим HybridWaveStrategy. У цьому випадку базова довжина циклу розраховується за класичною формулою Вебстера:

$$C = \frac{1.5 * L + 5}{1 - \sum y_i}$$

де L – втрати часу на фазу(жовтий сигнал і між фазна пауза), а y_i – ступінь завантаженості i фази, визначена як відношення потоку до пропускної здатності.

У коді розрахунків реалізовано наступним чином:

```
double sum_y = 0.0;
for (const auto& phase : signal_phases) {
    sum_y += phase.load_ratio();
}
double cycle_time = (1.5 * lost_time + 5.0) / (1.0 - sum_y);
```

Для підвищення точності керування використовується PID-регулятор, який коригує цикл залежно від динаміки черг. Похибка $e(t)$ є різницею між фактичним та бажаним значенням довжини черги. PID-регулювання виконується за формулою:

$$\Delta C = K_p * e(t) + K_i * \int e(t)dt + K_d * \frac{de(t)}{dt}$$

де K_p, K_i, K_d – коефіцієнти підсилення відповідно для пропорційної, інтегральної та диференціальної частини.

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

Фрагмент реалізації:

```
double error = current_queue - target_queue;
integral += error * dt;
double derivative = (error - previous_error) / dt;
double delta = Kp * error + Ki * integral + Kd * derivative;
cycle_time += delta;
```

У ситуаціях критичного навантаження, коли фіксуються довгі черги або інтенсивний потік, активується стратегія MaxPressureWaveStrategy. Тут перехрестя з найменшою чергою отримують більший зсув у фазі, що дозволяє зменшити затримки:

$$offset_i = \left(1 - \frac{queue_i}{\sum queue}\right) * C$$

де $queue_i$ – довжина черги на вході до i -го перехрестя, а C – довжина поточного циклу.

Відповідна реалізація:

```
double offset_i = (1.0 - (queue_i / total_queue)) * cycle_time;
```

Система підтримує три основні стани: Running, Paused і Stopped. Перший відповідає за активну фазу симуляції, другий дозволяє тимчасово призупинити обчислення, а третій завершує сценарій з вивантаженням усіх результатів. Ці стани дозволяють інтегрувати систему в автоматизовані pipeline аналізу та CI/CD.

Під час симуляції система публікує низку подій, зокрема SimulationSteppedEvent, StatsUpdatedEvent, TrafficModeChangedEvent, що дозволяє стороннім модулям реагувати на зміни. Це реалізовано за допомогою патерну Observer, який забезпечує слабе зв'язування між модулями та сприяє розширюваності архітектури системи.

Логіка вибору режиму керування та адаптації показана на наступній діаграмі, яка ілюструє внутрішню логіку MetaController, TrafficModeController, PID-регулювання та модуля AutoTuner, який періодично оновлює порогові значення та параметри адаптації.

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				34
Змн.	Арк.	№ докум.	Підпис	Дата		

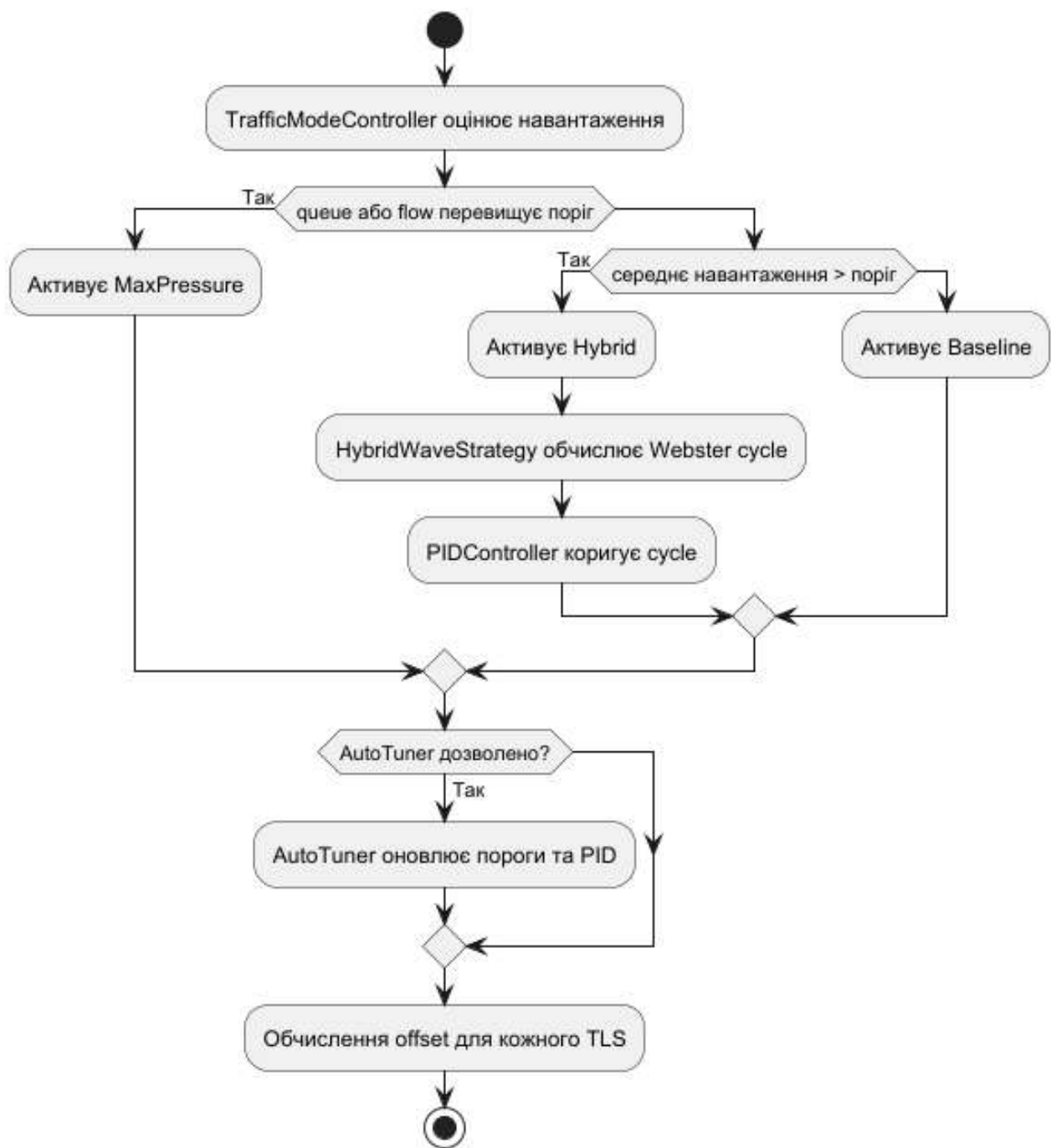


Рис. 2.9. UML-діаграма роботи модулів алгоритму

Усі алгоритмічні формули, включно з методами Вебстера, MaxPressure та PID-регулювання, базуються на класичних джерелах, таких як [24], [25], [26].

Таким чином, описаний алгоритм реалізує повний цикл управління світлофорами: від збору даних у симуляції – до прийняття адаптивних рішень і оновлення фаз, забезпечуючи ефективну, масштабовану й розширювану модель адаптивного керування дорожнім рухом.

Висновки до другого розділу

У другому розділі було сформовано архітектуру програмної системи адаптивного керування світлофорами, розроблену з урахуванням принципів об'єктно-орієнтованого програмування. Визначено основні класи системи, їхню структуру, ролі та зв'язки між ними. Описано механізми комунікації, синхронізації та адаптації алгоритмів під час симуляції трафіку.

Зовнішнє середовище моделювання інтегровано в систему через окремий модуль, а взаємодія між компонентами забезпечується за допомогою подій. Для ілюстрації логіки роботи розроблено діаграми класів, кооперації, послідовності та станів системи.

Сформована архітектура закладає основу для подальшої розробки функціоналу, спрощує впровадження нових стратегій і забезпечує гнучке масштабування системи.

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				36
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ДЕМОНСТРАЦІЯ РОБОТИ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1. Інструкція з використання програми

Розроблене програмне забезпечення передбачає запуск симуляції дорожнього руху на основі заданих сценаріїв із подальшим порівнянням базового режиму та адаптивного алгоритму керування світлофорами. Програма є кросплатформеною і підтримує запуск на Linux та macOS. Усі основні компоненти збираються командою *make build*, яка компілює проект разом із модулями симуляції, алгоритмами керування та бібліотекою тестування.

Сценарії запуску задаються через параметри командного рядка. Для прикладу, запуск симуляції з адаптивним алгоритмом для середнього навантаження відбувається за допомогою команди:

```
./ScenarioRunner --scenario "Medium Traffic S" \  
    --sumoCfg config/sumo/scenarios/s/medium_traffic/osm.sumocfg \  
    --netCfg config/sumo/scenarios/s/medium_traffic/osm.net.xml \  
    --custom
```

У системі також реалізовано графічний режим через SUMO GUI. Команда *make run-scenarios-gui* виконує всі сценарії з візуалізацією. Цей режим дає змогу бачити реальну зміну фаз світлофорів, утворення черг, перемикання стратегій алгоритму.



Рис. 3.1. Візуалізація роботи симуляції в SUMO GUI

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

```
Project -
Terminal
[14:12:39.525] [Debug] Building schedules for 5 w.corridors.
[14:12:39.925] [Debug] PID: 4974.0.000 lat:100.000 dev:0.071 * A=2.000
[14:12:39.925] [Debug] Cycle now 65.1431216216216 * (Minuter=62.1431216216216, PID A=2)
[14:12:39.935] [Debug] Cycle now 66.1431216216216 * (Minuter=62.1431216216216, PID A=2)
[14:12:39.935] [Debug] Cycle now 67.1431216216216 * (Minuter=62.1431216216216, PID A=2)
[14:12:39.935] [Debug] Cycle now 68.1431216216216 * (Minuter=62.1431216216216, PID A=2)
[14:12:39.935] [Debug] Cycle now 69.1431216216216 * (Minuter=62.1431216216216, PID A=2)
[14:12:40.492] [Info] Node switch: Hybrid -> Normal
[14:12:40.498] [Debug] Building schedules for 5 w.corridors
[14:12:41.454] [Debug] [DataServer] time 25000.000 * lat:0.000 lon:0.012 speed:24 speed:48
[14:12:41.492] [Info] Node switch: Normal -> Hybrid
[14:12:41.993] [Debug] Building schedules for 5 w.corridors
[14:12:41.992] [Debug] PID: 4974.0.000 lat:100.000 dev:0.030 * A=2.000
[14:12:41.993] [Debug] Cycle now 66.1431216216216 * (Minuter=62.1431216216216, PID A=2)
[14:12:41.993] [Debug] Cycle now 67.1431216216216 * (Minuter=62.1431216216216, PID A=2)
[14:12:41.993] [Debug] Cycle now 68.1431216216216 * (Minuter=62.1431216216216, PID A=2)
[14:12:41.993] [Debug] Cycle now 69.1431216216216 * (Minuter=62.1431216216216, PID A=2)
[14:12:41.992] [Debug] Cycle now 70.1431216216216 * (Minuter=70.1431216216216, PID A=2)
[14:12:42.243] [Debug] Building schedules for 5 w.corridors
[14:12:42.243] [Debug] Cycle now 67.1431216216216 * (Minuter=70, PID A=2)
[14:12:42.243] [Debug] Cycle now 68.1431216216216 * (Minuter=70, PID A=2)
[14:12:42.243] [Debug] Cycle now 69.1431216216216 * (Minuter=70, PID A=2)
[14:12:42.243] [Debug] Cycle now 70.1431216216216 * (Minuter=70, PID A=2)
[14:12:42.243] [Debug] Cycle now 71.1431216216216 * (Minuter=70, PID A=2)
[14:12:42.767] [Info] Node switch: Hybrid -> Normal
[14:12:42.767] [Debug] Building schedules for 5 w.corridors
[14:12:43.128] [Info] Node switch: Normal -> HighPressure
[14:12:43.128] [Debug] Building schedules for 5 w.corridors
[14:12:44.190] [Info] Node switch: HighPressure -> Normal
[14:12:44.190] [Debug] Building schedules for 5 w.corridors
[14:12:44.128] [Info] Node switch: Normal -> HighPressure
[14:12:44.128] [Debug] Building schedules for 5 w.corridors
[14:12:44.688] [Info] Node switch: HighPressure -> Normal
[14:12:44.688] [Debug] Building schedules for 5 w.corridors
[14:12:45.023] [Info] [DataServer] time 25000.000 * lat:0.010 lon:0.014 speed:28 speed:48
Simulation ended at time: 4009.00
Performance:
Duration: 128.75s
Peak CPU Memory: 31.12MB
FPS: 4407.338175
Vehicles:
Inserted: 1668
Routing: 42
Waiting: 0
Emergency Braking: 1
Statistics (Avg of 1000):
RouteLength: 3436.75
Speed: 9.94
Corruption: 381.49
WaitingTime: 46.82
TimeLoss: 125.38
traffic-simulator -> Run -> tests -> include -> Return -> h TrafficSimulator
```

У результаті виконання формуються вихідні дані у вигляді CSV-файлу *analysis/comparison.csv*, у якому зберігаються метрики роботи системи. Для побудови графіків і текстового звіту використовується команда *make report-metrics*, яка генерує аналітичний файл *report.md* і зображення у форматі PNG.

Першим показником є середній час у дорозі – він дозволяє оцінити, наскільки швидко транспорт може пересуватися мережею при використанні різних стратегій керування. По осі X відображено назви сценаріїв, по осі Y – середній час у секундах, отриманий із поля `avg travel time`.

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				38
Змн.	Арк.	№ докум.	Підпис	Дата		

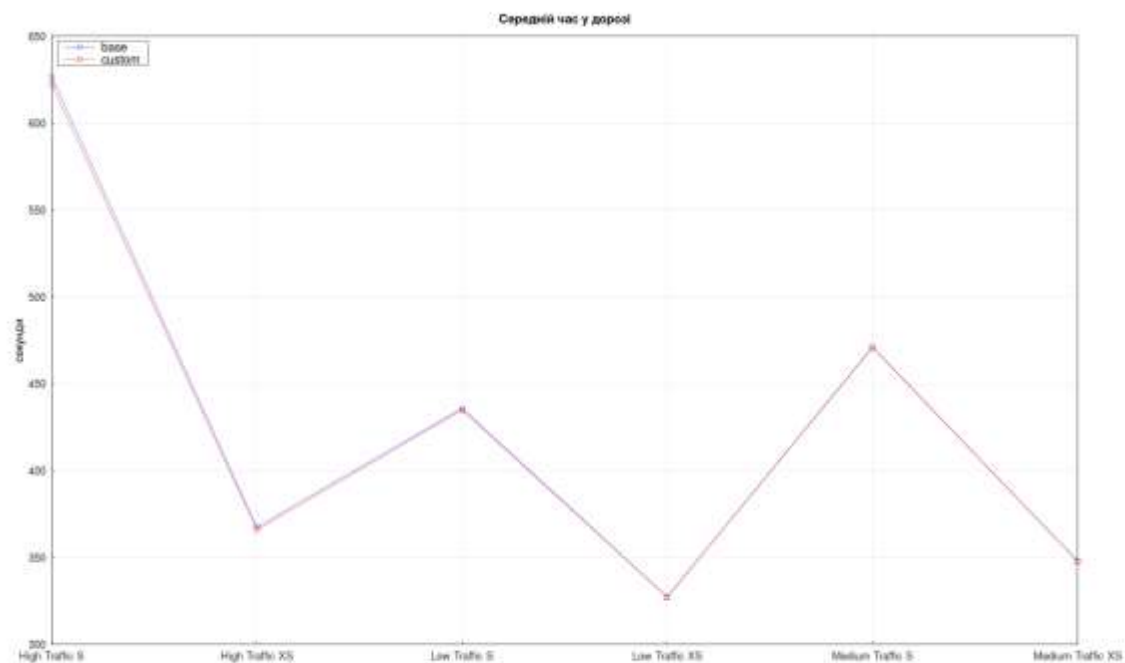


Рис. 3.3. Графік середнього часу в дорозі

Наступною метрикою є середній час очікування, який показує, як часто і надовго транспорт зупиняється на світлофорах. Осі графіка аналогічні, дані взяті з поля `avg_waiting_time`.

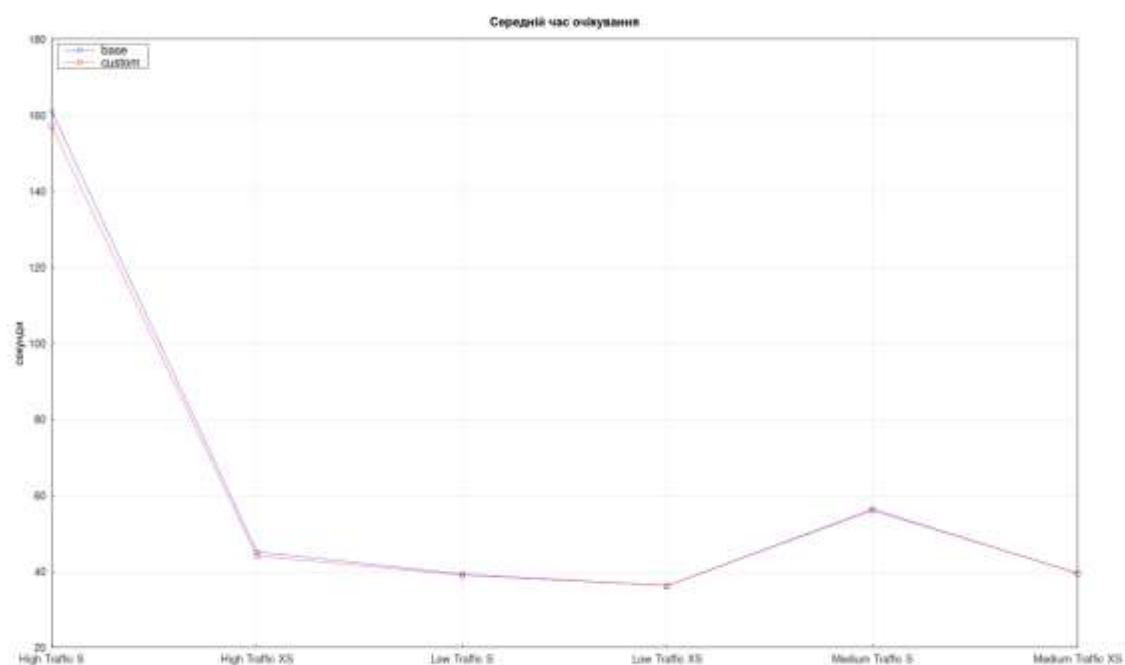


Рис. 3.4. Графік середнього часу очікування на перехрестях

Оцінка пропускної здатності демонструє загальну кількість транспортних засобів, які пройшли мережу за симульований період. Цей показник витягується з поля throughput CSV-файлу. По осі Y – кількість авто, що завершили маршрут.

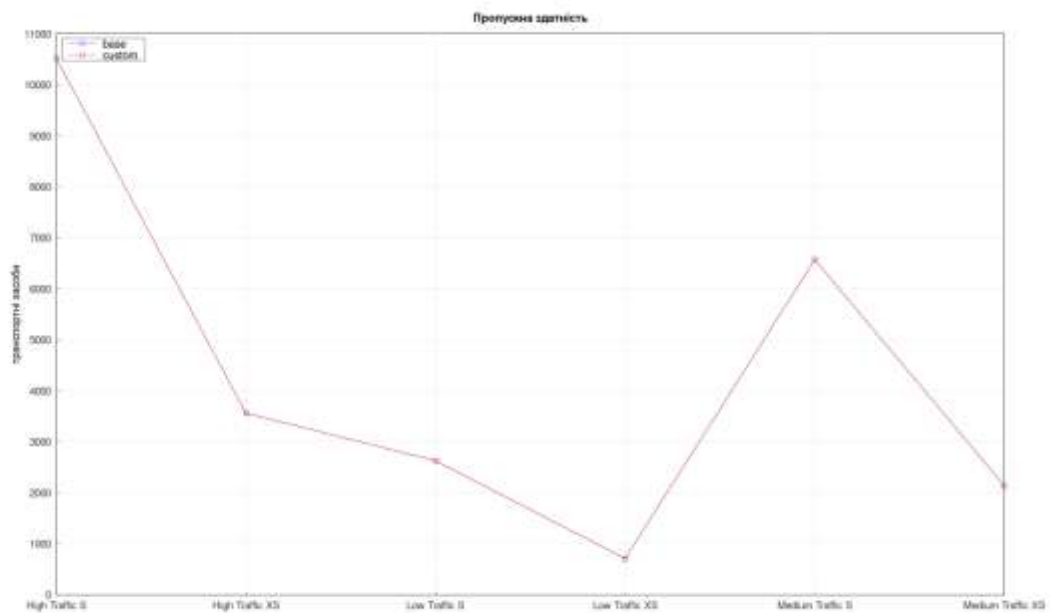


Рис. 3.5. Графік пропускної здатності транспортної мережі

Важливим параметром є пікова довжина черг. Зменшення цього значення свідчить про зниження локальних перевантажень. Дані взято з колонки queue_reak, по осі Y – довжина черги у метрах.

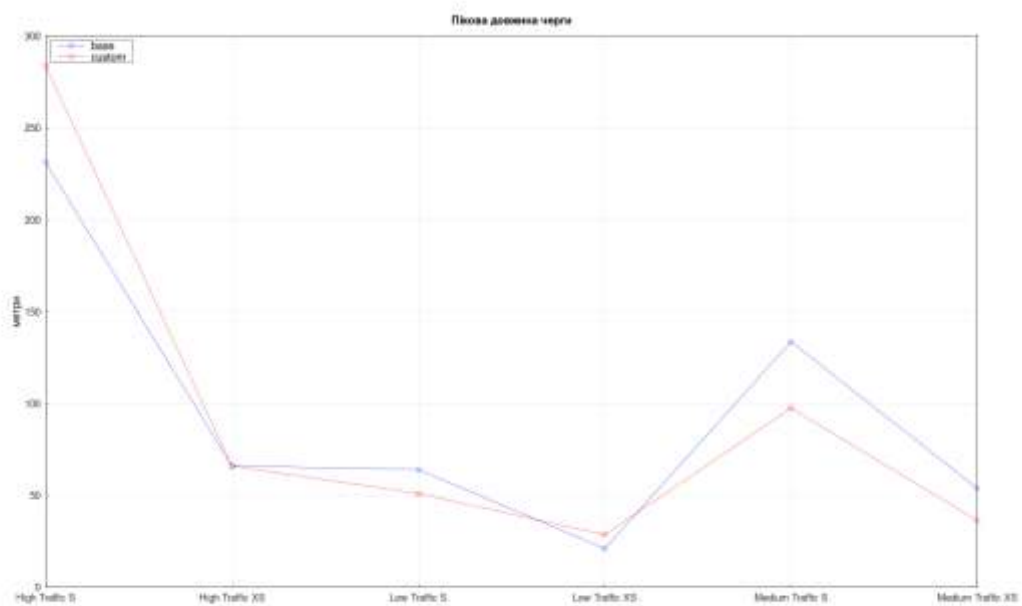


Рис. 3.6. Графік максимальної зафіксованої довжини черг

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

Ще одним критичним показником є загальні втрати часу. Це сумарний обсяг часу, який було втрачено в заторах та під час зупинок, розрахований за полем `time_loss`.

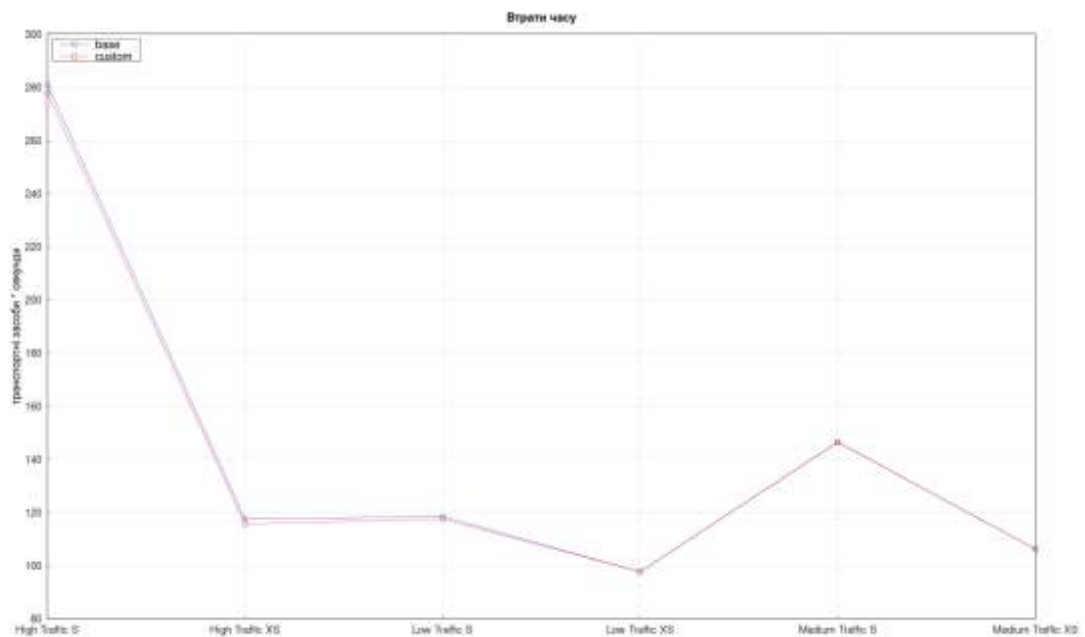


Рис. 3.7. Графік втрат часу в транспортному потоці

Стабільність руху також оцінюється за кількістю зупинених транспортних засобів під час симуляції. Величина взята з колонки `halted_peak`.

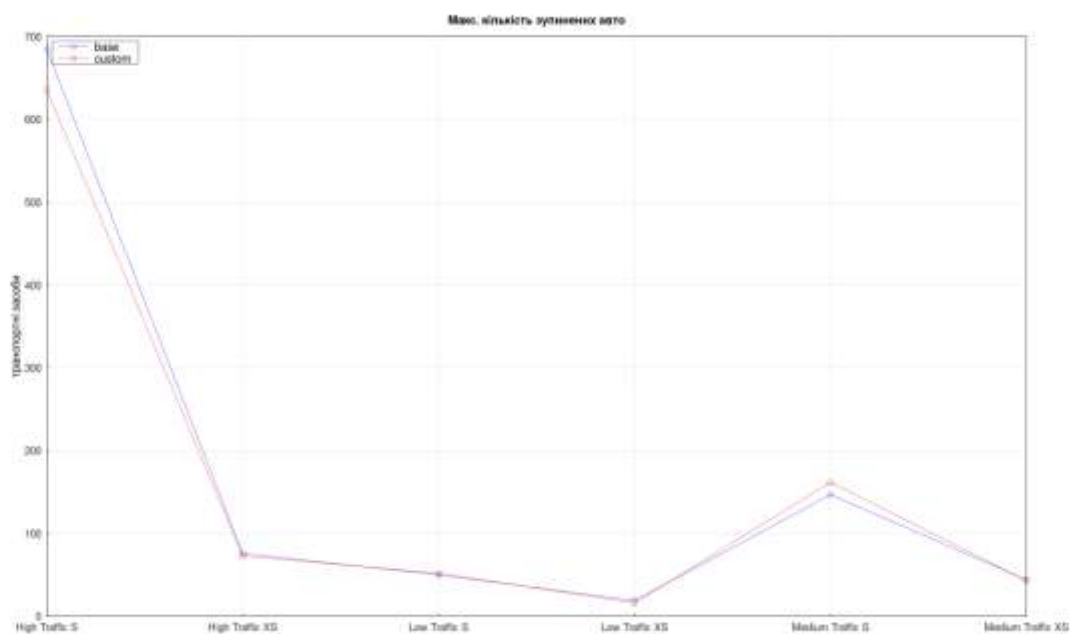


Рис. 3.8. Графік кількості зупинених авто

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

Показник середньої довжини черги дозволяє оцінити загальну ефективність перехресть. Цей графік побудовано за полем `avg_queue_length`.

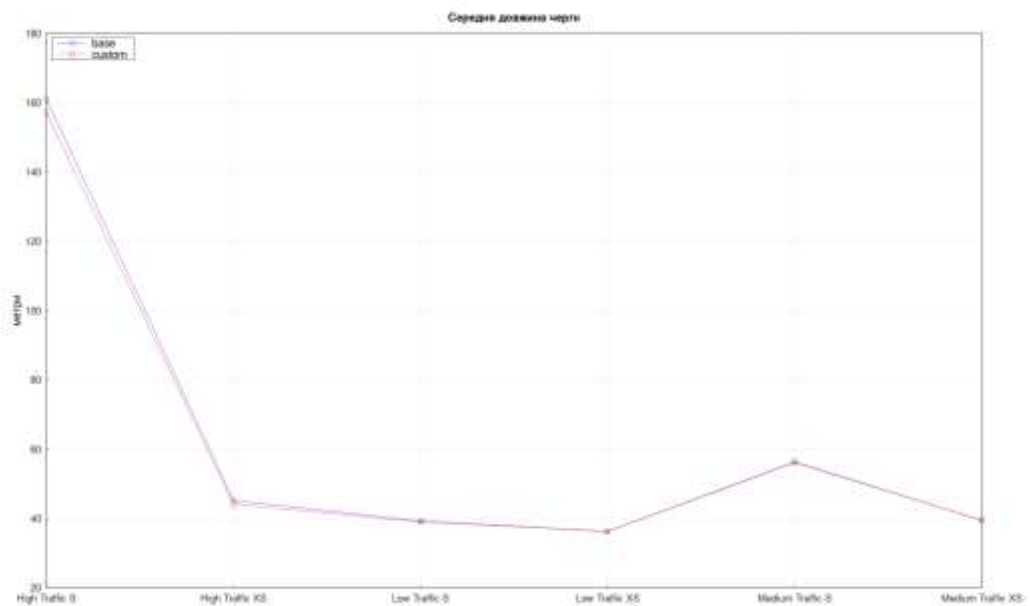


Рис. 3.9. Графік середньої довжини черги

Щільність трафіку на мережі відображає рівномірність завантаження. Пікова щільність, яка береться з поля `density_peak`, використовується для виявлення вузьких місць. Значення на осі Y подано у відсотках від максимальної пропускної здатності смуги.

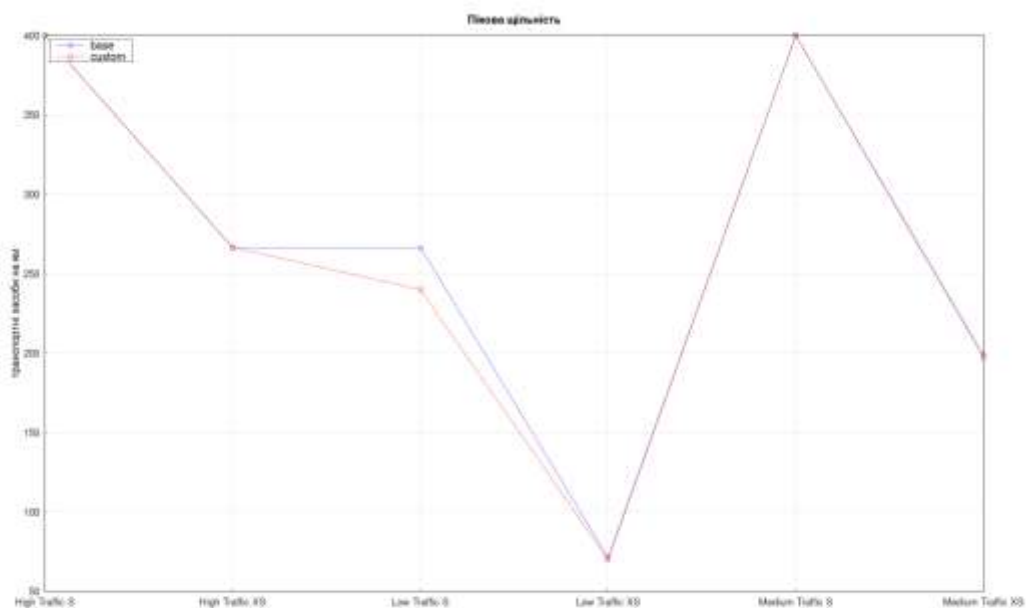


Рис. 3.10. Графік максимальної щільності транспортного потоку

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

Останнім показником є середня швидкість руху, яка зростає при ефективному розподілі фаз світлофорів. Дані взяті з поля `avg_speed`, значення подано у метрах за секунду.

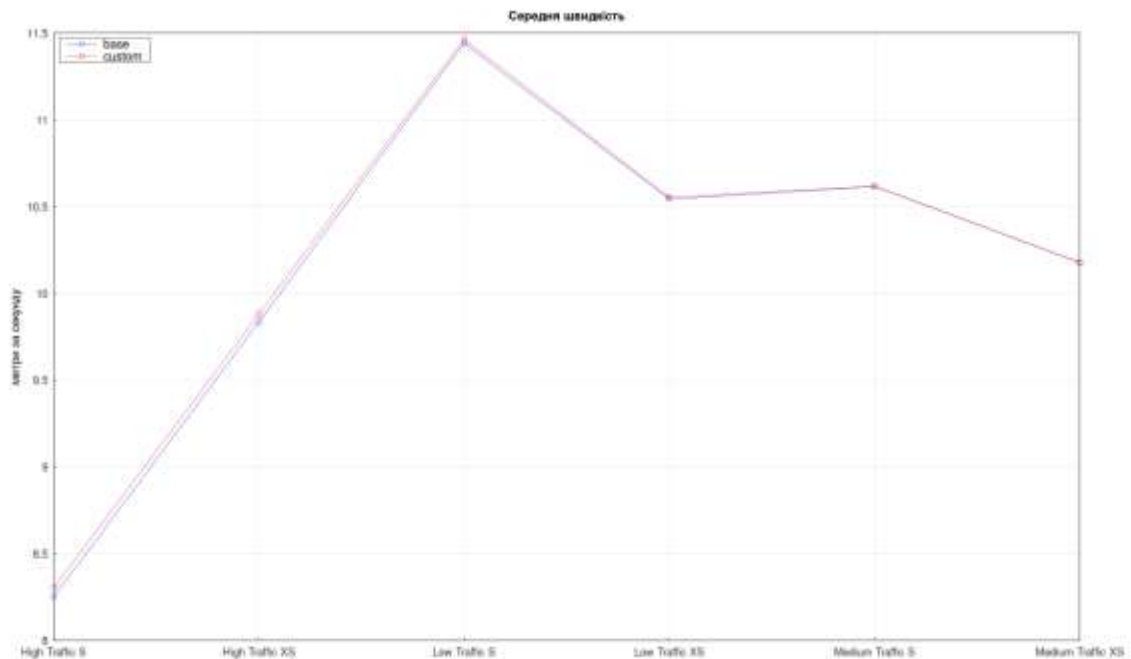


Рис. 3.11. Графік середньої швидкості транспорту

Для всіх згаданих метрик графіки створено автоматично після запуску симуляцій і подальшої обробки сценаріїв. Усі результати використовуються для кількісного порівняння роботи адаптивної стратегії проти фіксованої схеми керування.

Додатково, для кожного зміни у проєкті автоматично генерується документація за допомогою системи Doxygen. Вона охоплює всі класи, методи, структури, перерахування та інтерфейси. Побудова документації відбувається через CI/CD у GitHub, що забезпечує її актуальність. Результат доступний як локально у каталозі `library/html/index.html`, так і у вигляді HTML-сторінки на GitHub Pages.

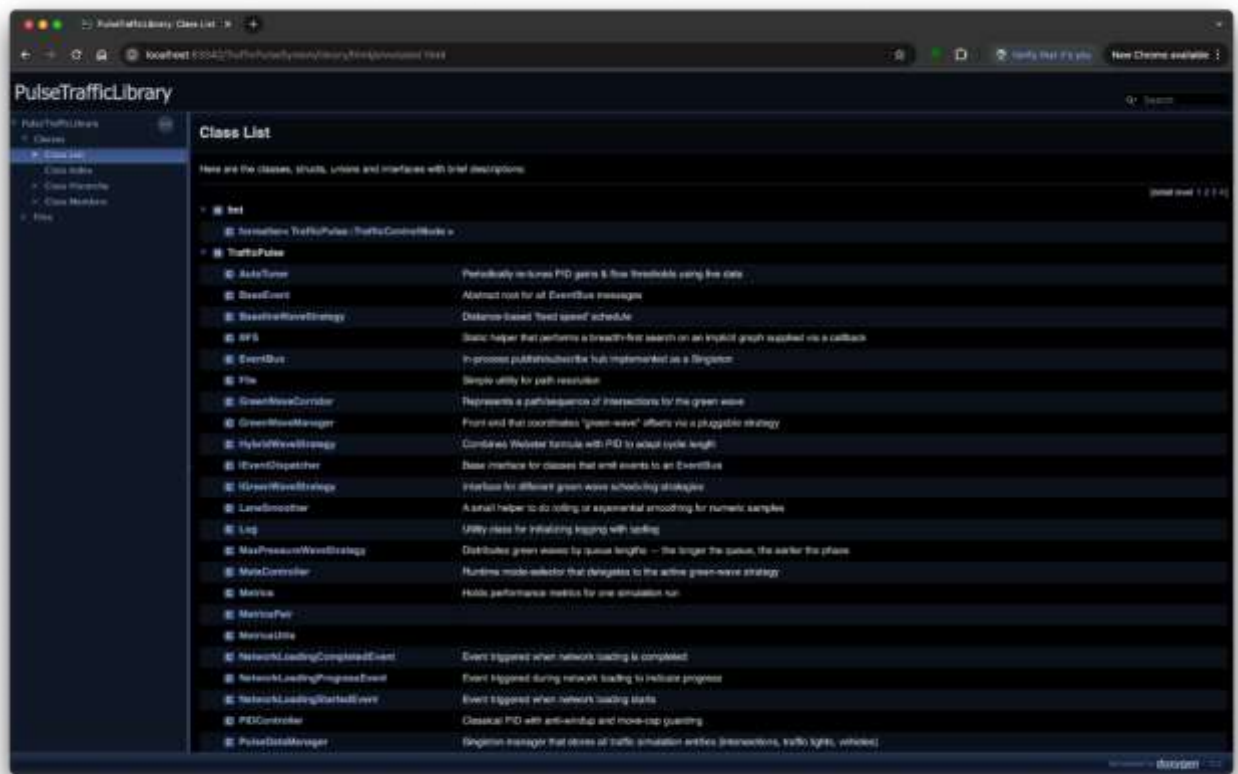


Рис. 3.12. Головна сторінка згенерованої Doxygen-документації

3.2. Результати тестування і перевірка алгоритму

Для підтвердження коректності роботи системи реалізовано повноцінний набір модульних і інтеграційних тестів. Основна увага приділяється стабільності роботи адаптивного алгоритму, правильності перемикання фаз світлофорів, збору статистики, зміні режимів керування, а також дотриманню принципів ізоляції відповідальності між компонентами.

Контролер вибору режиму TrafficModeController тестується на правильність логіки перемикання між режимами керування на основі рівня завантаження та довжини черг. Наприклад, за відсутності черг обирається базовий режим, а при перевищенні порогу активується стратегія MaxPressure:

```
EXPECT_EQ(c.evaluate(0.7, {0.0}, 1.0), TrafficControlMode::Baseline);
EXPECT_EQ(c.evaluate(0.7, {11.0}, 2.0), TrafficControlMode::MaxPressure);
```

Менеджер об'єктів PulseDataManager перевіряється на унікальність ID при додаванні перехресть, світлофорів і транспортних засобів. Повторна спроба додати об'єкт із вже існуючим ID має викликати виняток:

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

```
EXPECT_THROW(m.addIntersection(std::make_unique<PulseIntersection>("intA",
PulsePosition{9, 9})), PulseException);
```

Алгоритм `PulseTrafficAlgo` тестується на коректне оновлення фаз світлофорів. Після симуляції жоден світлофор не повинен залишатися в некоректному стані:

```
EXPECT_NE(data_manager->getTrafficLight("JunctionA")->getState(),
TrafficLightState::YELLOW);
```

Модуль збору статистики `PulseStatsCollector` забезпечує накопичення даних про рух транспорту, час очікування, кількість зупинок. Тестування гарантує, що значення є додатними і відповідають очікуваній поведінці:

```
EXPECT_GE(m_stats->getLaneWaitingCount("edgeA_0"), 0);
EXPECT_GE(m_stats->getIntersectionStats("JunctionA").vehicles_passed, 0);
```

Також реалізовано інтеграційні тести для перевірки переходів між станами системи. Клас `TrafficSystem` проходить послідовність запуску, зупинки та очищення пам'яті:

```
EXPECT_TRUE(has_green || has_red || has_yellow);
```

Усі тести виконуються за допомогою команди *make test-library*, а також автоматично в CI-процесі. Це забезпечує постійну перевірку відповідності логіки очікуваній поведінці системи, дозволяючи уникати регресій у реалізації.

Нижче представлено рисунок успішного проходження всіх тестів: 25 із 25, який підтверджує повну відповідність очікуванням:

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

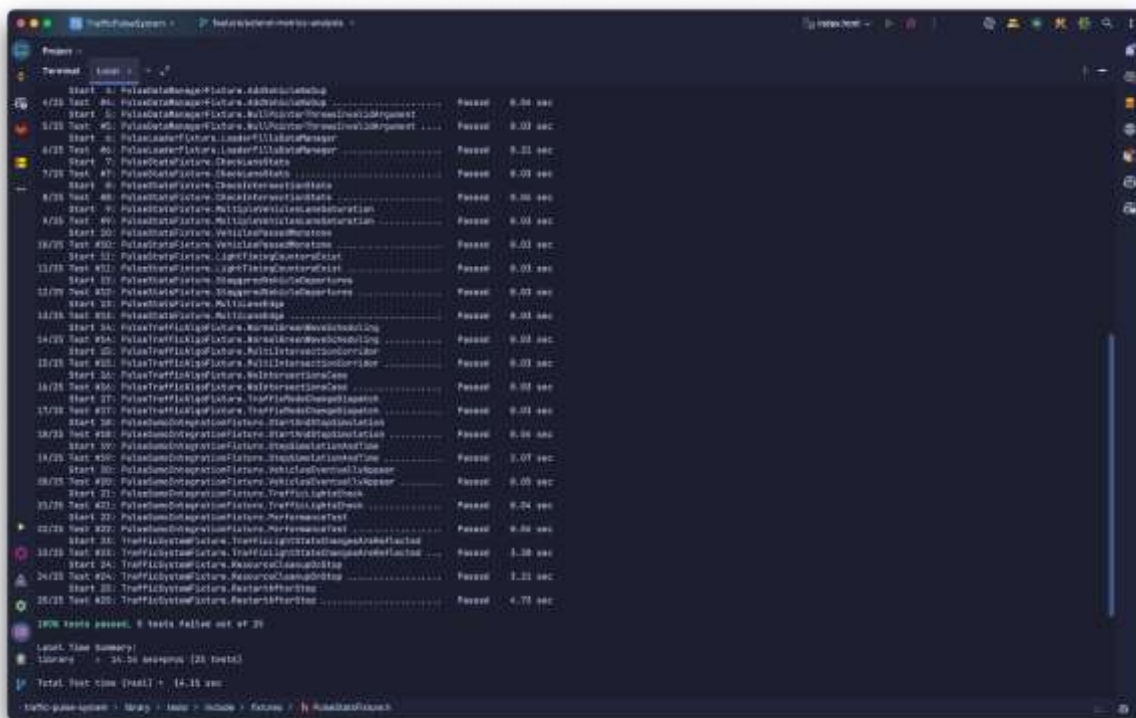


Рис. 3.11. Результат тестування у терміналі з успішним проходженням усіх тестів

Під час тестування були також виявлені та виправлені окремі проблеми: некоректна ініціалізація станів світлофорів у конфігураціях із двома перехрестями, а також неточне оновлення метрик очікування на порожніх смугах. Після внесення змін і повторного запуску тести пройшли успішно, що зафіксовано у СІ.

Таким чином, реалізована система має якісне тестове покриття, що охоплює всі критичні сценарії її роботи. Тестування підтверджує стабільність роботи алгоритмів, узгодженість взаємодії модулів і правильність поведінки системи в різних режимах.

Висновки до третього розділу

У третьому розділі було детально продемонстровано порядок використання розробленої системи адаптивного керування світлофорами, способи її запуску, а також результати симуляції з використанням графічного інтерфейсу SUMO. Надано приклади команд запуску, описано формати вхідних параметрів і показано способи аналізу результатів через побудовані графіки.

Окрему увагу приділено оцінці ефективності реалізованих алгоритмів. На основі метрик — середнього часу в дорозі, затримок, пропускної здатності,

		Бабушко А.С.						Арк.
		Власенко О.В.						46
Змн.	Арк.	№ докум.	Підпис	Дата				

довжини черг і щільності трафіку — доведено переваги адаптивного підходу в порівнянні з базовим сценарієм. Візуальні результати підтверджують зменшення заторів, зростання швидкості транспорту і покращення розподілу навантаження.

Також розглянуто систему автоматичного тестування і документації. Unit та інтеграційні тести забезпечують верифікацію логіки, цілісності даних і правильності взаємодії компонентів. Генерація документації через Doxygen гарантує прозорість структури системи та зручність подальшої підтримки.

Таким чином, продемонстровано повну функціональність системи, підтверджену як кількісно — через метрики, так і якісно — через логіку тестів, візуалізацію та автоматизовану документацію.

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Під час виконання кваліфікаційної роботи вдалося набути практичних навичок моделювання, проектування, реалізації та тестування програмного забезпечення для задач інтелектуального керування дорожнім рухом. Результатом стала система адаптивного керування світлофорами, здатна взаємодіяти із симулятором SUMO, приймати динамічні рішення на основі аналізу навантаження та забезпечувати пріоритетний проїзд транспорту за допомогою реалізованого алгоритму «зеленої хвилі».

У першому розділі було сформульовано загальну концепцію проекту та його актуальність, проаналізовано існуючі рішення, зокрема SCOOT, SCATS, системи на базі ML/AI та хмарні сервіси. Визначено їхні переваги й обмеження у контексті міських транспортних систем. Проведено детальний огляд технологій, обґрунтовано вибір мови програмування C++20, симулятора SUMO, бібліотек для логування, конфігурації та візуалізації. Запропоновано мікросервісну архітектуру, яка дозволяє масштабувати систему та підтримувати гнучку взаємодію між її модулями.

У другому розділі описано основні сценарії використання системи, з урахуванням різних ролей користувачів. Побудовано діаграми взаємодії, комунікації та алгоритмічної логіки, які дають змогу повніше уявити динаміку роботи СКС. Здійснено формалізацію об'єктно-орієнтованої структури системи, реалізовано ключові архітектурні патерни, зокрема Singleton, Observer, Proxy та Factory. Описано структуру класів, їх відповідальність та механізми взаємодії з симуляційним середовищем. Подано формули, які лежать в основі керування фазами світлофорів у різних режимах.

У третьому розділі продемонстровано порядок використання програми: збірка, запуск сценаріїв, аналіз результатів. Представлено приклади запуску з командного рядка, побудовано серію графіків для оцінки метрик продуктивності та ефективності алгоритму. На основі середнього часу поїздки, очікування, втрат часу та довжини черг було проведено порівняння з базовим режимом, що підтвердило

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

переваги адаптивної системи. Крім того, система протестована автоматизованими unit та інтеграційними тестами, а її документація генерується автоматично за допомогою Doxygen.

Загалом виконання проекту дало змогу зміцнити й удосконалити навички моделювання архітектури, розробки високопродуктивного коду, інтеграції з зовнішніми інструментами, а також забезпечення стабільності та надійності ПЗ через тестування. Усі поставлені завдання були реалізовані у повному обсязі, що підтверджує здатність самостійно проектувати, реалізовувати та аналізувати роботу складних програмних систем у сфері інтелектуального управління міською інфраструктурою.

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бабушко А. С., Власенко О. В., Янчук В. М. Моделювання та візуалізація «зеленої хвилі» в SUMO для адаптивного керування міським трафіком // Тези доповідей XV Міжнародної науково-технічної конференції «Інформаційно-комп'ютерні технології», 28–29 березня 2025 р. – Житомир: Державний університет «Житомирська політехніка», 2025. – С. 40–42.
2. Eclipse SUMO – Simulation of Urban Mobility [Електронний ресурс]. – Режим доступу: <https://www.eclipse.org/sumo/>
3. SUMO Documentation – Algorithm descriptions, outputs and API [Електронний ресурс]. – Режим доступу: <https://sumo.dlr.de/docs/>
4. Doxygen : Automatic C++ documentation generator [Електронний ресурс]. – Режим доступу: <https://www.doxygen.nl/>
5. Matplotlib++ : C++ plotting library based on matplotlib [Електронний ресурс]. – Режим доступу: <https://github.com/alandefreitas/matplotlibplusplus>
6. PugiXML : Light weight, simple and fast XML parser for C++ [Електронний ресурс]. – Режим доступу: <https://pugixml.org/>
7. yaml cpp : YAML parser and emitter in C++ [Електронний ресурс]. – Режим доступу: <https://github.com/jbeder/yaml-cpp>
8. spdlog : Extremely fast logging library [Електронний ресурс]. – Режим доступу: <https://github.com/gabime/spdlog>
9. Conan : C++ package manager [Електронний ресурс]. – Режим доступу: <https://conan.io/>
10. Docker : Containerization platform for application packaging [Електронний ресурс]. – Режим доступу: <https://www.docker.com/>
11. Pandoc : Universal document converter [Електронний ресурс]. – Режим доступу: <https://pandoc.org/>
12. clang format : C/C++ code formatter based on LLVM Clang [Електронний ресурс]. – Режим доступу: <https://clang.llvm.org/docs/ClangFormat.html>

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

13. GitLab CI/CD Documentation [Електронний ресурс]. – Режим доступу: <https://docs.gitlab.com/ci/>
14. PlantUML – Tool to generate UML diagrams from plain text [Електронний ресурс]. – Режим доступу: <https://plantuml.com/>
15. PlantText – Online PlantUML editor [Електронний ресурс]. – Режим доступу: <https://www.planttext.com/>
16. C++ Reference (C++20 Standard Library and Language) [Електронний ресурс]. – Режим доступу: <https://en.cppreference.com/w/cpp/20>
17. Homebrew – macOS package manager [Електронний ресурс]. – Режим доступу: <https://brew.sh/>
18. AUR – Arch User Repository [Електронний ресурс]. – Режим доступу: <https://aur.archlinux.org/>
19. GoogleTest – Google C++ Testing Framework [Електронний ресурс]. – Режим доступу: <https://github.com/google/googletest>
20. Graphviz – Graph drawing software [Електронний ресурс]. – Режим доступу: <https://graphviz.org/>
21. GitHub Pages Documentation [Електронний ресурс]. – Режим доступу: <https://pages.github.com/>
22. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns : Elements of Reusable Object Oriented Software. – Boston : Addison Wesley, 1994.
23. Refactoring.Guru – Catalog of Design Patterns [Електронний ресурс]. – Режим доступу: <https://refactoring.guru/design-patterns>
24. Webster F. V. Traffic Signal Settings. – Road Research Technical Paper № 39. – London : HMSO, 1958. – 24 с.
25. Åström K. J., Hägglund T. PID Controllers: Theory, Design and Tuning. – 2 nd ed. – Research Triangle Park, NC : ISA, 1995. – 343 p.
26. Max Pressure Traffic Signal Control : A Decentralized Framework / Zhao X., Liu R., et al. – Transportation Research Part C, 2024. – Vol. 155. – Article 103864.

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

27. Traffic Flow Theory / Federal Highway Administration. – Washington, DC : FHWA
HRT 19 007, 2019. – 514 p.

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

Додаток А.

Файл конфігурації *tp_config.yaml*:

```
adaptive:
  check_every: 60
  enabled: true
  warmup: 240
  pid:
    init_set_pt: 0.85
    cap_move: 2
  tune:
    enabled: true

flow:
  min_mode_hold: 180
  queue:
    per_inter: 12

algo:
  min_cycle: 40
```

Додаток В.

Файл comparison.csv, де зберігаються результати виконання сценаріїв:

timestamp,scenario,mode,avg_travel_time,avg_waiting_time,avg_speed,throughput,num_stops,avg_queue_length,time_loss,halted_peak,density_peak,queue_peak

```
2025-05-13T16:26:49,Low Traffic
XS,base,327.316,36.2791,10.5447,713,645,36.2791,97.61,19,71.39,21.03
2025-05-13T16:27:41,Low Traffic
XS,custom,327.297,36.2623,10.5546,713,652,36.2623,97.62,16,70.03,28.53
2025-05-13T16:29:23,Low Traffic
S,base,435.948,39.3072,11.4427,2630,2350,39.3072,118.51,50,266.67,64.11
2025-05-13T16:31:18,Low Traffic
S,custom,434.913,39.0213,11.4652,2630,2325,39.0213,117.47,52,240.06,51.01
2025-05-13T16:32:16,Medium Traffic
XS,base,347.861,39.4284,10.1786,2138,1949,39.4284,106.18,44,198.91,53.79
2025-05-13T16:33:25,Medium Traffic
XS,custom,347.839,39.5477,10.1805,2138,1971,39.5477,106.16,43,197.21,36.21
2025-05-13T16:35:52,Medium Traffic
S,base,471.221,56.3885,10.6166,6574,5999,56.3885,146.53,147,400,133.62
2025-05-13T16:38:38,Medium Traffic
S,custom,470.915,56.0615,10.619,6574,6009,56.0615,146.4,162,400,97.52
2025-05-13T16:39:49,High Traffic
XS,base,367.493,45.1642,9.83466,3563,3307,45.1642,117.45,76,266.67,66.02
2025-05-13T16:41:09,High Traffic
XS,custom,365.948,44.1198,9.88294,3563,3294,44.1198,115.7,73,266.67,66.02
2025-05-13T16:44:41,High Traffic
S,base,626.891,161.086,8.25019,10519,9937,161.086,281.14,685,400,231.19
2025-05-13T16:48:25,High Traffic
S,custom,622.407,157.172,8.30269,10519,9948,157.172,277.9,636,400,283.61
2025-05-14T14:48:14,Low Traffic
S,custom,434.913,39.0213,11.4652,2630,2325,6.34725,117.47,52,240.06,51.01
2025-05-16T12:06:11,Low Traffic
XS,custom,327.297,36.2623,10.5546,713,652,5.39559,97.62,16,70.03,28.53
2025-05-16T12:08:36,Low Traffic
S,custom,434.913,39.0213,11.4652,2630,2325,6.34725,117.47,52,240.06,51.01
2025-05-16T12:10:19,Medium Traffic
```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

XS,custom,347.839,39.5477,10.1805,2138,1971,6.18165,106.16,43,197.21,36.21
2025-05-16T12:14:18,Medium Traffic
S,custom,470.915,56.0615,10.619,6574,6009,8.9078,146.4,162,400,97.52
2025-05-16T12:16:23,High Traffic
XS,custom,365.948,44.1198,9.88294,3563,3294,7.06699,115.7,73,266.67,66.02
2025-05-16T12:22:20,High Traffic
S,custom,622.407,157.172,8.30269,10519,9948,17.2851,277.9,636,400,283.61

Додаток С.

Код роботи:

```
#pragma once

#ifndef CMAKE_BINARY_DIR
namespace TrafficPulse
{
#define CMAKE_BINARY_DIR ""
} // namespace TrafficPulse
#endif // CMAKE_BINARY_DIR

#pragma once

namespace TrafficPulse
{
#ifndef SUMO_CONFIG_PATH
#define SUMO_CONFIG_PATH "config/sumo/scenarios/xs/low_traffic/osm.sumocfg"
}

#endif // SUMO_CONFIG_PATH

#pragma once

namespace TrafficPulse
{
#ifndef SUMO_NET_PATH
#define SUMO_NET_PATH "config/sumo/scenarios/xs/low_traffic/osm.net.xml"
}

#endif // SUMO_NET_PATH

#ifndef AUTOTUNER_H
#define AUTOTUNER_H

#pragma once

#include "controllers/PIDController.h"
#include "utils/Settings.h"

namespace TrafficPulse
{
/**
 * @class AutoTuner
 * @brief Periodically re-tunes PID gains & flow thresholds using live data.
 *
 * The tuner performs windowed aggregation of traffic flow and load
 * and, every retune_every seconds, rescales:
 */
```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

```

* - @ref Settings::FlowThresh::low / high - to widen or narrow MaxPressure
trigger band.
* - PID gains (Kp, Ki, Kd) - proportional to the observed load.
*
* Design patterns:
* - Observer - samples fed from simulation loop,
* - Strategy - encapsulates tuning heuristic independent of PID.
*/
class AutoTuner {
public:
    /**
     * @brief Ctor.
     *
     * @param cfg Adaptive-tuning parameters from YAML.
     * @param flow Reference to global flow threshold struct to mutate.
     * @param pid Target PID controller whose gains are updated.
     */
    explicit AutoTuner(const Settings::Adaptive::Tune& cfg, Settings::FlowThresh&
flow, PIDController& pid);

    /**
     * @brief Feed one observation sample.
     *
     * @param sim_time Current simulation time [s].
     * @param total_flow_norm Normalised flow [0,1].
     * @param queues Per-intersection queue lengths [veh].
     */
    void sample(double sim_time, double total_flow_norm, const
std::vector<double>& queues);

private:
    /**
     * @brief Attempt a retuning if policy deadlines are met.
     * @param sim_time Current simulation time [s].
     */
    void tryRetune(double sim_time);

    /**
     * @brief Composite load metric: `w * avg_queue + (1-w) * flow_norm`.
     */
    [[nodiscard]] double computeLoad(double flow_norm, const std::vector<double>&
queue) const;

    Settings::FlowThresh& m_flow; ///< Shared flow thresholds.
    PIDController& m_pid; ///< Controlled PID instance.
    Settings::Adaptive::Tune m_cfg; ///< Immutable tune config.

    double m_last_tune_time{0.0}; ///< Timestamp of previous successful tune.
    double m_next_tune_time{0.0}; ///< Next scheduled evaluation point.

    double m_accumulated_flow{0.0}; ///<  $\Sigma$  normalised flow in the current window.
    double m_accumulated_load{0.0}; ///<  $\Sigma$  load metric in the current window.
    size_t m_samples{0}; ///< Number of aggregated samples.
};

} // namespace TrafficPulse

#endif //AUTOTUNER_H

#pragma once

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				56
Змн.	Арк.	№ докум.	Підпис	Дата		

```

#ifndef METACONTROLLER_H
#define METACONTROLLER_H

#include <chrono>
#include <memory>
#include <vector>

#include "controllers/AutoTuner.h"
#include "controllers/TrafficModeController.h"
#include "core/EventBus.h"
#include "services/GreenWaveManager.h"
#include "strategies/BaselineWaveStrategy.h"
#include "strategies/HybridWaveStrategy.h"
#include "strategies/IGreenWaveStrategy.h"
#include "strategies/MaxPressureWaveStrategy.h"
#include "types/TrafficControlMode.h"
#include "utils/Settings.h"

namespace TrafficPulse
{
    /**
     * @class MetaController
     * @brief Runtime mode-selector that delegates to the active green-wave strategy.
     *
     * High-level logic:
     * 1. Mode controller decides between Baseline | Hybrid | MaxPressure
     * 2. AutoTuner (optional) continuously recalibrates thresholds + PID.
     * 3. `calculateOffsets()` forwards the call to the currently active strategy.
     *
     * Implements **Factory** creating strategies, **Proxy** delegating offsets,
     * and **Observer** listening to traffic stats.
     */
    class MetaController {
    public:
        /**
         * @brief Ctor.
         *
         * @param cfg Configuration object containing the thresholds.
         */
        explicit MetaController(Settings& cfg);

        /**
         * @brief Per-tick update
         *
         * @param sim_time      Current simulation time, seconds.
         * @param total_flow    Normalised total flow in [0..1].
         * @param queue_lengths Vector of per-intersection queues (veh).
         */
        void update(double sim_time, double total_flow, const std::vector<double>& queue_lengths);

        /**
         * @brief Delegated offset calculation.
         *
         * @param intersection_ids Ordered a corridor list.
         * @param get_distance     Callback to return distance between two IDs.
         * @param current_time     Current simulation time.
         * @param queue_lengths    Per-intersection queue lengths.
         * @param total_flow       Normalised total flow in [0..1].
         */
        std::unordered_map<std::string, double>

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        calculateOffsets(const std::vector<std::string>& intersection_ids,
                        const std::function<double(const std::string&, const
std::string&)>& get_distance,
                        double current_time,
                        const std::vector<double>& queue_lengths,
                        double total_flow) const;

/**
 * @brief Get the current mode.
 *
 * @return Current traffic control mode.
 */
[[nodiscard]] TrafficControlMode getCurrentMode() const;

/**
 * @brief Get the active strategy.
 *
 * @return Reference to the active strategy.
 */
[[nodiscard]] const IGreenWaveStrategy& getActiveStrategy() const;

/** @copydoc MetaController::getActiveStrategy() */
[[nodiscard]] IGreenWaveStrategy& getActiveStrategy();

/**
 * @brief Force a particular mode (diagnostics / manual override).
 *
 * @param mode The mode to force.
 */
void forceMode(TrafficControlMode mode);

private:
/**
 * @brief Helper returning the mutable active strategy.
 *
 * @return Reference to the active strategy.
 */
IGreenWaveStrategy& activeStrategy();

/** @copydoc MetaController::activeStrategy() */
[[nodiscard]] const IGreenWaveStrategy& activeStrategy() const;

Settings::FlowThresh& m_flow; ///< Reference from Settings.
TrafficModeController m_mode_ctrl; ///< Finite-state mode machine.
std::unique_ptr<BaselineWaveStrategy> m_baseline; ///< Baseline strategy.
std::unique_ptr<HybridWaveStrategy> m_hybrid; ///< Hybrid strategy.
std::unique_ptr<MaxPressureWaveStrategy> m_max_pressure; ///< MaxPressure
strategy.
std::optional<AutoTuner> m_tuner; ///< Lazy-constructed if enabled.
};

} // namespace TrafficPulse

#endif // METACONTROLLER_H

#pragma once

#ifndef ADAPTIVECONTROLLER_H
#define ADAPTIVECONTROLLER_H

#include "utils/Settings.h"

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

```

namespace TrafficPulse
{
/**
 * @class PIDController
 * @brief Classical PID with anti-windup and move-cap guarding.
 *
 * Tunable via YAML: gains (Kp, Ki, Kd) and +-Δcap to prevent brutal phase jumps.
 * The controller is used to adjust the corridor **offset** each cycle so that
 * the measured flow approaches the target set-point while considering the average
 * queue.
 */
class PIDController {
public:
/**
 * @brief Ctor.
 *
 * @param cfg Adaptive PID parameters from YAML.
 */
explicit PIDController(const Settings::Adaptive& cfg);

/**
 * @brief Update PID and return a new adjustment Δoffset.
 *
 * @param measured_flow Actual normalised flow [0,1].
 * @param dt Time since last call [s].
 * @param avg_queue Mean queue length [veh].
 *
 * @return Δoffset in seconds.
 */
double update(double measured_flow, double dt, double avg_queue);

/**
 * @brief Set the target set-point.
 *
 * @param set_point Target set-point [0,1].
 */
void setTarget(double set_point);

/**
 * @brief Reset the PID controller.
 */
void reset();

/**
 * @brief Set the PID gains
 *
 * @param Kp Proportional gain.
 * @param Ki Integral gain.
 * @param Kd Derivative gain.
 */
void setGains(double Kp, double Ki, double Kd);

/**
 * @brief Reset the integral term.
 */
void resetIntegrator();

private:
const Settings::Adaptive& m_cfg; ///< Global adaptive config (read-only).

double m_Kp{0}; ///< Proportional gain.

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				59
Змн.	Арк.	№ докум.	Підпис	Дата		

```

double m_Ki{0}; ///< Integral gain.
double m_Kd{0}; ///< Derivative gain.

double m_integral{0.0}; ///< Integral error dt with anti-windup.
double m_set_point{0.0}; ///< Desired normalised flow.
double m_prev_error{0.0}; ///< Error at a previous step.
double m_last_offset_delta{0.0}; ///< Last  $\Delta$ offset produced.
};
} // namespace TrafficPulse

#endif //ADAPTIVECONTROLLER_H

#pragma once

#ifndef TRAFFICMODECONTROLLER_H
#define TRAFFICMODECONTROLLER_H

#include "types/TrafficControlMode.h"
#include "utils/Settings.h"

namespace TrafficPulse
{
/**
 * @brief Persistently chooses traffic-control mode based on flow / queues.
 *
 * The controller implements hysteresis with a configurable *dwell* time so
 * that modes do not flap under noisy conditions.
 */
class TrafficModeController {
public:
/**
 * @brief Configure the thresholds for automatic mode selection.
 * @param cfg Configuration object containing the thresholds.
 */
explicit TrafficModeController(const Settings::FlowThresh& cfg);

/**
 * @param total_flow Normalised total flow [0,1].
 * @param queue_lengths Vector of per-intersection queues [veh].
 * @param sim_time Current simulation time [s].
 * @return Possibly a new mode after evaluation.
 *
 * *Algorithm*:
 * 1. Early-exit if < dwell time since last switch.
 * 2. If any queue or total_flow crosses a high-threshold switch to
MaxPressure.
 * 3. Else if load > 1.0, switch to Hybrid, otherwise Baseline.
 */
TrafficControlMode evaluate(double total_flow, const std::vector<double>&
queue_lengths, double sim_time);

/**
 * @brief Get the currently selected mode.
 */
[[nodiscard]] TrafficControlMode getCurrentMode() const;

/**
 * @brief Instantly switch mode.
 *
 * @param mode Mode to switch to.
 */

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    void forceMode(TrafficControlMode mode);

private:
    TrafficControlMode m_current_mode = TrafficControlMode::Baseline; ///< Current
mode.
    Settings::FlowThresh m_cfg{}; ///< Flow thresholds.
    double m_last_switch_time = 0.0; ///< Last mode transition [s].
};
} // namespace TrafficPulse

#endif // TRAFFICMODECONTROLLER_H

#pragma once

#ifndef EVENTBUS_H
#define EVENTBUS_H

#include "events/IEventDispatcher.h"
#include <functional>
#include <memory>
#include <typeindex>
#include <unordered_map>
#include <vector>

namespace TrafficPulse
{
/**
 * @class EventBus
 * @brief In-process publish/subscribe hub implemented as a Singleton.
 *
 * Internally keeps a map: `type_index -> vector<callback>`.
 * All callbacks live until program exit—acceptable in this global context.
 */
class EventBus : public IEventDispatcher {
public:
    /// Returns the global instance (thread-unsafe lazy init).
    static EventBus& getInstance() {
        static EventBus instance;
        return instance;
    }

    EventBus() = default;
    EventBus(const EventBus&) = delete;
    EventBus& operator=(const EventBus&) = delete;

    /**
     * @tparam TPayload Payload type of the event.
     * @param handler Function invoked when such event is dispatched.
     */
    template <typename TPayload> void subscribe(std::function<void(const
TPayload&)> handler) {
        const std::type_index type_id = typeid(TPayload);
        m_subscribers[type_id].push_back([handler](const
std::shared_ptr<BaseEvent>& base_event) {
            auto typed_event =
std::static_pointer_cast<TypedEvent<TPayload>>(base_event);
            handler(typed_event->getPayload());
        });
    }

    /// Synchronously fan-out the event to all subscribers.

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        void dispatch(const std::shared_ptr<BaseEvent>& event) override {
            if(const auto it = m_subscribers.find(event->getType()); it !=
m_subscribers.end()) {
                for(const auto& subscriber : it->second) {
                    subscriber(event);
                }
            }
        }

private:
    std::unordered_map<std::type_index, std::vector<std::function<void(const
std::shared_ptr<BaseEvent>&)>>>
        m_subscribers; ///< Subscribers map with a key as a payload type and value
as a list of erased callbacks
    };
} // namespace TrafficPulse

#endif // EVENTBUS_H

#pragma once

#ifndef PULSEDATAMANAGER_H
#define PULSEDATAMANAGER_H

#include <memory>
#include <string>
#include <unordered_map>
#include <unordered_set>
#include <vector>

#include "core/EventBus.h"
#include "core/PulseException.h"
#include "core/SumoIntegration.h"
#include "entities/PulseIntersection.h"
#include "entities/PulseTrafficLight.h"
#include "entities/PulseVehicle.h"

namespace TrafficPulse
{
/**
 * @class PulseDataManager
 * @brief Singleton manager that stores all traffic simulation entities
(intersections, traffic lights, vehicles).
 *
 * It can sync data with a live SUMO simulation via SumoIntegration, ensuring that
 * local objects remain consistent with the real-time simulation.
 */
class PulseDataManager {
public:
    /**
     * @brief Retrieves the singleton instance.
     *
     * @return Reference to the single PulseDataManager.
     */
    static PulseDataManager& getInstance();

    /**
     * @brief Inject the global event bus so entities can publish notifications.
     *
     * @param bus Reference to the EventBus instance.
     */

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

```

void setEventBus(EventBus& bus);

/**
 * @brief Adds a new intersection to the data manager.
 *
 * @throws std::invalid_argument if the intersection is null
 * @throws PulseException if an intersection with the same ID already exists
 */
void addIntersection(std::unique_ptr<PulseIntersection> intersection);

/**
 * @brief Retrieves an intersection by ID.
 *
 * @param intersection_id The ID of the intersection to retrieve.
 *
 * @return Pointer to the intersection, or nullptr if not found.
 */
PulseIntersection* getIntersection(const std::string& intersection_id) const;

/**
 * @brief Adds a new traffic light to the data manager.
 *
 * @throws std::invalid_argument if traffic_light is null
 * @throws PulseException if a traffic light with the same ID already exists
 */
void addTrafficLight(std::unique_ptr<PulseTrafficLight> traffic_light);

/**
 * @brief Retrieves a traffic light by ID.
 *
 * @param traffic_light_id The ID of the traffic light.
 *
 * @return Pointer to the traffic light, or nullptr if not found.
 */
PulseTrafficLight* getTrafficLight(const std::string& traffic_light_id) const;

/**
 * @brief Adds a new vehicle to the data manager.
 *
 * @throws std::invalid_argument if the vehicle is null
 * @throws PulseException if a vehicle with the same ID already exists
 */
void addVehicle(std::unique_ptr<PulseVehicle> vehicle);

/**
 * @brief Retrieves a vehicle by ID.
 *
 * @param vehicle_id The ID of the vehicle.
 *
 * @return Pointer to the vehicle, or nullptr if not found.
 */
PulseVehicle* getVehicle(const std::string& vehicle_id) const;

/**
 * @brief Retrieves all intersections in the system.
 *
 * @return A list of pointers to all intersections.
 */
std::vector<PulseIntersection*> getAllIntersections() const;

/**
 * @brief Retrieves all traffic lights in the system.

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				63
Змн.	Арк.	№ докум.	Підпис	Дата		

```

*
* @return A list of pointers to all traffic lights.
*/
std::vector<PulseTrafficLight*> getAllTrafficLights() const;

/**
* @brief Retrieves all vehicles in the system.
*
* @return A list of pointers to all vehicles.
*/
std::vector<PulseVehicle*> getAllVehicles() const;

/**
* @brief Mark which traffic-light IDs correspond to physical TLS devices.
*
* @param tls A set of traffic light IDs.
*/
void setRealTrafficLights(std::unordered_set<std::string> tls);

/**
* @brief Checks if the given ID belongs to a real simulated TLS device.
*
* @param id The ID of the traffic light to check.
*
* @return true if the given ID belongs to a real simulated TLS device.
*/
bool isRealTrafficLight(const std::string& id) const;

/**
* @brief Retrieves the currently registered real-TLS set.
*
* @return Mutable reference to the currently registered real-TLS set.
*/
std::unordered_set<std::string>& getRealTrafficLights();

/**
* @brief Clears all stored data (used when resetting or re-syncing).
*/
void clearAll();

/**
* @brief Syncs data from SUMO the first time (or after clearing).
*       This loads all traffic lights as intersections, plus vehicles.
*
* @param sumo Reference to the SumoIntegration instance.
*/
void syncFromSumo(const SumoIntegration& sumo);

/**
* @brief Updates local data from the current SUMO simulation step.
*       Typically called each time we step the simulation.
*
* @param sumo Reference to the SumoIntegration instance.
*/
void updateFromSumo(const SumoIntegration& sumo);

/**
* @brief Deleted - singleton must not be copied.
*/
PulseDataManager(const PulseDataManager&) = delete;

/**

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				64
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        * @brief Deleted - singleton must not be reassigned.
        */
PulseDataManager& operator=(const PulseDataManager&) = delete;

private:
    /**
     * @brief Hidden ctor - use getInstance() instead.
     */
    PulseDataManager() = default;

    std::unordered_map<std::string, std::unique_ptr<PulseIntersection>>
        m_intersections; ///< All intersections keyed by ID
    std::unordered_map<std::string, std::unique_ptr<PulseTrafficLight>>
        m_traffic_lights; ///< All traffic lights keyed by ID.
    std::unordered_map<std::string, std::unique_ptr<PulseVehicle>> m_vehicles;
    ///< All vehicles keyed by ID.
    std::unordered_set<std::string> m_real_tls; ///< IDs of TLS that exist in the
    SUMO network.
    EventBus* m_event_bus = nullptr; ///< Pointer to the event bus for dispatching
    events.
};
} // namespace TrafficPulse
#endif // PULSEDATAMANAGER_H

#pragma once

#ifndef PULSEEXCEPTION_H
#define PULSEEXCEPTION_H

#include <sstream>
#include <stdexcept>
#include <string>
#include <utility>

#include "types/PulseErrorCode.h"
#include "utils/Log.h"

namespace TrafficPulse
{
    /**
     * @class PulseException
     * @brief Custom exception class for the Pulse library.
     *
     * Inherits from std::runtime_error and provides additional context
     * such as error codes and optional nested exceptions.
     */
    class PulseException : public std::runtime_error {
    public:
        /**
         * @brief Constructs a PulseException with message, error code, and optional
         * cause.
         *
         * @param message The error description.
         * @param code The specific error code.
         * @param cause Optional pointer to the nested exception.
         */
        explicit PulseException(const std::string& message,
                                const PulseErrorCode code = PulseErrorCode::Unknown,
                                std::exception_ptr cause = nullptr)
            : std::runtime_error(message)
            , error_code(code)
            , cause(std::move(cause)) { }
    };
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

```

/**
 * @brief Gets the associated error code.
 *
 * @return The PulseErrorCode.
 */
PulseErrorCode code() const noexcept {
    return error_code;
}

/**
 * @brief Gets the underlying cause if available.
 *
 * @return std::exception_ptr to the cause.
 */
std::exception_ptr getCause() const noexcept {
    return cause;
}

/**
 * @brief Provides a full description of the exception, including the code.
 *
 * @return The detailed C-style string.
 */
const char* what() const noexcept override {
    if(cached_what.empty()) {
        std::ostringstream oss;
        oss << std::runtime_error::what();
        spdlog::error("{} [Code: {}]", oss.str(), static_cast<int>(error_code));
        cached_what = oss.str();
    }
    return cached_what.c_str();
}

private:
    PulseErrorCode error_code; ///< Library-specific error enumeration.
    std::exception_ptr cause; ///< Optional nested exception for chaining.
    mutable std::string cached_what; ///< Lazily built `what()` string.
};

} // namespace TrafficPulse

#endif // PULSEEXCEPTION_H

#pragma once

#ifndef PULSELOADER_H
#define PULSELOADER_H

#include "core/EventBus.h"
#include "core/PulseDataManager.h"

#include <string>

namespace TrafficPulse
{
/**
 * @class PulseLoader
 * @brief Loads and parses a SUMO .net.xml file, populating PulseDataManager with
 * intersections,
 * traffic lights, etc. It notifies its subject (e.g. TrafficSystem) about
 * loading progress.
 */

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				66
Змн.	Арк.	№ докум.	Підпис	Дата		

```

class PulseLoader {
public:
    /**
     * @brief Constructs a PulseLoader with a reference to PulseDataManager and a
     path to the net file.
     *
     * @param data_manager Reference to the data manager.
     * @param net_file Path to the SUMO .net.xml.
     * @param bus Reference to the event bus.
     */
    PulseLoader(PulseDataManager& data_manager, std::string net_file, EventBus&
bus);

    /**
     * @brief Loads network data from the specified .net.xml file into the
     PulseDataManager.
     *
     *     Notifies LOADING_START, LOADING_PROGRESS, LOADING_COMPLETE events
     during the process.
     *
     * @throws PulseException if the file is missing or invalid (e.g. parsing
     error).
     */
    void loadNetworkData() const;

private:
    /**
     * @brief Generates a simple incremental ID for road connections.
     *
     * @return An integer ID unique for each call in the current session.
     */
    static int generateRoadID();

    PulseDataManager& m_data_manager; ///< Reference to the data manager.
    std::string m_net_file_path; ///< Path to the SUMO .net.xml file.
    EventBus& m_event_bus; ///< Reference to the event bus.
};
} // namespace TrafficPulse

#endif // PULSELOADER_H

#pragma once

#ifndef PULSESTATSCOLLECTOR_H
#define PULSESTATSCOLLECTOR_H

#include <string>
#include <unordered_map>

#include "core/EventBus.h"
#include "core/PulseDataManager.h"
#include "types/PulseIntersectionStats.h"
#include "types/PulseLaneStats.h"
#include "types/PulseTrafficLightStats.h"
#include "types/PulseVehicleType.h"
#include "types/StatsSnapshot.h"

namespace TrafficPulse
{
    /**
     * @class PulseStatsCollector
     * @brief Gathers essential statistics about vehicles, intersections, traffic
     lights each simulation step.

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

*/
class PulseStatsCollector {
public:
    /**
     * @brief Constructs a PulseStatsCollector.
     *
     * @param data_manager Reference to the data manager.
     * @param sumo_integration Reference to the SumoIntegration object.
     * @param bus Reference to the event bus.
     */
    explicit PulseStatsCollector(PulseDataManager& data_manager, SumoIntegration&
sumo_integration, EventBus& bus);

    /**
     * @brief Called each simulation step to update stats.
     *
     * @param current_sim_time The current simulation time.
     */
    void collectStepData(double current_sim_time);

    /**
     * @brief Retrieves the average speed among all vehicles in the last step.
     *
     * @return Average vehicle speed (m/s).
     */
    double getAverageSpeed() const;

    /**
     * @brief Retrieves the maximum speed among all vehicles in the last step.
     *
     * @return Maximum vehicle speed (m/s).
     */
    double getMaxSpeed() const;

    /**
     * @brief Retrieves the vehicle type counts from the last step.
     *
     * @return Map of vehicle type -> count.
     */
    std::unordered_map<PulseVehicleType, int> getVehicleTypeCounts() const;

    /**
     * @brief Returns the average wait time at a particular intersection.
     *
     * @param intersection_id The ID of the intersection.
     *
     * @return The average wait time in seconds.
     */
    double getIntersectionAvgWaitTime(const std::string& intersection_id) const;

    /**
     * @brief Returns the total number of vehicles passed through this
intersection so far.
     *
     * @param intersection_id The ID of the intersection.
     *
     * @return The pass count.
     */
    int getIntersectionVehiclesPassed(const std::string& intersection_id) const;

    /**
     * @brief Retrieves how many total seconds the given light has been green.

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

*
* @param traffic_light_id The ID of the traffic light.
*/
double getTimeSpentGreen(const std::string& traffic_light_id) const;

/**
* @brief Retrieves how many total seconds the given light has been red.
*
* @param traffic_light_id The ID of the traffic light.
*/
double getTimeSpentRed(const std::string& traffic_light_id) const;

/**
* @brief Retrieves how many vehicles have been stopped at red for this
traffic light.
*
* @param traffic_light_id The ID of the traffic light.
*/
int getVehiclesStoppedAtRed(const std::string& traffic_light_id) const;

/**
* @brief Retrieves how many times this traffic light has changed cycle from
red to green or green to red.
*
* @param traffic_light_id The ID of the traffic light.
*
* @return The number of cycles.
*/
int getLightCycleCount(const std::string& traffic_light_id) const;

/**
* @brief Retrieves how many vehicles are currently waiting at a specific
lane.
*
* @param lane_id The ID of the lane.
*
* @return The number of vehicles waiting at the lane.
*/
int getLaneWaitingCount(const std::string& lane_id) const;

/**
* @brief Retrieves the stats for a specific intersection.
*
* @param intersection_id The ID of the intersection.
*
* @return The stats for the specified intersection.
*/
PulseIntersectionStats getIntersectionStats(const std::string&
intersection_id) const;

/**
* @brief Builds and returns a snapshot of the current stats (vehicle,
intersection, lights).
*
* @return A StatsSnapshot containing all relevant data at this moment.
*/
StatsSnapshot getSnapshot() const;

private:
/**
* @brief Updates the vehicle-related stats (avg speed, max speed, type
counts).

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

```

*
* @param current_sim_time The current simulation time.
*/
void updateVehicleStats(double current_sim_time);

/**
* @brief Updates intersection-related stats (wait time, pass counts).
*
* @param current_sim_time The current simulation time.
*/
void updateIntersectionStats(double current_sim_time);

/**
* @brief Updates traffic-light-related stats (time spent green/red, vehicles
stopped, cycle count).
*
* @param current_sim_time The current simulation time.
*/
void updateTrafficLightStats(double current_sim_time);

/**
* @brief Gathers lane-based metrics from SumoIntegration and updates rolling
averages each step.
*
* @param current_sim_time The current simulation time in seconds.
*/
void updateLaneStats(double current_sim_time);

PulseDataManager& m_data_manager; ///< Reference to the data manager
SumoIntegration& m_sumo; ///< Reference to the SumoIntegration object
EventBus& m_event_bus; ///< Reference to the event bus

std::unordered_map<PulseVehicleType, int> m_vehicle_type_counts; ///< Vehicle
type counts
std::unordered_map<std::string, PulseIntersectionStats> m_intersection_stats;
///< Intersection stats
std::unordered_map<std::string, PulseTrafficLightStats> m_light_stats; ///<
Traffic light stats
std::unordered_map<std::string, PulseLaneStats> m_lane_map; ///< Lane stats
std::unordered_map<std::string, double> m_vehicle_intersection_memory; ///<
Memory of vehicle positions

double m_avg_speed = 0.0; ///< Average speed of all vehicles
double m_max_speed = 0.0; ///< Maximum speed of all vehicles
double m_previous_time = 0.0; ///< For measuring durations since the last step
};
} // namespace TrafficPulse

#endif // PULSESTATSCOLLECTOR_H

#pragma once

#ifndef PULSETRAFFICALGO_H
#define PULSETRAFFICALGO_H

#include "controllers/MetaController.h"
#include "core/PulseDataManager.h"
#include "services/GreenWaveManager.h"
#include "types/GreenWaveCoridor.h"
#include "types/TrafficControlMode.h"
#include "utils/Settings.h"

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

#include <string>
#include <unordered_map>
#include <vector>

namespace TrafficPulse
{
class SumoIntegration;
class EventBus;
/**
 * @class PulseTrafficAlgo
 * @brief Implements a green-wave traffic control approach using BFS-based
corridor detection,
 *      with an option to update simulation traffic lights if SUMO is running.
 */
class PulseTrafficAlgo {
public:
    /**
     * @brief Constructs the traffic algorithm with references to data manager and
sumo integration.
     *
     * @param data_manager Reference to PulseDataManager for local entity data.
     * @param sumo_integration Reference to SumoIntegration for real-time control
of lights.
     * @param bus Reference to EventBus for dispatching events.
     * @param cfg Configuration object for setting up the algorithm.
     */
    PulseTrafficAlgo(PulseDataManager& data_manager, SumoIntegration&
sumo_integration, EventBus& bus, Settings& cfg);

    /**
     * @brief Initializes the green wave schedules by finding corridors and
computing offsets.
     *      Typically called once after the system is loaded with intersections.
     *
     * @param current_sim_time Initial simulation time.
     */
    void initializeGreenWave(double current_sim_time);

    /**
     * @brief Runs the green wave logic at a given simulation time.
     *
     * @param current_sim_time Current simulation time (e.g., from SUMO or a
global clock).
     */
    void runOnce(double current_sim_time);

    /**
     * @brief Enables or disables the control of simulation lights.
     *      When disabled, only local traffic light objects are updated.
     *
     * @param enable True to control simulation lights, false to update only local
objects.
     */
    void setControlSimulationLights(bool enable);

    /**
     * @brief Sets the traffic mode for the algorithm.
     *
     * @param mode Mode to set.
     */
    void setTrafficMode(TrafficControlMode mode);

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

/**
 * @brief Gets the current traffic control mode.
 *
 * @return Current traffic control mode.
 */
TrafficControlMode getMode() const;

/**
 * @brief Gets the cycle length for the green wave algorithm.
 *
 * @return Cycle length in seconds.
 */
double getCycleLength() const;

private:
/**
 * @brief Decide if an intersection is major enough
 *
 * @param inter Pointer to the intersection to check.
 */
bool isMajorIntersection(const PulseIntersection* inter) const;

/**
 * @brief Flow normalised to [0,1] using a high threshold.
 *
 * @param raw Raw flow value to normalise.
 *
 * @return Normalised flow value in [0,1].
 */
double normaliseFlow(double raw) const;

/**
 * @brief Finds major corridors in the intersection graph using BFS or a
custom approach.
 *
 * Here we just pick one or two main paths for demonstration.
 */
std::vector<GreenWaveCorridor> findMainCorridors() const;

/**
 * @brief Calculates offsets for each intersection in each corridor based on
road distances
 *
 * and the corridor's target speed. Offsets stored in
m_intersectionGreenOffset.
 */
void calculateGreenWaveSchedules();

/**
 * @brief Checks for emergency vehicles on inbound lanes and forces the
corresponding intersection light to GREEN.
 *
 * @param current_sim_time The current simulation time.
 */
void applyEmergencyOverride(double current_sim_time);

/**
 * @brief Return average free-flow speed for a corridor, weighted by lane
lengths, falls back to default_speed.
 *
 * @param intersection_ids List of intersection IDs to consider.
 *
 * @return Estimated target speed in m/s.
 */

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				72
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    double estimateTargetSpeed(const std::vector<std::string>& intersection_ids)
const;

/**
 * @brief Applies an offset to the traffic light's schedule.
 *
 * @param tl_id The ID of the traffic light.
 * @param offset_sec The offset in seconds to apply.
 */
static void applyOffset(const std::string& tl_id, double offset_sec);

PulseDataManager& m_data_manager; ///< Reference to the data manager.
SumoIntegration& m_sumo_integration; ///< Reference to the SUMO integration
instance.
EventBus& m_event_bus; ///< Reference to the event bus for dispatching events.
const Settings& m_cfg; ///< Immutable global settings reference.

std::unique_ptr<GreenWaveManager> m_green_wave_manager =
    nullptr; ///< Pointer to the green wave manager for managing green wave
schedules.
std::shared_ptr<MetaController> m_meta; ///< Pointer to the meta-controller
for managing traffic control strategies.

double m_last_departed{0.0}; ///< Vehicles departed at previous sample.
double m_last_flow_time{0.0}; ///< Timestamp of last flow calculation.
double m_current_flow{0.0}; ///< Vehicles departed since last sample.
double m_current_sim_time{0.0}; ///< Current simulation timestamp
double m_next_adapt_check{0.0}; ///< Next time green-wave strategy should
re-evaluate itself.
bool m_control_simulation_lights{false}; ///< Flag to determine if simulation
lights should be updated.

std::unordered_map<std::string, double> m_offsets; ///< Offset for each wave
m/sec.
/**
 * @brief TLS state backup when an emergency override is active.
 *
 * Map: TLS-ID → {previous_state, restore_at_time}.
 */
std::unordered_map<std::string, std::pair<std::string, double>>
m_tls_restore_map;
std::vector<GreenWaveCorridor> m_corridors; ///< Latest BFS-discovered
corridors with their target speeds.
TrafficControlMode m_prev_mode = TrafficControlMode::Baseline; ///< Previous
traffic mode for change detection.
};
} // namespace TrafficPulse

#endif // PULSETRAFFICALGO_H

#pragma once

#ifndef SUMOINTEGRATION_H
#define SUMOINTEGRATION_H

#include "core/PulseException.h"
#include "types/SumoOutput.h"

#include <string>
#include <vector>

namespace TrafficPulse

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

```

{
/**
 * @class SumoIntegration
 * @brief Demonstrates using libsumo for starting, stepping, and controlling SUMO
from C++.
 *
 * The constructor accepts an optional flag to bypass filesystem configuration
checks.
 * This is useful for mocking purposes.
 */
class SumoIntegration {
public:
/**
 * @brief Constructs a SumoIntegration object.
 * @param sumo_config The path to a SUMO config file.
 * @param net_file The path to a SUMO network file.
 * @param extra Additional SUMO output files to be generated.
 * @param bypass_config_check If true, bypasses filesystem checks for the
config and net files.
 * @param run_with_gui If true, runs the simulation with a GUI.
 */
explicit SumoIntegration(const std::string& sumo_config,
                        const std::string& net_file,
                        const std::vector<SumoOutput>& extra = {},
                        bool bypass_config_check = false,
                        bool run_with_gui = false);

virtual ~SumoIntegration() = default;

/**
 * @brief Starts the SUMO simulation using libsumo.
 */
virtual void startSimulation();

/**
 * @brief Steps the simulation forward by one timestep.
 */
virtual void stepSimulation() const;

/**
 * @brief Stops the SUMO simulation.
 */
virtual void stopSimulation();

/**
 * @brief Checks if the simulation is running.
 * @return True if the simulation is running, false otherwise.
 */
[[nodiscard]] virtual bool isRunning() const;

/**
 * @brief Retrieves all vehicle IDs.
 * @return A vector of vehicle IDs.
 */
[[nodiscard]] virtual std::vector<std::string> getAllVehicles() const;

/**
 * @brief Retrieves the (x,y) position of a given vehicle by ID.
 */
[[nodiscard]] virtual std::pair<double, double> getVehiclePosition(const
std::string& vehicle_id) const;

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

/**
 * @brief Retrieves a list of traffic light IDs.
 * @return A vector of traffic light IDs.
 */
[[nodiscard]] virtual std::vector<std::string> getAllTrafficLights() const;

/**
 * @brief Retrieves a traffic light's state as a string.
 */
[[nodiscard]] virtual std::string getTrafficLightState(const std::string&
tl_id) const;

/**
 * @brief Sets the state of the specified traffic light.
 */
virtual void setTrafficLightState(const std::string& tl_id, const std::string&
state) const;

/**
 * @brief Retrieves the "type" string of a vehicle from SUMO.
 * @param vehicle_id The ID of the vehicle in SUMO.
 * @return A string describing the vehicle type.
 * @throws PulseException if SUMO is not running or if there's an underlying
SUMO error.
 */
[[nodiscard]] virtual std::string getVehicleType(const std::string&
vehicle_id) const;

/**
 * @brief Retrieves all lane IDs in the current SUMO simulation derived from
edges.
 */
[[nodiscard]] virtual std::vector<std::string> getAllLanes() const;

/**
 * @brief Retrieves all vehicle IDs currently on the given lane.
 * @param lane_id The ID of the lane.
 */
[[nodiscard]] virtual std::vector<std::string> getLaneVehicles(const
std::string& lane_id) const;

/**
 * @brief Returns the last step's mean speed in m/s for the given lane, as
tracked by SUMO.
 * @param lane_id The ID of the lane.
 */
[[nodiscard]] virtual double getLaneMeanSpeed(const std::string& lane_id)
const;

/**
 * @brief Returns the last step's occupancy for the given lane [0,1].
 * @param lane_id The ID of the lane.
 */
[[nodiscard]] virtual double getLaneOccupancy(const std::string& lane_id)
const;

/**
 * @brief Retrieve the static speed-limit of a lane in m/s.
 *
 * @throws PulseException if SUMO not running.
 */
[[nodiscard]] virtual double getLaneSpeedLimit(const std::string& lane_id)

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const;

/**
 * @brief Departure time since simulation start.
 *
 * @param vehicle_id The ID of the vehicle.
 */
[[nodiscard]] virtual double getVehicleDepartTime(const std::string&
vehicle_id) const;

/**
 * @brief Number of vehicles currently queued on @p lane_id.
 *
 * @param lane_id The ID of the lane.
 */
[[nodiscard]] virtual double getLaneQueueLength(const std::string& lane_id)
const;

/**
 * @brief Returns the number of vehicles that have departed from the
simulation.
 *
 * @return The number of departed vehicles.
 */
[[nodiscard]] virtual double getNumDeparted() const;

/**
 * @brief Returns the geometric length of the given lane in meters.
 */
[[nodiscard]] double getLaneLength(const std::string& lane_id) const;

/**
 * @brief Retrieves the path to the SUMO net file.
 */
[[nodiscard]] const std::string& getNetPath() const;

/**
 * @brief Retrieves the path to the SUMO configuration file.
 */
[[nodiscard]] const std::string& getConfigPath() const;

/**
 * @brief Retrieves the simulation time step (deltaT) in seconds.
 */
[[nodiscard]] static double getDeltaT();

/**
 * @brief Retrieves the current simulation time in seconds.
 */
[[nodiscard]] static double getTime();

/**
 * @brief Call this once you have a .net.xml path to parse lanes with pugixml
and store them internally.
 * @throws PulseException if the file is missing or fails to parse, or if
<net> the root is missing.
 */
virtual void loadLaneData();

protected:
std::string m_sumo_config; ///< Path to the SUMO configuration file.
std::string m_net_config; ///< Path to the SUMO network file.

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        std::vector<std::string> m_lane_ids; ///< All lane IDs found in the net file
        bool m_started{false}; ///< Flag indicating if the simulation has started.
        bool m_run_with_gui{false}; ///< Flag indicating if the simulation is running
        with a GUI.
        std::vector<SumoOutput> m_extra; ///< Additional SUMO output files to be
        generated.
    };
} // namespace TrafficPulse

#endif // SUMOINTEGRATION_H

#pragma once

#ifndef TRAFFICSYSTEM_H
#define TRAFFICSYSTEM_H

#include "constants/SumoConfigPath.h"
#include "constants/SumoNetPath.h"
#include "controllers/TrafficModeController.h"
#include "core/EventBus.h"
#include "types/RunState.h"
#include "types/SumoOutput.h"

#include <atomic>
#include <memory>

namespace TrafficPulse
{
class PulseTrafficAlgo;

/**
 * @brief Main system class for traffic pulse.
 *
 * This class creates and manages its own modules and runs the simulation loop.
 * It acts as a Subject so that observers can be notified
 * about system events.
 */
class TrafficSystem {
public:
    /**
     * @brief Constructs a TrafficSystem.
     * @param update_frequency The frequency in seconds at which the system
     updates observers.
     * @param sumo_config_path The path to the SUMO configuration file.
     * @param sumo_net_path The path to the SUMO network file.
     * @param bus The event bus to use for communication.
     * @param algo_enabled Flag to enable or disable the traffic algorithm.
     * @param run_with_gui Flag to run the simulation with a GUI.
     * @param extra Additional SUMO output files to be generated.
     */
    explicit TrafficSystem(double update_frequency = 5.0,
                           const std::string& sumo_config_path = SUMO_CONFIG_PATH,
                           const std::string& sumo_net_path = SUMO_NET_PATH,
                           EventBus& bus = EventBus::getInstance(),
                           bool algo_enabled = true,
                           bool run_with_gui = false,
                           const std::vector<SumoOutput>& extra = {});

    ~TrafficSystem();

    /**
     * @brief Runs the simulation loop continuously until a stop is requested.

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    */
    void run();

    /**
     * @brief Requests the simulation to stop.
     */
    void requestStop();

    /**
     * @brief Pauses the simulation.
     */
    void pause();

    /**
     * @brief Resumes the simulation.
     */
    void resume();

    /**
     * @brief Checks if the simulation is paused.
     */
    [[nodiscard]] bool isPaused() const;

    /**
     * @brief Checks if the simulation is running.
     */
    [[nodiscard]] bool isRunning() const;

    /**
     * @brief Provides access to the data manager.
     */
    [[nodiscard]] class PulseDataManager& getDataManager() const;

    /**
     * @brief Provides access to the event bus.
     */
    [[nodiscard]] EventBus& getEventBus() const;

    /**
     * @brief Provides access to algo module.
     */
    [[nodiscard]] PulseTrafficAlgo& getAlgo() const;

    /**
     * @brief Loads static data from the SUMO network file.
     */
    void loadStaticData() const;

private:
    Settings m_cfg; ///< Configuration settings for the system.
    PulseDataManager& m_data_manager; ///< Reference to the data manager.
    EventBus& m_event_bus; ///< Event bus for managing events.

    std::unique_ptr<class SumoIntegration> m_sumo; ///< Pointer to the SUMO
integration module.
    std::unique_ptr<PulseTrafficAlgo> m_algo; ///< Pointer to the traffic
algorithm module.
    std::unique_ptr<class PulseStatsCollector> m_stats_collector; ///< Pointer to
the stat collector module.
    std::unique_ptr<class PulseLoader> m_loader; ///< Pointer to the data loader
module.

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    double m_update_frequency; ///< The frequency at which the simulation updates
    (in seconds).
    bool m_algo_enabled{true}; ///< Flag to enable or disable the traffic
    algorithm.
    std::atomic<RunState> m_run_state{
        RunState::Running}; ///< Current state of the simulation (running, paused,
    stopped).
};
} // namespace TrafficPulse

#endif // TRAFFICSYSTEM_H

#pragma once

#ifndef PULSE_INTERSECTION_H
#define PULSE_INTERSECTION_H

#include "core/PulseException.h"
#include "entities/PulseRoadConnection.h"
#include "entities/PulseTrafficLight.h"
#include "types/PulsePosition.h"

#include <stdexcept>
#include <string>
#include <unordered_map>
#include <vector>

namespace TrafficPulse
{
    /**
     * @class PulseIntersection
     * @brief Represents a traffic intersection in a graph-based model.
     */
    class PulseIntersection {
    public:
        /**
         * @brief Constructs an intersection with a given ID, position, and radius.
         *
         * @param intersection_id Unique identifier for this intersection.
         * @param position The (x, y) coordinates of this intersection.
         * @param tls_id (Optional) The ID of the traffic light associated with this
         intersection.
         */
        PulseIntersection(std::string intersection_id, const PulsePosition& position,
            std::string tls_id = {});

        /**
         * @brief Retrieves the intersection ID.
         *
         * @return The unique identifier of the intersection as a string.
         */
        std::string getId() const;

        /**
         * @brief Retrieves the position of the intersection.
         *
         * @return The (x, y) coordinates of the intersection.
         */
        PulsePosition getPosition() const;

        /**
         * @brief Adds the ID of an inbound lane that leads into this intersection.

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    *
    * @param lane_id The string ID of the lane, e.g. "edgeA_0".
    */
    void addInboundLane(const std::string& lane_id);

    /**
    * @brief Retrieves all inbound lane IDs that feed traffic into this
    intersection.
    *
    * @return A reference to a vector of lane strings.
    */
    const std::vector<std::string>& getInboundLanes() const;

    /**
    * @brief Adds a road connection between this intersection and another.
    *
    * @param road_id Unique identifier for the road connection.
    * @param intersection Pointer to the intersection at the other end of the
    road.
    * @param traffic_light Pointer to the traffic light controlling the road.
    * @param distance The distance in meters between the intersections.
    *
    * @throws std::invalid_argument if either the intersection or the traffic
    light pointer is null.
    * @throws PulseException if a connection with the given road_id already
    exists.
    */
    void
    addRoadConnection(int road_id, PulseIntersection* intersection,
    PulseTrafficLight* traffic_light, double distance);

    /**
    * @brief Retrieves all road connections from this intersection.
    *
    * @return A map of road ID to PulseRoadConnection.
    */
    const std::unordered_map<int, PulseRoadConnection>& getConnectedRoads() const;

    /**
    * @brief Retrieves the ID of the traffic light associated with this
    intersection.
    *
    * @return Traffic light ID associated with this intersection.
    */
    const std::string& getTrafficLightId() const;

private:
    std::string m_intersection_id; ///< Unique identifier for the intersection.
    std::string m_tls_id; ///< Traffic light ID associated with this intersection.
    PulsePosition m_position; ///< The (x, y) coordinates of the intersection.
    std::vector<std::string> m_inbound_lanes; ///< List of inbound lanes.
    std::unordered_map<int, PulseRoadConnection> m_connected_roads; ///< Road
    connections to other intersections.
};
} // namespace TrafficPulse

#endif // PULSE_INTERSECTION_H

#pragma once

#ifndef PULSEROADCONNECTION_H
#define PULSEROADCONNECTION_H

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

#include "entities/PulseTrafficLight.h"

namespace TrafficPulse
{
class PulseIntersection;

/**
 * @class PulseRoadConnection
 * @brief Represents a road connecting two intersections.
 */
class PulseRoadConnection {
public:
    /**
     * @brief Constructs a road connection between two intersections.
     *
     * @param intersection Reference to the connected intersection.
     * @param traffic_light Reference to the traffic light controlling this road.
     * @param distance Distance in meters between intersections.
     */
    PulseRoadConnection(PulseIntersection& intersection, PulseTrafficLight*
traffic_light, double distance);

    /**
     * @brief Retrieves the connected intersection.
     *
     * @return Reference to the connected intersection.
     */
    [[nodiscard]] PulseIntersection& getConnectedIntersection() const;

    /**
     * @brief Retrieves the traffic light controlling this road.
     *
     * @return Pointer to the traffic light.
     */
    [[nodiscard]] PulseTrafficLight* getTrafficLight() const;

    /**
     * @brief Retrieves the distance between intersections.
     *
     * @return Distance in meters.
     */
    [[nodiscard]] double getDistance() const;

    /**
     * @brief Updates the traffic light assigned to this road connection.
     *
     * @param traffic_light New traffic light.
     */
    void setTrafficLight(PulseTrafficLight* traffic_light);

    /**
     * @brief Updates the distance between intersections.
     *
     * @param distance New distance value.
     */
    void setDistance(double distance);

private:
    PulseIntersection& m_connected_intersection; ///< Destination node.
    PulseTrafficLight* m_traffic_light; ///< TLS controlling entry.
    double m_distance; ///< Length [m].

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

```

};
} // namespace TrafficPulse
#endif // PULSEROADCONNECTION_H

#pragma once

#ifndef PULSE_TRAFFIC_LIGHT_H
#define PULSE_TRAFFIC_LIGHT_H

#include "types/TrafficLightDurations.h"
#include "types/TrafficLightState.h"

#include <string>
#include <vector>

namespace TrafficPulse
{
/**
 * @class PulseTrafficLight
 * @brief Represents a traffic light in the simulation.
 */
class PulseTrafficLight {
public:
/**
 * @brief Constructs a traffic light with a unique ID and optional durations.
 *
 * @param traffic_light_id Unique string identifier for the traffic light.
 * @param durations Custom durations for each light state.
 */
explicit PulseTrafficLight(std::string traffic_light_id, const
TrafficLightDurations& durations = {});

/**
 * @brief Retrieves the traffic light ID.
 *
 * @return The unique string identifier.
 */
[[nodiscard]] std::string getId() const;

/**
 * @brief Sets the current state of the traffic light.
 *
 * @param state The new state (RED, GREEN, etc.).
 */
void setState(TrafficLightState state);

/**
 * @brief Retrieves the current state of the traffic light.
 *
 * @return The current state.
 */
[[nodiscard]] TrafficLightState getState() const;

/**
 * @brief Sets the duration for each state.
 *
 * @param durations The new durations.
 */
void setDurations(const TrafficLightDurations& durations);

/**
 * @brief Gets the current state durations.

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    *
    * @return The TrafficLightDurations struct.
    */
[[nodiscard]] TrafficLightDurations getDurations() const;

/**
 * @brief Adds a lane ID to the list of lanes this traffic light controls.
 *
 * @param lane_id The ID of the lane.
 */
void addControlledLane(const std::string& lane_id);

/**
 * @brief Retrieves all lane IDs controlled by this traffic light.
 *
 * @return A reference to a vector of lane IDs.
 */
[[nodiscard]] const std::vector<std::string>& getControlledLanes() const;

private:
    std::string m_traffic_light_id; ///< Unique identifier.
    TrafficLightState m_current_state; ///< Current state.
    TrafficLightDurations m_durations; ///< Durations for each state.
    std::vector<std::string> m_controlled_lanes; ///< Lanes controlled by this
light.
};
} // namespace TrafficPulse

#endif // PULSE_TRAFFIC_LIGHT_H

#pragma once

#ifndef PULSEVEHICLE_H
#define PULSEVEHICLE_H

#include "types/PulsePosition.h"
#include "types/PulseVehicleRole.h"
#include "types/PulseVehicleType.h"

#include <string>

namespace TrafficPulse
{
    /**
     * @class PulseVehicle
     * @brief Represents a vehicle moving through the traffic system.
     */
    class PulseVehicle {
    public:
        /**
         * @brief Constructs a vehicle with a unique ID, type, and role.
         *
         * @param vehicle_id Unique string identifier.
         * @param type Type of the vehicle (CAR, BUS, etc.).
         * @param role Role of the vehicle (NORMAL, EMERGENCY, etc.).
         * @param position Initial position of the vehicle.
         */
        explicit PulseVehicle(std::string vehicle_id,
                               PulseVehicleType type,
                               PulseVehicleRole role,
                               const PulsePosition& position);

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				83
Змн.	Арк.	№ докум.	Підпис	Дата		

```

/**
 * @brief Retrieves the vehicle ID.
 *
 * @return The unique string identifier.
 */
[[nodiscard]] std::string getId() const;

/**
 * @brief Retrieves the type of the vehicle.
 *
 * @return The vehicle type.
 */
[[nodiscard]] PulseVehicleType getType() const;

/**
 * @brief Retrieves the role of the vehicle.
 *
 * @return The vehicle role.
 */
[[nodiscard]] PulseVehicleRole getRole() const;

/**
 * @brief Retrieves the current position of the vehicle.
 *
 * @return The vehicle's position.
 */
[[nodiscard]] PulsePosition getPosition() const;

/**
 * @brief Updates the vehicle's position.
 *
 * @param new_position The new position of the vehicle.
 */
void updatePosition(const PulsePosition& new_position);

/**
 * @brief Gets the vehicle's current speed in m/s.
 *
 * @return The current speed of the vehicle.
 */
[[nodiscard]] virtual double getSpeed() const;

/**
 * @brief Sets the vehicle's current speed in m/s.
 *
 * @param speedVal The new speed of the vehicle.
 */
virtual void setSpeed(double speedVal);

private:
    std::string m_vehicle_id; ///< Unique identifier.
    PulseVehicleType m_type; ///< Type of vehicle.
    PulseVehicleRole m_role; ///< Role of vehicle.
    PulsePosition m_position; ///< Current position of the vehicle.
    double m_speed = 0.0; ///< Current speed of the vehicle in m/s.
};
} // namespace TrafficPulse

#endif // PULSEVEHICLE_H

#pragma once

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				84
Змн.	Арк.	№ докум.	Підпис	Дата		

```

#ifdef BASEEVENT_H
#define BASEEVENT_H

#include <string>
#include <typeindex>

namespace TrafficPulse
{
    /**
     * @class BaseEvent
     * @brief Abstract root for all EventBus messages.
     */
    class BaseEvent {
    public:
        virtual ~BaseEvent() = default;

        /** @brief Run-time type tag used for routing. */
        [[nodiscard]] virtual std::type_index getType() const = 0;

        /** @brief Demangled type name for debugging/logging. */
        [[nodiscard]] virtual std::string getName() const = 0;
    };

    /**
     * @tparam TPayload Plain struct that carries event data.
     */
    template <typename TPayload> class TypedEvent : public BaseEvent {
    public:
        explicit TypedEvent(TPayload payload)
            : m_payload(std::move(payload)) { }

        [[nodiscard]] std::type_index getType() const override {
            return typeid(TPayload);
        }

        [[nodiscard]] std::string getName() const override {
            return typeid(TPayload).name();
        }

        /// Access to the stored payload.
        const TPayload& getPayload() const {
            return m_payload;
        }

    private:
        TPayload m_payload; ///< Actual event data.
    };
} // namespace TrafficPulse

#endif // BASEEVENT_H

#pragma once

#ifdef IEVENTDISPATCHER_H
#define IEVENTDISPATCHER_H

#include "BaseEvent.h"

#include <memory>

namespace TrafficPulse
{

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

/**
 * @interface IEventDispatcher
 * @brief Base interface for classes that emit events to an EventBus.
 */
class IEventDispatcher {
public:
    virtual ~IEventDispatcher() = default;

    /**
     * Send an event to all subscribers.
     * @param event Shared pointer to a concrete BaseEvent.
     */
    virtual void dispatch(const std::shared_ptr<BaseEvent>& event) = 0;
};
} // namespace TrafficPulse

#endif // IEVENTDISPATCHER_H

#pragma once

#ifndef NETWORKLOADINGCOMPLETEDEVENT_H
#define NETWORKLOADINGCOMPLETEDEVENT_H

#include <string>

namespace TrafficPulse
{
    /**
     * @struct NetworkLoadingCompletedEvent
     * @brief Event triggered when network loading is completed.
     */
    struct NetworkLoadingCompletedEvent
    {
        std::string message; ///< Message associated with the event.
    };
} // namespace TrafficPulse

#endif // NETWORKLOADINGCOMPLETEDEVENT_H

#pragma once

#ifndef NETWORKLOADINGPROGRESSEVENT_H
#define NETWORKLOADINGPROGRESSEVENT_H

#include <string>

namespace TrafficPulse
{
    /**
     * @struct NetworkLoadingProgressEvent
     * @brief Event triggered during network loading to indicate progress.
     */
    struct NetworkLoadingProgressEvent
    {
        std::string message; ///< Message associated with the event.
        float progress; ///< Progress percentage (0.0 to 1.0).
    };
} // namespace TrafficPulse

#endif // NETWORKLOADINGPROGRESSEVENT_H

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІПЗ	Арк.
		Власенко О.В.				86
Змн.	Арк.	№ докум.	Підпис	Дата		

```

#pragma once

#ifndef NETWORKLOADINGSTARTEDEVENT_H
#define NETWORKLOADINGSTARTEDEVENT_H

#include <string>

namespace TrafficPulse
{
    /**
     * @struct NetworkLoadingStartedEvent
     * @brief Event triggered when network loading starts.
     */
    struct NetworkLoadingStartedEvent
    {
        std::string message; ///< Message associated with the event.
    };
} // namespace TrafficPulse

#endif // NETWORKLOADINGSTARTEDEVENT_H

#pragma once

#ifndef SIMULATIONENDEDEVENT_H
#define SIMULATIONENDEDEVENT_H

#include <string>

namespace TrafficPulse
{
    /**
     * @struct SimulationEndedEvent
     * @brief Event triggered when the simulation ends.
     */
    struct SimulationEndedEvent
    {
        std::string message; ///< Message associated with the end of the simulation.
    };
} // namespace TrafficPulse

#endif // SIMULATIONENDEDEVENT_H

#pragma once

#ifndef SIMULATIONSTARTEDEVENT_H
#define SIMULATIONSTARTEDEVENT_H

namespace TrafficPulse
{
    /**
     * @struct SimulationStartedEvent
     * @brief Event triggered when the simulation starts.
     */
    struct SimulationStartedEvent
    {
    };
} // namespace TrafficPulse

#endif // SIMULATIONSTARTEDEVENT_H

#pragma once

#ifndef SIMULATIONSTEPEVENT_H

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				87
Змн.	Арк.	№ докум.	Підпис	Дата		

```

#define SIMULATIONSTEPEVENT_H

namespace TrafficPulse
{
/**
 * @struct SimulationStepEvent
 * @brief Event triggered at each simulation step.
 */
struct SimulationStepEvent
{
    double sim_time; ///< Current simulation time in seconds
};
} // namespace TrafficPulse

#endif // SIMULATIONSTEPEVENT_H

#pragma once

#ifndef STATSUPDATEDEVENT_H
#define STATSUPDATEDEVENT_H

#include "types/PulseVehicleType.h"

#include <unordered_map>

namespace TrafficPulse
{
/**
 * @struct StatsUpdatedEvent
 * @brief Event triggered when the simulation statistics are updated.
 */
struct StatsUpdatedEvent
{
    double sim_time; ///< Current simulation time in seconds
    double avg_speed; ///< Average speed of vehicles in the simulation
    double max_speed; ///< Maximum speed of vehicles in the simulation
    std::unordered_map<PulseVehicleType, int> vehicle_type_counts; ///< Count of
vehicles by type
};
} // namespace TrafficPulse

#endif // STATSUPDATEDEVENT_H

#pragma once

#ifndef TRAFFICLIGHTCHANGEDEVENT_H
#define TRAFFICLIGHTCHANGEDEVENT_H

#include <string>

namespace TrafficPulse
{
/**
 * @struct TrafficLightChangedEvent
 * @brief Event triggered when the state of a traffic light changes.
 */
struct TrafficLightChangedEvent
{
    std::string traffic_light_id; ///< Unique identifier for the traffic light
    std::string new_state; ///< New state of the traffic light (e.g., "RED",
"GREEN", "YELLOW")
};
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				88
Змн.	Арк.	№ докум.	Підпис	Дата		

```

} // namespace TrafficPulse

#endif // TRAFFICLIGHTCHANGEDEVENT_H

#pragma once

#ifndef TRAFFICMODECHANGEDEVENT_H
#define TRAFFICMODECHANGEDEVENT_H

#include "types/TrafficControlMode.h"

namespace TrafficPulse
{
/**
 * @struct TrafficModeChangedEvent
 * @brief Dispatched when the traffic coordination mode changes.
 */
struct TrafficModeChangedEvent
{
    TrafficControlMode new_mode; ///< The new traffic control mode.
};
} // namespace TrafficPulse

#endif //TRAFFICMODECHANGEDEVENT_H

#pragma once

#ifndef VEHICLEMOVEDEVENT_H
#define VEHICLEMOVEDEVENT_H

#include <string>

namespace TrafficPulse
{
/**
 * @struct VehicleMovedEvent
 * @brief Represents an event where a vehicle has moved to a new position.
 */
struct VehicleMovedEvent
{
    std::string vehicle_id; ///< Unique identifier for the vehicle
    double x; ///< New x-coordinate of the vehicle
    double y; ///< New y-coordinate of the vehicle
};
} // namespace TrafficPulse

#endif // VEHICLEMOVEDEVENT_H

#pragma once

#ifndef GREENWAVEMANAGER_H
#define GREENWAVEMANAGER_H

#include "strategies/IGreenWaveStrategy.h"

#include <deque>
#include <functional>
#include <memory>
#include <string>
#include <unordered_map>
#include <vector>

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				89
Змн.	Арк.	№ докум.	Підпис	Дата		

```

namespace TrafficPulse
{
/**
 * @class GreenWaveManager
 * @brief Front-end that coordinates "green-wave" offsets via a pluggable
strategy.
 *
 * Keeps sliding-window queue samples and an EMA of observed speed so that
 * distance-based offsets can be converted to seconds.
 */
class GreenWaveManager {
public:
    explicit GreenWaveManager(std::shared_ptr<IGreenWaveStrategy> strategy);

/**
 * @brief Builds offsets for the given intersections.
 *
 * @param intersection_ids Ordered list of intersections in the corridor.
 * @param get_distance Callback to return distance between two IDs.
 * @param current_time Current simulation time.
 * @param queue_lengths Queue lengths for each intersection.
 * @param total_flow Total flow rate.
 *
 * @return Map TLS-ID → offset [s].
 *
 * Steps:
 * 1. Calculate the maximum intersection distance in the corridor.
 * 2. Ask the active strategy for raw metre offsets.
 * 3. Convert metres → seconds using the observed or implied speed.
 */
    std::unordered_map<std::string, double>
    buildOffsets(const std::vector<std::string>& intersection_ids,
                const std::function<double(const std::string&, const std::string&)>&
get_distance,
                double current_time,
                const std::vector<double>& queue_lengths,
                double total_flow) const;

/**
 * @brief Update the EMA speed using one probe vehicle.
 *
 * @param distance Distance travelled by the vehicle.
 * @param travel_time Travel time of the vehicle.
 */
    void registerVehicleSample(double distance, double travel_time) const;

/**
 * @brief Push a single queue-length reading into the sliding window.
 *
 * @param queue_length The length of the queue.
 */
    void registerQueueSample(double queue_length);

/** @return Current mean queue length over the sliding window. */
    [[nodiscard]] double getAvgQueue() const;

/** @return Currently active scheduling strategy. */
    [[nodiscard]] std::shared_ptr<IGreenWaveStrategy> getStrategy() const {
        return m_strategy;
    }

/** @brief Replace the active strategy at runtime. */

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

```

void setStrategy(std::shared_ptr<IGreenWaveStrategy> strategy);

/** @return Exponential-averaged platoon speed(m/s). */
double getObservedSpeed() const;

/** @return Longest intersection distance seen in the last build. */
double getLastMaxDist() const;

private:
    std::shared_ptr<IGreenWaveStrategy> m_strategy; ///< Active scheduling
    strategy.
    std::deque<double> m_q_samples; ///< Recent queue lengths.

    mutable double m_last_max_dist = 0.0; ///< last max distance between
    intersections
    mutable double m_last_cycle = 0.0; ///< last cycle length
    mutable double m_observed_speed = 0.0; ///< observed speed of the corridor
};
} // namespace TrafficPulse

#endif // GREENWAVEMANAGER_H

#pragma once

#ifndef BASELINEWAVESTRATEGY_H
#define BASELINEWAVESTRATEGY_H

#include "IGreenWaveStrategy.h"

namespace TrafficPulse
{
    /**
     * @class BaselineWaveStrategy
     * @brief Distance-based 'fixed speed' schedule.
     */
    class BaselineWaveStrategy : public IGreenWaveStrategy {
    public:
        explicit BaselineWaveStrategy(double target_speed_m_s, double cycle_length);

        /**
         * Build cumulative-distance offsets.
         *
         * Example: if distances between nodes are 120 m and 80 m, offsets become:
         * {A:0 m, B:120 m, C:200 m}.
         *
         * @return Map {intersection-id → offset in metres}.
         */
        std::unordered_map<std::string, double>
        calculateOffsets(const std::vector<std::string>& intersection_ids,
                        const std::function<double(const std::string&, const
                        std::string&)>& get_distance,
                        double current_time,
                        const std::vector<double>& queue_lengths,
                        double total_flow) const override;

        /** @return The fixed cycle length given in the constructor in seconds. */
        [[nodiscard]] double getCycleLength() const override;

    private:
        double m_speed; ///< Nominal free-flow speed in for UI only(m/s).
        double m_cycle; ///< Fixed cycle length (s).
    };
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

```

};

} // namespace TrafficPulse

#endif // BASELINEWAVESTRATEGY_H

#pragma once

#ifndef HYBRIDWAVESTRATEGY_H
#define HYBRIDWAVESTRATEGY_H

#include "controllers/PIDController.h"
#include "strategies/IGreenWaveStrategy.h"
#include "utils/Settings.h"

namespace TrafficPulse
{
/**
 * @class HybridWaveStrategy
 * @brief Combines Webster formula with PID to adapt cycle length.
 *
 * Webster:  $C = (1.5 * L + 5) / (1 - \sum(y))$ 
 * PID then nudges `C` +- cap move using average queue as bias.
 */

class HybridWaveStrategy : public IGreenWaveStrategy {
public:
    /**
     * @param inter_green Lost-time per phase: yellow + all-red [s].
     * @param phases Number of phases per cycle.
     * @param max_cycle Upper bound for cycle length [s].
     * @param cfg Adaptive PID configuration from YAML.
     */
    explicit HybridWaveStrategy(double inter_green, int phases, double max_cycle,
    const Settings::Adaptive& cfg);

    /**
     * @brief Calculates offsets for each intersection.
     *
     * @return Map of intersection ID to offset in seconds.
     *
     * Steps:
     * 1. Compute Webster cycle from current flow & queues.
     * 2. If adaptive mode is enabled & warm-up passed -> PID adjusts the cycle.
     * 3. Clamp movement to +- cap move and update the target.
     * 4. Convert cumulative distance to phase index:  $d / (cycle / phases)$ .
     */
    std::unordered_map<std::string, double>
    calculateOffsets(const std::vector<std::string>& intersection_ids,
    const std::function<double(const std::string&, const
    std::string&)>& get_distance,
    double current_time,
    const std::vector<double>& queue_lengths,
    double total_flow) const override;

    /** @brief Cycle length in seconds. */
    [[nodiscard]]
    double getCycleLength() const override {
        return m_last_cycle;
    }
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				92
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    /** @brief Number of phases per cycle. */
    [[nodiscard]]
    bool isAdaptiveEnabled() const override {
        return m_enabled;
    }

    /** @brief Minimum time between PID updates. */
    [[nodiscard]]
    double getCheckEvery() const override {
        return m_check_every;
    }

    /** @brief Direct access to the embedded PID. */
    PIDController& pid();

    /** @brief Direct access to the embedded PID. */
    const PIDController& pid() const;

private:
    /** @brief Webster cycle from sum(y). Result clamped to [30 s, max cycle]. */
    double computeWebsterCycle(double sum_y) const;

    /** @brief Calculates sum(y) based on flow share and average queue. */
    double computeSumY(double flow_rate, size_t N, const std::vector<double>&
queue_lengths) const;

    double m_inter_green; ///< Lost-time per phase [s].
    int m_phases; ///< Number of phases per cycle.
    double m_max_cycle; ///< Maximum allowed cycle [s].

    mutable PIDController m_pid; ///< Cycle-length PID controller.
    mutable double m_last_cycle{0.0}; ///< Last applied cycle [s].
    bool m_enabled; ///< Adaptive mode flag.
    double m_check_every; ///< Re-evaluation period [s].
    double m_min_hold; ///< Minimum hold before next PID move [s].
    double m_cap_move; ///< Absolute cap for delta cycle per update [s].
    double m_avg_queue_weight; ///< Weight of queue component in sum(y).
    double m_warmup; ///< Warm-up time before PID starts [s].
    mutable double m_last_adapt_time = 0.0; ///< Timestamp of last PID update.
};
} // namespace TrafficPulse

#endif // HYBRIDWAVESTRATEGY_H

#pragma once

#ifndef IGREENWAVESTRATEGY_H
#define IGREENWAVESTRATEGY_H

#include <functional>
#include <string>
#include <unordered_map>
#include <vector>

namespace TrafficPulse
{
    /**
     * @class IGreenWaveStrategy
     * @brief Interface for different green wave scheduling strategies.
     */
    class IGreenWaveStrategy {
    public:

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				93
Змн.	Арк.	№ докум.	Підпис	Дата		

```

virtual ~IGreenWaveStrategy() = default;

/**
 * @brief Calculates offsets for each intersection.
 *
 * @param intersection_ids Ordered a corridor list.
 * @param get_distance Callback to return distance between two IDs.
 * @param current_time Current simulation time.
 * @param queue_lengths Queue lengths for each intersection.
 * @param total_flow Total flow rate.
 *
 * @return Map of intersection ID to offset (seconds).
 */
virtual std::unordered_map<std::string, double>
calculateOffsets(const std::vector<std::string>& intersection_ids,
                const std::function<double(const std::string&, const
std::string&)>& get_distance,
                double current_time,
                const std::vector<double>& queue_lengths,
                double total_flow) const = 0;

/**
 * @return Current cycle length in seconds as understood by the strategy.
 */
[[nodiscard]] virtual double getCycleLength() const = 0;

/**
 * @return True if the strategy supports periodic adaptive recalculation.
 */
[[nodiscard]] virtual bool isAdaptiveEnabled() const {
    return false;
}

/**
 * @return Suggested period, in seconds, between adaptive checks.
 *         Zero means 'no recommendation'.
 */
[[nodiscard]] virtual double getCheckEvery() const {
    return 0.0;
}
};
} // namespace TrafficPulse

#endif // IGREENWAVESTRATEGY_H

#pragma once

#ifndef MAXPRESSUREWAVESTRATEGY_H
#define MAXPRESSUREWAVESTRATEGY_H

#include "IGreenWaveStrategy.h"

namespace TrafficPulse
{
/**
 * @class MaxPressureWaveStrategy
 * @brief Distributes green waves by queue lengths – the longer the queue, the
earlier the phase
 *
 * Offset infinite queue length: longer queue -> earlier green within the cycle.
 */

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				94
Змн.	Арк.	№ докум.	Підпис	Дата		

```

class MaxPressureWaveStrategy : public IGreenWaveStrategy {
public:
    /**
     * @param cycle_length Cycle length used to scale offsets(s).
     */
    explicit MaxPressureWaveStrategy(double cycle_length);

    /**
     * @brief Generate time offsets from queue lengths.
     *
     * Formula:
     * 
$$\text{offset}_i = (1 - \text{queue}_i / \text{sum\_queue}) * \text{cycle}$$

     * where
     * - queue_i - current queue length at intersection *i*
     * - sum_queue - sum of all queue lengths in the corridor
     * - cycle - fixed cycle length supplied to the constructor
     *
     * The largest queue gets an offset close to 0s (earliest green);
     * smaller queues are delayed proportionally within the cycle.
     *
     * If sum(queue) equals zero every intersection receives offset = cycle,
     * which produces no phase shift but keeps the map valid.
     *
     * @param intersection_ids Ordered a list of intersection IDs.
     * @param get_distance Not used by this strategy.
     * @param current_time Not used.
     * @param queue_lengths Queue length at each intersection.
     * @param total_flow Not used.
     *
     * @return Map {intersection-id -> offset in seconds}.
     */
    std::unordered_map<std::string, double>
    calculateOffsets(const std::vector<std::string>& intersection_ids,
                    const std::function<double(const std::string&, const
std::string&)>& get_distance,
                    double current_time,
                    const std::vector<double>& queue_lengths,
                    double total_flow) const override;

    /** @return Fixed cycle length in seconds. */
    [[nodiscard]] double getCycleLength() const override;

private:
    double m_cycle; ///< Cycle length(s) used to scale offsets.
};

} // namespace TrafficPulse

#endif // MAXPRESSUREWAVESTRATEGY_H

#pragma once

#ifdef GREENWAVECORIDOR_H
#define GREENWAVECORIDOR_H

#include <string>
#include <vector>

namespace TrafficPulse
{
    /**
     * @struct GreenWaveCorridor

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				95
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    * @brief Represents a path/sequence of intersections for the green wave.
    */
struct GreenWaveCorridor
{
    std::vector<std::string> intersection_ids; ///< Ordered intersections along
this path
    double target_speed_m_s; ///< The speed to maintain the wave
};
} // namespace TrafficPulse

#endif // GREENWAVECORIDOR_H

#pragma once

#ifndef LANESMOOTHER_H
#define LANESMOOTHER_H

#include <deque>

namespace TrafficPulse
{
/**
    * @struct LaneSmoother
    * @brief A small helper to do rolling or exponential smoothing for numeric
samples.
    */
struct LaneSmoother
{
    std::deque<double> samples; ///< Samples for rolling average
    static constexpr size_t MAX_SIZE = 5; ///< Max size of the rolling average

    /**
        * @brief Smoothing factor for an exponential average
        *
        * Exponential smoothing factor (0 < alpha <= 1).
        */
        double alpha = 0.3;
        bool exp_initied = false; ///< Flag to check if the exponential average has
been initialized
        double exp_value = 0.0; ///< Last exponential average

    /**
        * @brief Add a sample to the rolling average and update the exponential
average.
        *
        * @param val The value to add.
        */
        void addSample(const double val) {
            samples.push_back(val);
            if(samples.size() > MAX_SIZE) {
                samples.pop_front();
            }

            // Exponential
            if(!exp_initied) {
                exp_value = val;
                exp_initied = true;
            }
            else {
                // Exponential smoothing formula
                exp_value = alpha * val + (1.0 - alpha) * exp_value;
            }
        }
    }
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				96
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    /** @return The rolling average of the samples. */
    [[nodiscard]] double getRollingAverage() const {
        if(samples.empty())
            return 0.0;

        double sum = 0.0;

        for(const auto v : samples)
            sum += v;

        return sum / static_cast<double>(samples.size());
    }

    /** @return The exponential average of the samples. */
    [[nodiscard]] double getExponential() const {
        return exp_initd ? exp_value : 0.0;
    }
};
} // namespace TrafficPulse

#endif // LANESMOOTHER_H

#pragma once

#ifndef METRICS_H
#define METRICS_H

#include <map>
#include <string>

namespace TrafficPulse
{
    /**
     * @struct Metrics
     * @brief Holds performance metrics for one simulation run.
     */
    struct Metrics
    {
        double avg_travel_time{}; ///< s
        double avg_waiting_time{}; ///< s
        double avg_speed{}; ///< m/s
        int throughput{}; ///< trips
        int num_stops{};
        double avg_queue_length{}; ///< s
        double time_loss{}; ///< veh*s
        double halted_peak{}; ///< veh
        double density_peak{}; ///< veh/km
        double queue_peak{}; ///< veh
    };

    struct MetricsPair
    {
        Metrics base; ///< Base scenario
        Metrics custom; ///< Custom scenario
    };

    using MetricsMap = std::map<std::string, MetricsPair>; // scenario -> pair
} // namespace TrafficPulse

#endif // METRICS_H

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				97
Змн.	Арк.	№ докум.	Підпис	Дата		

```

#pragma once

#ifndef PULSEENTITYTYPE_H
#define PULSEENTITYTYPE_H

namespace TrafficPulse
{
    /**
     * @enum PulseEntityType
     * @brief Defines the types of entities that can be created by the factory.
     */
    enum class PulseEntityType
    {
        INTERSECTION,
        TRAFFIC_LIGHT,
        VEHICLE
    };
} // namespace TrafficPulse

#endif // PULSEENTITYTYPE_H

#pragma once

#ifndef PULSEERRORCODE_H
#define PULSEERRORCODE_H

namespace TrafficPulse
{
    /**
     * @enum PulseErrorCode
     * @brief Represents specific types of errors in the Pulse library.
     */
    enum class PulseErrorCode
    {
        Unknown,
        InvalidFilePath,
        AlreadyRunning,
        NotRunning,
        ParsingError,
        DuplicateEntity
    };
} // namespace TrafficPulse

#endif // PULSEERRORCODE_H

#pragma once

#ifndef PULSEEVENTS_H
#define PULSEEVENTS_H

namespace TrafficPulse
{
    /**
     * @class PulseEvents
     * @brief Enum to represent different types of events that can occur during the
     simulation.
     */
    enum class PulseEvents
    {
        SIMULATION_START, ///< Simulation has started.
        SIMULATION_STEP,  ///< Simulation has advanced by one step.
        SIMULATION_END,   ///< Simulation has ended.
    };
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				98
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    VEHICLE_STATUS_CHANGE, ///< Vehicle has changed status.
    TRAFFIC_LIGHT_CHANGE, ///< Traffic light has changed state.
    LOADING_START, ///< Loading of network data has started.
    LOADING_PROGRESS, ///< Loading of network data is in progress.
    LOADING_COMPLETE, ///< Loading of network data has completed.
    STATS_UPDATE, ///< Statistics have been updated.
};
} // namespace TrafficPulse

#endif // PULSEEVENTS_H

#pragma once

#ifndef PULSEINTERSECTIONSTATS_H
#define PULSEINTERSECTIONSTATS_H

namespace TrafficPulse
{
/**
 * @struct PulseIntersectionStats
 * @brief Structure to hold statistics for intersections in the Traffic Pulse
simulation.
 */
struct PulseIntersectionStats
{
    double total_wait_time = 0.0; // sum of wait times
    int vehicles_passed = 0; // total count across sim
};
} // namespace TrafficPulse

#endif // PULSEINTERSECTIONSTATS_H

#pragma once

#ifndef PULSELANEINFO_H
#define PULSELANEINFO_H

namespace TrafficPulse
{
/**
 * @struct PulseLaneInfo
 * @brief Structure to hold lane information in the Traffic Pulse simulation.
 */
struct PulseLaneInfo
{
    double speed;
    double length;
};
} // namespace TrafficPulse
#endif // PULSELANEINFO_H

#pragma once

#ifndef LANESTATS_H
#define LANESTATS_H

#include "LaneSmoother.h"

namespace TrafficPulse
{
/**
 * @struct PulseLaneStats

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				99
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    * @brief Stores lane-based stats (speed, occupancy, # waiting, etc.).
    */
struct PulseLaneStats
{
    LaneSmoother speed_smoother;
    LaneSmoother occupancy_smoother;
    int waiting_count = 0;
};
} // namespace TrafficPulse

#endif // LANESTATS_H

#pragma once

#ifndef PULSEPOSITION_H
#define PULSEPOSITION_H

namespace TrafficPulse
{
    /**
     * @struct PulsePosition
     * @brief Struct to hold the position of an object in 2D space.
     */
    struct PulsePosition
    {
        double x; ///< X-coordinate of the object.
        double y; ///< Y-coordinate of the object.

        /**
         * @brief Constructor for a position.
         *
         * @param x X-coordinate.
         * @param y Y-coordinate.
         */
        PulsePosition(const double x = 0.0, const double y = 0.0)
            : x(x)
            , y(y) { }

        /**
         * @brief Equality operator for comparing positions.
         */
        bool operator==(const PulsePosition& other) const {
            return x == other.x && y == other.y;
        }
    };
} // namespace TrafficPulse

#endif // PULSEPOSITION_H

#pragma once

#ifndef PULSETRAFFICLIGHTSTATS_H
#define PULSETRAFFICLIGHTSTATS_H

#include "types/TrafficLightState.h"

namespace TrafficPulse
{
    /**
     * @struct PulseTrafficLightStats
     * @brief Structure to hold statistics for traffic lights in the Traffic Pulse
     simulation.
    */

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				100
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    */
    struct PulseTrafficLightStats
    {
        double time_green = 0.0;
        double time_red = 0.0;
        int vehicles_stopped = 0; // Aggregated or per step
        int cycle_count = 0; // How many times it changed state
        TrafficLightState last_state;
        double last_state_change_time = 0.0;
    };
} // namespace TrafficPulse

#endif // PULSETRAFFICLIGHTSTATS_H

#pragma once

#ifndef PULSEVEHICLEROLE_H
#define PULSEVEHICLEROLE_H

namespace TrafficPulse
{
    /**
     * @class PulseVehicleRole
     * @brief Enum to represent the role of a vehicle in the simulation.
     */
    enum class PulseVehicleRole
    {
        NORMAL, ///< Normal vehicle.
        EMERGENCY, ///< Emergency vehicle.
    };
} // namespace TrafficPulse

#endif // PULSEVEHICLEROLE_H

#pragma once

#ifndef PULSEVEHICLETYPE_H
#define PULSEVEHICLETYPE_H

namespace TrafficPulse
{
    /**
     * @class PulseVehicleType
     * @brief Enum to represent the type of vehicle in the simulation.
     */
    enum class PulseVehicleType
    {
        CAR,
        BUS,
        MOTORCYCLE,
        TRUCK,
        TRAM,
        E_SCOOTER,
        BICYCLE,
        PEDESTRIAN,
        EMERGENCY,
    };
} // namespace TrafficPulse

#endif // PULSEVEHICLETYPE_H

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		101

```

#pragma once

#ifndef RUNSTATE_H
#define RUNSTATE_H

namespace TrafficPulse
{
    /**
     * @class RunState
     * @brief Enumeration representing the state of a simulation.
     */
    enum class RunState
    {
        Running,
        Paused,
        Stopped
    };
} // namespace TrafficPulse

#endif //RUNSTATE_H

#pragma once

#ifndef STATSSNAPSHOT_H
#define STATSSNAPSHOT_H

#include <string>
#include <unordered_map>

#include "types/PulseIntersectionStats.h"
#include "types/PulseVehicleType.h"

namespace TrafficPulse
{
    /**
     * @struct StatsSnapshot
     * @brief Represents a consolidated snapshot of key statistics at a given
     simulation time.
     */
    struct StatsSnapshot
    {
        double sim_time; ///< The simulation time at which this snapshot was taken.
        double avg_speed; ///< Average speed among all vehicles.
        double max_speed; ///< Maximum speed among all vehicles.
        std::unordered_map<PulseVehicleType, int> vehicle_type_counts; ///< Count of
vehicles by type.
        std::unordered_map<std::string, PulseIntersectionStats> intersection_map; ///<
Intersection statistics.
        std::unordered_map<std::string, PulseTrafficLightStats> traffic_light_map;
        ///< Traffic light statistics.
    };
} // namespace TrafficPulse

#endif // STATSSNAPSHOT_H

#pragma once

#ifndef SUMOOUTPUT_H
#define SUMOOUTPUT_H

#include <string>

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				102
Змн.	Арк.	№ докум.	Підпис	Дата		

```

namespace TrafficPulse
{
/**
 * @struct SumoOutput
 * @brief Structure to hold the output file information for SUMO
 */
struct SumoOutput
{
    std::string flag;
    std::string file;
    bool hasValue() const {
        return !file.empty();
    }
};
} // namespace TrafficPulse

#endif //SUMOOUTPUT_H

#pragma once

#ifndef TRAFFICMODE_H
#define TRAFFICMODE_H

#include <fmt/format.h>
#include <yaml-cpp/node/node.h>

namespace TrafficPulse
{
/**
 * @enum TrafficControlMode
 * @brief Describes the global traffic coordination mode.
 */
enum class TrafficControlMode
{
    Auto,
    Baseline,
    Hybrid,
    MaxPressure
};
} // namespace TrafficPulse

template <> struct fmt::formatter<TrafficPulse::TrafficControlMode> :
    fmt::formatter<std::string>
{
    template <typename FormatContext>
    auto format(const TrafficPulse::TrafficControlMode mode, FormatContext& ctx)
    const {
        std::string name;
        switch(mode) {
            case TrafficPulse::TrafficControlMode::Auto:
                name = "Auto";
                break;
            case TrafficPulse::TrafficControlMode::Baseline:
                name = "Normal";
                break;
            case TrafficPulse::TrafficControlMode::Hybrid:
                name = "Hybrid";
                break;
            case TrafficPulse::TrafficControlMode::MaxPressure:
                name = "MaxPressure";
                break;
            default:

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				103
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        name = "Unknown";
        break;
    }
    return fmt::formatter<std::string>::format(name, ctx);
}

};

namespace YAML
{
template <> struct convert<TrafficPulse::TrafficControlMode>
{
    static bool decode(const Node& node, TrafficPulse::TrafficControlMode& mode) {
        if(!node.IsScalar()) {
            return false;
        }

        const std::string value = node.as<std::string>();
        if(value == "Auto") {
            mode = TrafficPulse::TrafficControlMode::Auto;
        }
        else if(value == "Baseline") {
            mode = TrafficPulse::TrafficControlMode::Baseline;
        }
        else if(value == "Hybrid") {
            mode = TrafficPulse::TrafficControlMode::Hybrid;
        }
        else if(value == "MaxPressure") {
            mode = TrafficPulse::TrafficControlMode::MaxPressure;
        }
        else {
            return false;
        }
        return true;
    }
};
} // namespace YAML

#endif //TRAFFICMODE_H

#pragma once

#ifndef TRAFFICLIGHTDURATIONS_H
#define TRAFFICLIGHTDURATIONS_H

namespace TrafficPulse
{
/**
 * @struct TrafficLightDurations
 * @brief Struct to hold durations for different traffic light states.
 */
struct TrafficLightDurations
{
    double red; ///< Duration of red light.
    double yellow; ///< Duration of yellow light.
    double green; ///< Duration of green light.
    double walk; ///< Duration of walk signal for pedestrians.
    double dont_walk; ///< Duration of don't walk signals for pedestrians.

    /** @brief Default constructor with default timings in seconds. */
    TrafficLightDurations(const double red = 30.0,
                          const double yellow = 5.0,
                          const double green = 30.0,

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				104
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        const double walk = 15.0,
        const double dont_walk = 5.0)

        : red(red)
        , yellow(yellow)
        , green(green)
        , walk(walk)
        , dont_walk(dont_walk) { }
};
} // namespace TrafficPulse

#endif // TRAFFICLIGHTDURATIONS_H

#pragma once

#ifndef TRAFFICLIGHTSTATE_H
#define TRAFFICLIGHTSTATE_H

namespace TrafficPulse
{
    /**
     * @brief Enum representing the state of the traffic light.
     */
    enum class TrafficLightState
    {
        RED,
        YELLOW,
        GREEN,
        UNKNOWN, ///< Traffic light is in an unknown state, flashing yellow to
        indicate a malfunction.
        WALK, ///< Pedestrian walk signal.
        DONT_WALK ///< Pedestrian doesn't walk signal.
    };
} // namespace TrafficPulse

#endif // TRAFFICLIGHTSTATE_H

#pragma once

#ifndef BFSUTILS_H
#define BFSUTILS_H

#include <functional>
#include <queue>
#include <string>
#include <unordered_map>
#include <vector>

namespace TrafficPulse
{
    /**
     * @class BFS
     * @brief Static helper that performs a breadth-first search on an implicit graph
     supplied via a callback.
     */
    class BFS {
    public:
        /**
         * @brief Performs a breadth-first search starting from a given root,
         * using a callback to retrieve neighbors. Returns the visited
         * nodes in order of visitation.
         *
         * @param root_id The ID of the root node to start BFS from.
         */

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				105
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    * @param neighbors_of A callback that, given a node ID, returns a list of
    adjacent node IDs.
    * @param max_depth The maximum depth to search.
    *
    * @return A vector of node IDs in the order they were visited by BFS.
    */
    static std::vector<std::string>
    runBFS(const std::string& root_id,
           const std::function<std::vector<std::string>(const std::string&)>&
neighbors_of,
           const std::size_t max_depth = SIZE_MAX) {
        std::vector<std::string> visited_order;
        if(root_id.empty()) {
            return visited_order;
        }

        std::unordered_map<std::string, bool> visited;
        std::queue<std::string> q;

        q.push(root_id);
        visited[root_id] = true;
        std::unordered_map<std::string, std::size_t> depth;
        depth[root_id] = 0;

        while(!q.empty()) {
            auto current_id = q.front();
            q.pop();

            visited_order.push_back(current_id);

            if(depth[current_id] >= max_depth)
                continue;

            for(const auto& neighbor_id : neighbors_of(current_id))
                if(!visited[neighbor_id]) {
                    visited[neighbor_id] = true;
                    depth[neighbor_id] = depth[current_id] + 1;
                    q.push(neighbor_id);
                }
        }

        return visited_order;
    }
};
} // namespace TrafficPulse

#endif // BFSUTILS_H

#pragma once

#ifndef FILEUTILS_H
#define FILEUTILS_H

#include <filesystem>
#include <string>

namespace TrafficPulse
{
    /**
     * @class File
     * @brief Simple utility for path resolution.

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				106
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    */
class File {
public:
    /**
     * @brief Resolve a path relative to application dir if not absolute.
     *
     * @param input The input path to resolve.
     *
     * @return Resolved an absolute path.
     *
     * @throws std::runtime_error if the file cannot be found.
     */
    static std::filesystem::path resolvePath(const std::string& input) {
        std::filesystem::path p(input);

        if(p.is_absolute()) {
            return p;
        }

        if(const auto cwd = std::filesystem::current_path() / p;
std::filesystem::exists(cwd)) {
            return cwd;
        }

#ifdef TRAFFIC_PULSE_MODULE_BUILD_DIR
        auto mod = std::filesystem::path(TRAFFIC_PULSE_MODULE_BUILD_DIR) / p;
        if(std::filesystem::exists(mod)) {
            return mod;
        }
#endif

        throw std::runtime_error("File::resolvePath: cannot find " + input);
    }
};

} // namespace TrafficPulse

#endif //FILEUTILS_H

#pragma once

#ifndef LOGUTILS_H
#define LOGUTILS_H

#include <cstdlib>
#include <spdlog/sinks/rotating_file_sink.h>
#include <spdlog/sinks/stdout_color_sinks.h>
#include <spdlog/spdlog.h>
#include <string_view>

namespace TrafficPulse
{
    /**
     * @class Log
     * @brief Utility class for initializing logging with spdlog.
     */
    class Log {
    public:
        /**
         * @brief Initializes the logging system.
         *
         * @param level The logging level.
         */

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІІЗ	Арк.
		Власенко О.В.				107
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        * @param to_file Whether to log to a file.
        */
        static void init(std::string_view level = "info", bool to_file = true);
};
} // namespace TrafficPulse

#endif //LOGUTILS_H

#pragma once

#ifndef METRICS_UTILS_H
#define METRICS_UTILS_H

#include "types/Metrics.h"

#include <iosfwd>
#include <iostream>
#include <sstream>
#include <string>
#include <vector>

namespace TrafficPulse
{
    // TODO: docss
    class MetricsUtils {
    public:
        static MetricsMap readCsv(const std::string& csv_path); // throw-on-fail
        static void printTable(const MetricsMap& mm, std::ostream& os = std::cout);

    private:
        static std::vector<std::string> split(const std::string& s, char d) {
            std::vector<std::string> v;
            std::stringstream ss(s);
            std::string p;
            while(std::getline(ss, p, d))
                v.push_back(p);
            return v;
        }
        static std::string canonical(std::string s) {
            if(auto p = s.find('('); p != std::string::npos)
                s.erase(p);
            s.erase(s.find_last_not_of(" \t\r\n") + 1);
            return s;
        }
    };
} // namespace TrafficPulse

#endif //METRICS_UTILS_H

#pragma once

#ifndef SETTINGSUTILS_H
#define SETTINGSUTILS_H

#include "types/TrafficControlMode.h"
#include <yaml-cpp/yaml.h>

namespace TrafficPulse
{
    /**
     * @struct Settings

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		108

```

* @brief Configuration settings for the TrafficPulse system.
*/
struct Settings
{
    /** @brief Parameters that influence green-wave algorithms. */
    struct Algo
    {
        double max_cycle{90};
        std::size_t max_corr{5};
        double emergency_thr{80};
        std::size_t bfs_depth{5};
        double default_speed{13};
        int inter_green{3};
        int phases{4};
        TrafficControlMode force_mode{TrafficControlMode::Auto};
    } algo;

    /** @brief Thresholds used by mode selection & Max-Pressure detector. */
    struct FlowThresh
    {
        double low{0.2};
        double high{1.0};
        double min_hold{120};
        struct
        {
            double threshold{30}, per_inter{20};
        } queue;
        struct
        {
            double max_pressure{30}, other{20};
        } dwell;
    } flow;

    /** @brief Parameters for adaptive PID cycle tuning. */
    struct Adaptive
    {
        bool enabled{false};
        int check_every{30};
        int warmup{0};

        struct PID
        {
            double Kp{20}, Ki{0.01}, Kd{0.02};
            double cap_move{10};
            int min_hold{10};
            double avg_q_w{0.3};
            double init_set_pt{0.5};
        } pid;

        struct Tune
        {
            bool enabled{false};
            double retune_every{300};
            double min_hold{120};
            double queue_w{0.3};
            struct
            {
                double Kp{30}, Ki{1}, Kd{1};
            } lim;
        } tune;
    } adaptive;
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				109
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    /**
     * @brief Load YAML from a path and return populated Settings.
     *
     * @param path Path to the YAML file.
     *
     * @throws std::runtime_error on parse/validation errors.
     */
    static Settings load(const std::string& path = "config/tp_config.yaml");
};

} // namespace TrafficPulse

#endif // SETTINGSUTILS_H

#pragma once

#ifndef STATSUTILS_H
#define STATSUTILS_H

#include "types/PulsePosition.h"
#include <cmath>

namespace TrafficPulse
{
    /**
     * @class Stats
     * @brief Utility class for statistical calculations.
     */
    class Stats {
    public:
        /**
         * @brief Computes Euclidean distance between two positions.
         *
         * @param p1 The first position.
         * @param p2 The second position.
         *
         * @return The Euclidean distance.
         */
        static double euclideanDistanceBetween(const PulsePosition& p1, const
PulsePosition& p2) {
            const double dx = p1.x - p2.x;
            const double dy = p1.y - p2.y;
            return std::sqrt(dx * dx + dy * dy);
        }
    };
} // namespace TrafficPulse
#endif // STATSUTILS_H

#pragma once

#ifndef SUMOUTILS_H
#define SUMOUTILS_H

#include <string>

#include "types/PulseVehicleRole.h"
#include "types/PulseVehicleType.h"

namespace TrafficPulse
{
    /**
     * @class Sumo

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				110
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    * @brief Utility class for interpreting SUMO types to Pulse types.
    */
class Sumo {
public:
    /**
     * @brief Tries to interpret a SUMO vehicle "type" string and returns a
     PulseVehicleType.
     * This is partial logic—further expansions can be done in the future.
     *
     * @param sumo_type_str The SUMO vehicle type string.
     */
    static PulseVehicleType deduceVehicleTypeFromSumo(const std::string&
sumo_type_str) {
        if(sumo_type_str == "passenger") {
            return PulseVehicleType::CAR;
        }
        if(sumo_type_str == "truck" || sumo_type_str == "trailer") {
            return PulseVehicleType::TRUCK;
        }
        if(sumo_type_str == "emergency") {
            return PulseVehicleType::EMERGENCY;
        }

        return PulseVehicleType::CAR;
    }

    /**
     * @brief Tries to interpret a role from a SUMO property or other logic.
     * This is partial logic—further expansions can be done in the future.
     *
     * @param sumo_role_str The SUMO vehicle role string.
     */
    static PulseVehicleRole deduceVehicleRoleFromSumo(const std::string&
sumo_role_str) {
        if(sumo_role_str == "police" || sumo_role_str == "fire" || sumo_role_str ==
"ambulance") {
            return PulseVehicleRole::EMERGENCY;
        }

        return PulseVehicleRole::NORMAL;
    }
};
} // namespace TrafficPulse
#endif // SUMOUTILS_H

#pragma once

#ifndef XMLUTILS_H
#define XMLUTILS_H

#include <pugixml.hpp>
#include <types/Metrics.h>

namespace TrafficPulse
{
    /**
     * @class XML
     * @brief XML utility class for parsing and manipulating XML documents.
     */
    class XML {
    public:
        /**

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІПЗ	Арк.
		Власенко О.В.				111
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    * @brief Helper to count the number of child nodes that match a certain
    predicate.
    *
    * @param parent The parent node to search within.
    * @param child_name The name of the child nodes to look for.
    * @param pred The predicate function to apply to each child node.
    */
    template <typename Pred>
    static int count_children_if(const pugi::xml_node& parent, const char*
child_name, Pred pred) {
        int count = 0;

        for(auto node : parent.children(child_name)) {
            if(pred(node)) {
                count++;
            }
        }

        return count;
    }

    static Metrics parseTripInfo(const std::string& file);
    static void enrichFromSummary(const std::string& file, Metrics&);
    static void enrichFromStats(const std::string& file, Metrics&);
    static void enrichFromQueue(const std::string& file, Metrics&);
    static void enrichFromEdge(const std::string& file, Metrics&);

private:
    static bool loadDoc(const std::string& fn, pugi::xml_document& d);
};
} // namespace TrafficPulse

#endif // XMLUTILS_H

#pragma once

// --- Controllers ---
#include "controllers/AutoTuner.h"
#include "controllers/MetaController.h"
#include "controllers/PIDController.h"
#include "controllers/TrafficModeController.h"

// --- Core ---
#include "core/EventBus.h"
#include "core/PulseDataManager.h"
#include "core/PulseException.h"
#include "core/PulseLoader.h"
#include "core/PulseStatsCollector.h"
#include "core/PulseTrafficAlgo.h"
#include "core/SumoIntegration.h"
#include "core/TrafficSystem.h"

// --- Entities ---
#include "entities/PulseIntersection.h"
#include "entities/PulseRoadConnection.h"
#include "entities/PulseTrafficLight.h"
#include "entities/PulseVehicle.h"

// --- Events ---
#include "events/BaseEvent.h"
#include "events/IEventDispatcher.h"
#include "events/NetworkLoadingCompletedEvent.h"

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				112
Змн.	Арк.	№ докум.	Підпис	Дата		

```

#include "events/NetworkLoadingProgressEvent.h"
#include "events/NetworkLoadingStartedEvent.h"
#include "events/SimulationEndedEvent.h"
#include "events/SimulationStartedEvent.h"
#include "events/SimulationStepEvent.h"
#include "events/StatsUpdatedEvent.h"
#include "events/TrafficLightChangedEvent.h"
#include "events/TrafficModeChangedEvent.h"
#include "events/VehicleMovedEvent.h"

// --- Services ---
#include "services/GreenWaveManager.h"

// --- Strategies ---
#include "strategies/BaselineWaveStrategy.h"
#include "strategies/HybridWaveStrategy.h"
#include "strategies/IGreenWaveStrategy.h"
#include "strategies/MaxPressureWaveStrategy.h"

// --- Types ---
#include "types/GreenWaveCoridor.h"
#include "types/LaneSmoother.h"
#include "types/Metrics.h"
#include "types/PulseEntityType.h"
#include "types/PulseErrorCode.h"
#include "types/PulseEvents.h"
#include "types/PulseIntersectionStats.h"
#include "types/PulseLaneInfo.h"
#include "types/PulseLaneStats.h"
#include "types/PulsePosition.h"
#include "types/PulseTrafficLightStats.h"
#include "types/PulseVehicleRole.h"
#include "types/PulseVehicleType.h"
#include "types/RunState.h"
#include "types/StatsSnapshot.h"
#include "types/SumoOutput.h"
#include "types/TrafficControlMode.h"
#include "types/TrafficLightDurations.h"
#include "types/TrafficLightState.h"

// --- Utils ---
#include "utils/BFS.h"
#include "utils/File.h"
#include "utils/Log.h"
#include "utils/Metrics.h"
#include "utils/Settings.h"
#include "utils/Stats.h"
#include "utils/Sumo.h"
#include "utils/XML.h"

#include "controllers/AutoTuner.h"
#include "utils/Log.h"

#include <algorithm>
#include <numeric>

namespace TrafficPulse
{

AutoTuner::AutoTuner(const Settings::Adaptive::Tune& cfg, Settings::FlowThresh&
flow, PIDController& pid)
    : m_flow(flow)

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				113
Змн.	Арк.	№ докум.	Підпис	Дата		

```

, m_pid(pid)
, m_cfg(cfg) {
m_next_tune_time = cfg.retune_every;
}

void AutoTuner::sample(const double sim_time, const double total_flow_norm, const
std::vector<double>& queues) {
    if(!m_cfg.enabled)
        return;

    m_accumulated_flow += total_flow_norm;
    m_accumulated_load += computeLoad(total_flow_norm, queues);
    ++m_samples;

    if(sim_time >= m_next_tune_time)
        tryRetune(sim_time);
}

double AutoTuner::computeLoad(const double flow_norm, const std::vector<double>&
queue) const {
    const double avg_queue = queue.empty() ? 0.0 : std::accumulate(queue.begin(),
queue.end(), 0.0) / queue.size();

    return m_cfg.queue_w * avg_queue + (1.0 - m_cfg.queue_w) * flow_norm;
}

void AutoTuner::tryRetune(const double sim_time) {
    if(sim_time - m_last_tune_time < m_cfg.min_hold) {
        m_next_tune_time += m_cfg.retune_every;
        return;
    }

    const double mean_flow = m_accumulated_flow / std::max<size_t>(1, m_samples);
    const double mean_load = m_accumulated_load / std::max<size_t>(1, m_samples);

    m_accumulated_flow = m_accumulated_load = 0.0;
    m_samples = 0;

    m_flow.low = std::clamp(mean_flow * 0.5, 0.05, 0.5);
    m_flow.high = std::clamp(mean_flow * 1.2, 0.6, 2.0);

    const double k = std::clamp(mean_load, 0.2, 1.5);
    const double Kp = std::clamp(k * 10.0, m_cfg.lim.Kp * 0.02, m_cfg.lim.Kp);
    const double Ki = std::clamp(k * 0.01, 0.0, m_cfg.lim.Ki);
    const double Kd = std::clamp(k * 0.02, 0.0, m_cfg.lim.Kd);

    m_pid.setGains(Kp, Ki, Kd);
    m_pid.resetIntegrator();

    spdlog::info("[AutoTuner] tune @{:0f}s  load={:0.2f} ⇒ "
        "Kp={:0.2f} Ki={:0.3f} Kd={:0.3f}  low={:0.2f} high={:0.2f}",
        sim_time,
        mean_load,
        Kp,
        Ki,
        Kd,
        m_flow.low,
        m_flow.high);

    m_last_tune_time = sim_time;
    m_next_tune_time = sim_time + m_cfg.retune_every;
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				114
Змн.	Арк.	№ докум.	Підпис	Дата		

```

} // namespace TrafficPulse

#include "controllers/MetaController.h"

namespace TrafficPulse
{
MetaController::MetaController(Settings& cfg)
    : m_flow(cfg.flow)
    , m_mode_ctrl(m_flow)
    , m_baseline(std::make_unique<BaselineWaveStrategy>(cfg.algo.default_speed,
cfg.algo.max_cycle))
    , m_hybrid(
        std::make_unique<HybridWaveStrategy>(cfg.algo.inter_green,
cfg.algo.phases, cfg.algo.max_cycle, cfg.adaptive))
    ,
m_max_pressure(std::make_unique<MaxPressureWaveStrategy>(cfg.algo.max_cycle)) {
    if(cfg.adaptive.tune.enabled) {
        m_tuner.emplace(cfg.adaptive.tune, m_flow, m_hybrid->pid());
    }
}

void MetaController::update(const double sim_time, const double total_flow, const
std::vector<double>& queue_lengths) {
    if(m_tuner)
        m_tuner->sample(sim_time, total_flow, queue_lengths);

    m_mode_ctrl.evaluate(total_flow, queue_lengths, sim_time);
}

std::unordered_map<std::string, double>
MetaController::calculateOffsets(const std::vector<std::string>& intersection_ids,
                                const std::function<double(const std::string&, const
std::string&)>& get_distance,
                                const double current_time,
                                const std::vector<double>& queue_lengths,
                                const double total_flow) const {
    return activeStrategy().calculateOffsets(intersection_ids, get_distance,
current_time, queue_lengths, total_flow);
}

TrafficControlMode MetaController::getCurrentMode() const {
    return m_mode_ctrl.getCurrentMode();
}

IGreenWaveStrategy& MetaController::getActiveStrategy() {
    return activeStrategy();
}

const IGreenWaveStrategy& MetaController::getActiveStrategy() const {
    return activeStrategy();
}

IGreenWaveStrategy& MetaController::activeStrategy() {
    switch(m_mode_ctrl.getCurrentMode()) {
    case TrafficControlMode::Hybrid:
        return *m_hybrid;
    case TrafficControlMode::MaxPressure:
        return *m_max_pressure;
    case TrafficControlMode::Baseline:
    default:

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІПЗ	Арк.
		Власенко О.В.				115
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return *m_baseline;
    }
}

const IGreenWaveStrategy& MetaController::activeStrategy() const {
    return const_cast<MetaController*>(this)->activeStrategy();
}

void MetaController::forceMode(const TrafficControlMode mode) {
    m_mode_ctrl.forceMode(mode);
}

} // namespace TrafficPulse

#include "controllers/PIDController.h"
#include "utils/Log.h"
#include <algorithm>

namespace TrafficPulse
{
    PIDController::PIDController(const Settings::Adaptive& cfg)
        : m_cfg(cfg)
        , m_Kp(cfg.pid.Kp)
        , m_Ki(cfg.pid.Ki)
        , m_Kd(cfg.pid.Kd) { }

    double PIDController::update(const double measured_flow, const double dt, const
    double avg_queue) {
        if(dt <= 1e-6 || !std::isfinite(measured_flow) || !std::isfinite(avg_queue))
            return m_last_offset_delta;

        const double error = m_set_point - measured_flow + avg_queue;

        m_integral += error * dt;
        const double max_int = m_cfg.pid.cap_move / std::max(m_Ki, 1e-6);
        m_integral = std::clamp(m_integral, -max_int, max_int);

        const double derivative = (error - m_prev_error) / std::max(dt, 1e-6);

        double adj = m_Kp * error + m_Ki * m_integral + m_Kd * derivative;

        adj = std::clamp(adj, -m_cfg.pid.cap_move, m_cfg.pid.cap_move);
        if(!std::isfinite(adj))
            adj = 0.0;

        spdlog::debug("PID: err={:.3f} int={:.3f} der={:.3f} ⇒ Δ={:.3f}", error,
        m_integral, derivative, adj);

        m_prev_error = error;
        return m_last_offset_delta = adj;
    }

    void PIDController::setTarget(double sp) {
        m_set_point = sp;
        m_integral = 0.0;
        m_prev_error = 0.0;
    }

    void PIDController::setGains(const double Kp, const double Ki, const double Kd) {
        m_Kp = Kp;
        m_Ki = Ki;
    }
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				116
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        m_Kd = Kd;
    }

void PIDController::resetIntegrator() {
    m_integral = 0.0;
}

void PIDController::reset() {
    m_integral = 0.0;
    m_prev_error = 0.0;
}

} // namespace TrafficPulse

#include "controllers/TrafficModeController.h"
#include "utils/Log.h"
#include "utils/Settings.h"
#include <algorithm>

namespace TrafficPulse
{
TrafficModeController::TrafficModeController(const Settings::FlowThresh& cfg)
    : m_cfg(cfg) { }

TrafficControlMode TrafficModeController::evaluate(const double total_flow,
                                                    const std::vector<double>& queue_lengths,
                                                    const double sim_time) {

    const double dwell =
        (m_current_mode == TrafficControlMode::MaxPressure) ?
m_cfg.dwell.max_pressure : m_cfg.dwell.other;

    if(sim_time - m_last_switch_time < dwell) {
        return m_current_mode;
    }

    const bool over_any =
        std::ranges::any_of(queue_lengths, [&](const double queue) { return queue >
m_cfg.queue.per_inter; });
    const bool over_total = total_flow > m_cfg.queue.threshold;
    TrafficControlMode next;
    if(over_any || over_total) {
        next = TrafficControlMode::MaxPressure;
    }
    else {
        const double load = total_flow / std::max(m_cfg.high, 1e-3);
        next = (load > 1.0) ? TrafficControlMode::Hybrid :
TrafficControlMode::Baseline;
    }

    if(next != m_current_mode) {
        spdlog::info("Mode switch: {} -> {}", m_current_mode, next);

        m_current_mode = next;
        m_last_switch_time = sim_time;
    }

    return m_current_mode;
}

TrafficControlMode TrafficModeController::getCurrentMode() const {
    return m_current_mode;
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				117
Змн.	Арк.	№ докум.	Підпис	Дата		

```

void TrafficModeController::forceMode(const TrafficControlMode mode) {
    m_current_mode = mode;
    m_last_switch_time = std::numeric_limits<double>::lowest();
}

} // namespace TrafficPulse

#include <stdexcept>

#include "core/PulseDataManager.h"
#include "core/PulseException.h"
#include "utils/Sumo.h"

#include <ranges>

namespace TrafficPulse
{
PulseDataManager& PulseDataManager::getInstance() {
    static PulseDataManager instance;
    return instance;
}

void PulseDataManager::setEventBus(EventBus& bus) {
    m_event_bus = &bus;
}

void PulseDataManager::addIntersection(std::unique_ptr<PulseIntersection>
intersection) {
    if(!intersection) {
        throw std::invalid_argument("Cannot add a null intersection.");
    }

    const std::string& id = intersection->getId();
    if(m_intersections.contains(id)) {
        throw PulseException("Intersection with this ID already exists: " + id,
PulseErrorCode::DuplicateEntity);
    }

    m_intersections[id] = std::move(intersection);
}

PulseIntersection* PulseDataManager::getIntersection(const std::string&
intersection_id) const {
    const auto it = m_intersections.find(intersection_id);
    return (it != m_intersections.end()) ? it->second.get() : nullptr;
}

void PulseDataManager::addTrafficLight(std::unique_ptr<PulseTrafficLight>
traffic_light) {
    if(!traffic_light) {
        throw std::invalid_argument("Cannot add a null traffic light.");
    }

    const std::string& id = traffic_light->getId();
    if(m_traffic_lights.contains(id)) {
        throw PulseException("Traffic light with this ID already exists: " + id,
PulseErrorCode::DuplicateEntity);
    }

    m_traffic_lights[id] = std::move(traffic_light);
}
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				118
Змн.	Арк.	№ докум.	Підпис	Дата		

```

PulseTrafficLight* PulseDataManager::getTrafficLight(const std::string&
traffic_light_id) const {
    const auto it = m_traffic_lights.find(traffic_light_id);
    return (it != m_traffic_lights.end()) ? it->second.get() : nullptr;
}

void PulseDataManager::addVehicle(std::unique_ptr<PulseVehicle> vehicle) {
    if(!vehicle) {
        throw std::invalid_argument("Cannot add a null vehicle.");
    }

    const std::string& id = vehicle->getId();
    if(m_vehicles.contains(id)) {
        throw PulseException("Vehicle with this ID already exists: " + id,
PulseErrorCode::DuplicateEntity);
    }

    m_vehicles[id] = std::move(vehicle);
}

PulseVehicle* PulseDataManager::getVehicle(const std::string& vehicle_id) const {
    const auto it = m_vehicles.find(vehicle_id);
    return (it != m_vehicles.end()) ? it->second.get() : nullptr;
}

std::vector<PulseIntersection*> PulseDataManager::getAllIntersections() const {
    std::vector<PulseIntersection*> results;
    results.reserve(m_intersections.size());
    for(const auto& intersection : m_intersections | std::views::values) {
        results.push_back(intersection.get());
    }
    return results;
}

std::vector<PulseTrafficLight*> PulseDataManager::getAllTrafficLights() const {
    std::vector<PulseTrafficLight*> results;
    results.reserve(m_traffic_lights.size());
    for(const auto& light : m_traffic_lights | std::views::values) {
        results.push_back(light.get());
    }
    return results;
}

std::vector<PulseVehicle*> PulseDataManager::getAllVehicles() const {
    std::vector<PulseVehicle*> results;
    results.reserve(m_vehicles.size());
    for(const auto& vehicle : m_vehicles | std::views::values) {
        results.push_back(vehicle.get());
    }
    return results;
}

void PulseDataManager::setRealTrafficLights(std::unordered_set<std::string> tls) {
    m_real_tls = std::move(tls);
}

bool PulseDataManager::isRealTrafficLight(const std::string& id) const {
    return m_real_tls.contains(id);
}

std::unordered_set<std::string>& PulseDataManager::getRealTrafficLights() {

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				119
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return m_real_tls;
    }

void PulseDataManager::clearAll() {
    m_intersections.clear();
    m_traffic_lights.clear();
    m_vehicles.clear();
}

void PulseDataManager::syncFromSumo(const SumoIntegration& sumo) {
    clearAll();

    for(const auto& id : sumo.getAllTrafficLights()) {
        addTrafficLight(std::make_unique<PulseTrafficLight>(id));
    }

    setRealTrafficLights({sumo.getAllTrafficLights().begin(),
sumo.getAllTrafficLights().end()});
    updateFromSumo(sumo);
}

void PulseDataManager::updateFromSumo(const SumoIntegration& sumo) {
    const auto live_ids = sumo.getAllVehicles();
    const std::unordered_set<std::string> live(live_ids.begin(), live_ids.end());

    for(auto it = m_vehicles.begin(); it != m_vehicles.end(); ) {
        it = live.contains(it->first) ? std::next(it) : m_vehicles.erase(it);
    }

    for(const auto& id : live_ids) {
        const auto [first_position, second_position] = sumo.getVehiclePosition(id);

        if(!m_vehicles.contains(id)) {
            const auto type_str = sumo.getVehicleType(id);
            const auto type = Sumo::deduceVehicleTypeFromSumo(type_str);

            addVehicle(std::make_unique<PulseVehicle>(
                id, type, PulseVehicleRole::NORMAL, PulsePosition{first_position,
second_position}));
        }
        else {
            m_vehicles[id]->updatePosition({first_position, second_position});
        }
    }
}
} // namespace TrafficPulse

#include "core/PulseLoader.h"
#include "core/PulseDataManager.h"
#include "core/PulseException.h"
#include "entities/PulseIntersection.h"
#include "entities/PulseTrafficLight.h"
#include "events/NetworkLoadingCompletedEvent.h"
#include "events/NetworkLoadingProgressEvent.h"
#include "events/NetworkLoadingStartedEvent.h"
#include "types/PulsePosition.h"
#include "utils/XML.h"

#include <filesystem>
#include <utility>

namespace

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				120
Змн.	Арк.	№ докум.	Підпис	Дата		

```

{
int g_next_road_id = 1;
}

namespace TrafficPulse
{
PulseLoader::PulseLoader(PulseDataManager& data_manager, std::string net_file,
EventBus& bus)
: m_data_manager(data_manager)
, m_net_file_path(std::move(net_file))
, m_event_bus(bus) { }

void PulseLoader::loadNetworkData() const {
PulseDataManager::getInstance().clearAll();

m_event_bus.dispatch(std::make_shared<TypedEvent<NetworkLoadingStartedEvent>>(
NetworkLoadingStartedEvent{.message = "Starting network file load..."}));

if(!std::filesystem::exists(m_net_file_path)) {
throw PulseException("= Network file not found: " + m_net_file_path,
PulseErrorCode::InvalidFilePath);
}
spdlog::info("Loading started. Parsing network file: {}", m_net_file_path);

pugi::xml_document doc;
if(pugi::xml_parse_result result = doc.load_file(m_net_file_path.c_str());
!result) {
throw PulseException("Could not parse network file: " + m_net_file_path +
", error: " + result.description(),
PulseErrorCode::ParsingError);
}
pugi::xml_node net_node = doc.child("net");
if(!net_node) {
throw PulseException("No <net> element found in network file.",
PulseErrorCode::ParsingError);
}

int total_junctions = XML::count_children_if(net_node, "junction",
[] (pugi::xml_node) { return true; });
int total_tls = XML::count_children_if(net_node, "junction", [] (const
pugi::xml_node node) {
return std::string(node.attribute("type").as_string("")) ==
"traffic_light";
});

int total_edges = 0;
for(auto edge_node : net_node.children("edge")) {
if(std::string func = edge_node.attribute("function").as_string(""); func
!= "internal") {
total_edges++;
}
}

float items_parsed = 0.0f;
auto total_items = static_cast<float>(total_junctions + total_tls +
total_edges);

auto notify_progress = [&](const std::string& msg) {
m_event_bus.dispatch(std::make_shared<TypedEvent<NetworkLoadingProgressEvent>>(
NetworkLoadingProgressEvent{.message = msg, .progress = items_parsed /
total_items}));
}
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				121
Змн.	Арк.	№ докум.	Підпис	Дата		

```

};

bool found_any_junction = false;
for(auto jnode : net_node.children("junction")) {
    found_any_junction = true;

    std::string j_id = jnode.attribute("id").as_string();
    std::string j_type = jnode.attribute("type").as_string("");
    double x = jnode.attribute("x").as_double(0.0);
    double y = jnode.attribute("y").as_double(0.0);
    std::string tls_id;
    if(j_type == "traffic_light")
        tls_id = j_id;
    else if(jnode.attribute("tl"))
        tls_id = jnode.attribute("tl").as_string("");

    auto intersection = std::make_unique<PulseIntersection>(j_id,
PulsePosition{x, y}, tls_id);
    m_data_manager.addIntersection(std::move(intersection));

    items_parsed += 1.0f;
    notify_progress("Parsing junction: " + j_id);
}
if(!found_any_junction) {
    throw PulseException("No <junction> element found in network file.",
PulseErrorCode::ParsingError);
}

for(auto jnode : net_node.children("junction")) {
    std::string j_type = jnode.attribute("type").as_string("");
    if(j_type == "traffic_light") {
        std::string tl_id = jnode.attribute("id").as_string();
        auto tl = std::make_unique<PulseTrafficLight>(tl_id);
        m_data_manager.addTrafficLight(std::move(tl));

        items_parsed += 1.0f;
        notify_progress("Parsing traffic light: " + tl_id);
    }
}

for(auto edge_node : net_node.children("edge")) {
    std::string func = edge_node.attribute("function").as_string("");
    if(func == "internal") {
        continue;
    }

    std::string from_id = edge_node.attribute("from").as_string("");
    std::string to_id = edge_node.attribute("to").as_string("");

    PulseIntersection* from_inter = m_data_manager.getIntersection(from_id);
    PulseIntersection* to_inter = m_data_manager.getIntersection(to_id);
    if(from_inter && to_inter) {
        double distance = 100.0;
        if(pugi::xml_node lane_node = edge_node.child("lane")) {
            distance = lane_node.attribute("length").as_double(100.0);
        }

        PulseTrafficLight* tl = m_data_manager.getTrafficLight(from_id);

        int road_id = generateRoadID();
        from_inter->addRoadConnection(road_id, to_inter, tl, distance);
    }
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				122
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if(bool is_one_way = edge_node.attribute("oneway").as_bool(false);
!is_one_way) {
            int rev_road_id = generateRoadID();
            PulseTrafficLight* rev_tl = m_data_manager.getTrafficLight(to_id);

            to_inter->addRoadConnection(rev_road_id, from_inter, rev_tl,
distance);
        }
    }

    for(auto lane_node = edge_node.child("lane"); lane_node; lane_node =
lane_node.next_sibling("lane")) {
        std::string lane_id = lane_node.attribute("id").as_string("");

        if(to_inter) {
            to_inter->addInboundLane(lane_id);
        }

        if(auto* tl = m_data_manager.getTrafficLight(to_id)) {
            tl->addControlledLane(lane_id);
        }
    }

    items_parsed += 1.0f;
    notify_progress("Parsing edge from " + from_id + " to " + to_id);
}

spdlog::info("Network file parsed successfully.");

m_event_bus.dispatch(std::make_shared<TypedEvent<NetworkLoadingCompletedEvent>>(
    NetworkLoadingCompletedEvent{.message = "Network data loading
completed!"}));
}

int PulseLoader::generateRoadID() {
    return g_next_road_id++;
}
} // namespace TrafficPulse

#include "core/PulseStatsCollector.h"
#include "entities/PulseIntersection.h"
#include "entities/PulseTrafficLight.h"
#include "entities/PulseVehicle.h"
#include "events/StatsUpdatedEvent.h"

#include <algorithm>

namespace TrafficPulse
{
    PulseStatsCollector::PulseStatsCollector(PulseDataManager& data_manager,
SumoIntegration& sumo_integration,
EventBus& bus)
        : m_data_manager(data_manager)
        , m_sumo(sumo_integration)
        , m_event_bus(bus) { }

    void PulseStatsCollector::collectStepData(const double current_sim_time) {
        updateVehicleStats(current_sim_time);
        updateLaneStats(current_sim_time);
        updateIntersectionStats(current_sim_time);
        updateTrafficLightStats(current_sim_time);
    }
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				123
Змн.	Арк.	№ докум.	Підпис	Дата		

```

m_previous_time = current_sim_time;

const StatsSnapshot snap = getSnapshot();

m_event_bus.dispatch(std::make_shared<TypedEvent<StatsUpdatedEvent>>(
    StatsUpdatedEvent{.sim_time = snap.sim_time,
                      .avg_speed = snap.avg_speed,
                      .max_speed = snap.max_speed,
                      .vehicle_type_counts = snap.vehicle_type_counts}));
}

double PulseStatsCollector::getAverageSpeed() const {
    return m_avg_speed;
}

double PulseStatsCollector::getMaxSpeed() const {
    return m_max_speed;
}

std::unordered_map<PulseVehicleType, int>
PulseStatsCollector::getVehicleTypeCounts() const {
    return m_vehicle_type_counts;
}

double PulseStatsCollector::getIntersectionAvgWaitTime(const std::string&
intersection_id) const {
    const auto it = m_intersection_stats.find(intersection_id);

    if(it == m_intersection_stats.end()) {
        return 0.0;
    }

    const auto& [total_wait_time, vehicles_passed] = it->second;
    if(vehicles_passed == 0)
        return 0.0;
    return total_wait_time / static_cast<double>(vehicles_passed);
}

int PulseStatsCollector::getIntersectionVehiclesPassed(const std::string&
intersection_id) const {
    const auto it = m_intersection_stats.find(intersection_id);

    if(it == m_intersection_stats.end()) {
        return 0;
    }

    return it->second.vehicles_passed;
}

double PulseStatsCollector::getTimeSpentGreen(const std::string& traffic_light_id)
const {
    const auto it = m_light_stats.find(traffic_light_id);

    if(it == m_light_stats.end()) {
        return 0.0;
    }

    return it->second.time_green;
}

double PulseStatsCollector::getTimeSpentRed(const std::string& traffic_light_id)

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				124
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const {
    const auto it = m_light_stats.find(traffic_light_id);

    if(it == m_light_stats.end()) {
        return 0.0;
    }

    return it->second.time_red;
}

int PulseStatsCollector::getVehiclesStoppedAtRed(const std::string&
traffic_light_id) const {
    const auto it = m_light_stats.find(traffic_light_id);

    if(it == m_light_stats.end()) {
        return 0;
    }

    return it->second.vehicles_stopped;
}

int PulseStatsCollector::getLightCycleCount(const std::string& traffic_light_id)
const {
    const auto it = m_light_stats.find(traffic_light_id);

    if(it == m_light_stats.end()) {
        return 0;
    }

    return it->second.cycle_count;
}

int PulseStatsCollector::getLaneWaitingCount(const std::string& lane_id) const {
    if(!m_lane_map.contains(lane_id))
        return 0;
    return m_lane_map.at(lane_id).waiting_count;
}

PulseIntersectionStats PulseStatsCollector::getIntersectionStats(const
std::string& intersection_id) const {
    const auto it = m_intersection_stats.find(intersection_id);

    if(it == m_intersection_stats.end()) {
        return {0.0, 0};
    }

    return it->second;
}

StatsSnapshot PulseStatsCollector::getSnapshot() const {
    StatsSnapshot snap;
    snap.sim_time = m_previous_time;
    snap.avg_speed = m_avg_speed;
    snap.max_speed = m_max_speed;
    snap.vehicle_type_counts = m_vehicle_type_counts;
    snap.intersection_map = m_intersection_stats;
    snap.traffic_light_map = m_light_stats;
    return snap;
}

void PulseStatsCollector::updateVehicleStats(const double /*current_sim_time*/) {
    m_avg_speed = 0.0;

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІЗ	Арк.
		Власенко О.В.				125
Змн.	Арк.	№ докум.	Підпис	Дата		

```

m_max_speed = 0.0;
m_vehicle_type_counts.clear();

const auto vehicles = m_data_manager.getAllVehicles();
if(vehicles.empty()) {
    return;
}

double sum_speed = 0.0;
int count = 0;
for(const auto* vehicle : vehicles) {
    const double speed = vehicle->getSpeed();
    sum_speed += speed;
    if(speed > m_max_speed) {
        m_max_speed = speed;
    }
    count++;

    auto vt = vehicle->getType();
    m_vehicle_type_counts[vt]++;
}

if(count > 0) {
    m_avg_speed = sum_speed / static_cast<double>(count);
}
}

void PulseStatsCollector::updateIntersectionStats(const double current_sim_time) {
    double dt = current_sim_time - m_previous_time;
    if(dt < 0.0)
        dt = 0.0;

    for(const auto intersections = m_data_manager.getAllIntersections(); const
auto* inter : intersections) {
        auto& [total_wait_time, vehicles_passed] = m_intersection_stats[inter-
>getId()];

        double added_wait_time = 0.0;
        constexpr int added_pass_count = 0;

        for(const auto& inbound_lanes = inter->getInboundLanes(); const auto&
lane_id : inbound_lanes) {
            if(!m_lane_map.contains(lane_id)) {
                continue;
            }

            const auto& lane_stats = m_lane_map.at(lane_id);

            added_wait_time += lane_stats.waiting_count * dt;
        }

        total_wait_time += added_wait_time;
        vehicles_passed += added_pass_count;
    }
}

void PulseStatsCollector::updateTrafficLightStats(const double current_sim_time) {
    double dt = current_sim_time - m_previous_time;
    if(dt < 0.0)
        dt = 0.0;

    const auto traffic_lights = m_data_manager.getAllTrafficLights();

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				126
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    auto vehicles = m_data_manager.getAllVehicles();

    for(const auto* traffic_light : traffic_lights) {
        auto tl_id = traffic_light->getId();
        auto& [time_green, time_red, vehicles_stopped, cycle_count, last_state,
last_state_change_time] =
            m_light_stats[tl_id];
        const auto current_state = traffic_light->getState();

        if(current_state == last_state) {
            if(current_state == TrafficLightState::GREEN) {
                time_green += dt;
            }
            else if(current_state == TrafficLightState::RED) {
                time_red += dt;
            }
        }
        else {
            cycle_count++;
            double changed_dt = current_sim_time - last_state_change_time;
            if(changed_dt < 0.0)
                changed_dt = 0.0;

            if(last_state == TrafficLightState::GREEN) {
                time_green += changed_dt;
            }
            else if(last_state == TrafficLightState::RED) {
                time_red += changed_dt;
            }
            last_state = current_state;
            last_state_change_time = current_sim_time;
        }

        if(current_state == TrafficLightState::RED) {
            const auto& controlled_lanes = traffic_light->getControlledLanes();
            int total_waiting = 0;
            for(auto& lid : controlled_lanes) {
                if(m_lane_map.contains(lid)) {
                    total_waiting += m_lane_map.at(lid).waiting_count;
                }
            }
            vehicles_stopped += total_waiting;
        }
    }
}

void PulseStatsCollector::updateLaneStats(double /*current_sim_time*/) {
    if(!m_sumo.isRunning()) {
        return;
    }

    for(const auto lane_ids = m_sumo.getAllLanes(); auto& lane_id : lane_ids) {
        if(!m_lane_map.contains(lane_id)) {
            m_lane_map[lane_id] = PulseLaneStats{};
        }
        auto& [speed_smoother, occupancy_smoother, waiting_count] =
m_lane_map[lane_id];
        const double speed = m_sumo.getLaneMeanSpeed(lane_id);
        const double occupancy = m_sumo.getLaneOccupancy(lane_id);

        speed_smoother.addSample(speed);
        occupancy_smoother.addSample(occupancy);
    }
}

```

		Бабушко А.С.			ІІЗ.КР.Б.-121-25-ІІЗ	Арк.
		Власенко О.В.				127
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        int waiting = 0;
        if(speed < 0.3) {
            auto vehicles_list = m_sumo.getLaneVehicles(lane_id);
            waiting = static_cast<int>(vehicles_list.size());
        }
        waiting_count = waiting;
    }
}
} // namespace TrafficPulse

#include "core/PulseTrafficAlgo.h"
#include "entities/PulseIntersection.h"
#include "events/TrafficModeChangedEvent.h"
#include "services/GreenWaveManager.h"
#include "utils/BFS.h"
#include "utils/Log.h"
#include "utils/Stats.h"

#include <algorithm>
#include <cmath>
#include <libsumo/TrafficLight.h>
#include <ranges>
#include <unordered_set>

namespace TrafficPulse
{
    PulseTrafficAlgo::PulseTrafficAlgo(PulseDataManager& data_manager,
                                       SumoIntegration& sumo_integration,
                                       EventBus& bus,
                                       Settings& cfg)
        : m_data_manager(data_manager)
        , m_sumo_integration(sumo_integration)
        , m_event_bus(bus)
        , m_cfg(cfg) {
        m_meta = std::make_shared<MetaController>(cfg);
        m_prev_mode = m_meta->getCurrentMode();
        m_next_adapt_check = 0.0;

        auto strat_alias =
            std::shared_ptr<IGreenWaveStrategy>(m_meta,
            &const_cast<IGreenWaveStrategy>(m_meta->getActiveStrategy()));
        m_green_wave_manager = std::make_unique<GreenWaveManager>(strat_alias);
    }

    void PulseTrafficAlgo::initializeGreenWave(const double current_sim_time) {
        switch(m_cfg.algo.force_mode) {
            case TrafficControlMode::MaxPressure: {
                spdlog::info("Forcing mode to MaxPressure");
                m_meta->forceMode(TrafficControlMode::MaxPressure);
                m_green_wave_manager->setStrategy(
                    std::shared_ptr<IGreenWaveStrategy>(m_meta,
                    &const_cast<IGreenWaveStrategy>(m_meta->getActiveStrategy())));
                break;
            }
            case TrafficControlMode::Hybrid: {
                spdlog::info("Forcing mode to Hybrid");
                m_meta->forceMode(TrafficControlMode::Hybrid);
                m_green_wave_manager->setStrategy(
                    std::shared_ptr<IGreenWaveStrategy>(m_meta,
                    &const_cast<IGreenWaveStrategy>(m_meta->getActiveStrategy())));
                break;
            }
        }
    }
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				128
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
    case TrafficControlMode::Baseline: {
        spdlog::info("Forcing mode to Baseline");
        m_meta->forceMode(TrafficControlMode::Baseline);
        m_green_wave_manager->setStrategy(
            std::shared_ptr<IGreenWaveStrategy>(m_meta,
&const_cast<IGreenWaveStrategy>(m_meta->getActiveStrategy())));
        break;
    }
}

m_last_departed = m_sumo_integration.getNumDeparted();
m_last_flow_time = current_sim_time;
m_offsets.clear();
m_corridors = findMainCorridors();

    spdlog::info("Initialized green wave with: m_control_simulation_lights - {}",
m_control_simulation_lights);

    calculateGreenWaveSchedules();

    m_next_adapt_check = current_sim_time + m_green_wave_manager->getStrategy()-
>getCheckEvery();

    spdlog::info("Calculated {} offsets, cycle={} s", m_offsets.size(),
m_cfg.algo.max_cycle);
}

void PulseTrafficAlgo::runOnce(const double current_sim_time) {
    m_current_sim_time = current_sim_time;

    std::vector<double> queue_lengths;
    queue_lengths.reserve(m_data_manager.getRealTrafficLights().size());
    for(auto const& tls : m_data_manager.getRealTrafficLights()) {
        double sum_q = 0.0;
        if(const auto* inter = m_data_manager.getIntersection(tls)) {
            for(auto const& ln : inter->getInboundLanes())
                sum_q += m_sumo_integration.getLaneQueueLength(ln);
        }
        queue_lengths.push_back(sum_q);
    }

    if(!m_control_simulation_lights)
        return;

    for(auto const& tls : m_data_manager.getRealTrafficLights()) {
        const auto* inter = m_data_manager.getIntersection(tls);
        if(!inter)
            continue;

        for(auto const& lane : inter->getInboundLanes()) {
            try {
                auto vids = m_sumo_integration.getLaneVehicles(lane);
                if(vids.empty())
                    continue;
                double t_on = current_sim_time -
m_sumo_integration.getVehicleDepartTime(vids.front());
                if(t_on < 1.0 || t_on > 120.0)
                    continue;
                double dist = m_sumo_integration.getLaneLength(lane);
                double qlen = m_sumo_integration.getLaneQueueLength(lane);
                m_green_wave_manager->registerVehicleSample(dist, t_on);
            }
            catch(...) {}
        }
    }
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				129
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        m_green_wave_manager->registerQueueSample(qlen);
    }
    catch(...) { /* ignore lanes that disappear */
    }
}

applyEmergencyOverride(current_sim_time);

if(!m_last_flow_time) {
    m_last_flow_time = current_sim_time;
    m_last_departed = m_sumo_integration.getNumDeparted();
    return;
}

if(const double dt = current_sim_time - m_last_flow_time; dt < 1.0)
    return;

const double departed = m_sumo_integration.getNumDeparted();
m_current_flow = departed - m_last_departed;
m_last_departed = departed;

m_meta->update(current_sim_time, normaliseFlow(m_current_flow),
queue_lengths);

const auto new_mode = m_meta->getCurrentMode();
bool need_rebuild = false;
if(new_mode != m_prev_mode) {
    m_prev_mode = new_mode;
    need_rebuild = true;

    const auto new_alias =
        std::shared_ptr<IGreenWaveStrategy>(m_meta,
&const_cast<IGreenWaveStrategy>(m_meta->getActiveStrategy()));

    m_green_wave_manager->setStrategy(new_alias);
}

if(auto strat_ptr = m_green_wave_manager->getStrategy();
    strat_ptr->isAdaptiveEnabled() && current_sim_time + 1e-3 >=
m_next_adapt_check) {
    need_rebuild = true;
    m_next_adapt_check = current_sim_time + strat_ptr->getCheckEvery();
}

if(need_rebuild) {
    const auto strat_alias =
        std::shared_ptr<IGreenWaveStrategy>(m_meta,
&const_cast<IGreenWaveStrategy>(m_meta->getActiveStrategy()));

    m_green_wave_manager->setStrategy(strat_alias);

    calculateGreenWaveSchedules();
}

m_last_flow_time = current_sim_time;
}

std::vector<GreenWaveCorridor> PulseTrafficAlgo::findMainCorridors() const {
    std::vector<GreenWaveCorridor> corridors;
    const auto intersections = m_data_manager.getAllIntersections();
    if(intersections.empty())

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				130
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return corridors;

std::unordered_set<std::size_t> corridor_hashes;
std::unordered_set<std::string> visited;
std::vector<std::string> ids;
ids.reserve(intersections.size());
for(auto* i : intersections)
    ids.push_back(i->getId());
std::ranges::sort(ids);

for(const auto& root : ids) {
    if(visited.contains(root))
        continue;
    if(corridors.size() >= m_cfg.algo.max_corr)
        break;

    auto path = BFS::runBFS(
        root,
        [&](auto const& v) {
            std::vector<std::string> neigh;
            if(auto* pi = m_data_manager.getIntersection(v)) {
                for(auto const& [_ , rc] : pi->getConnectedRoads())
                    neigh.push_back(rc.getConnectedIntersection().getId());
            }
            return neigh;
        },
        m_cfg.algo.bfs_depth);

    for(auto& id : path)
        visited.insert(id);

    std::erase_if(path, [&](auto const& id) { return
!isMajorIntersection(m_data_manager.getIntersection(id)); });
    if(path.size() < 2)
        continue;

    auto pos0 = m_data_manager.getIntersection(root)->getPosition();
    auto far_it = std::ranges::max_element(path, [&](auto const& a, auto const&
b) {
        auto pa = m_data_manager.getIntersection(a)->getPosition();
        auto pb = m_data_manager.getIntersection(b)->getPosition();
        return Stats::euclideanDistanceBetween(pos0, pa) <
Stats::euclideanDistanceBetween(pos0, pb);
    });

    std::vector<std::string> forward{root, *far_it};
    std::vector<std::string> backward = forward;
    std::ranges::reverse(backward);

    auto hashCorr = [&](auto const& seq) {
        std::size_t h = 0;
        for(auto const& s : seq)
            h ^= std::hash<std::string>{}(s) + 0x9e3779b97f4a7c15ULL + (h << 6) +
(h >> 2);
        return h;
    };
    auto tryAdd = [&](auto const& seq) {
        if(auto h = hashCorr(seq); !corridor_hashes.count(h)) {
            corridor_hashes.insert(h);
            corridors.emplace_back(GreenWaveCorridor{seq,
estimateTargetSpeed(seq)});
            return true;
        }
    };

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				131
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
    return false;
};

if(tryAdd(forward) && corridors.size() >= m_cfg.algo.max_corr)
    break;
if(tryAdd(backward) && corridors.size() >= m_cfg.algo.max_corr)
    break;
}

return corridors;
}

void PulseTrafficAlgo::calculateGreenWaveSchedules() {
    m_offsets.clear();
    spdlog::debug("Building schedules for {} m_corridors", m_corridors.size());

    for(auto const& corridor : m_corridors) {
        std::vector<double> queue_lengths;
        queue_lengths.reserve(corridor.intersection_ids.size());
        for(auto const& id : corridor.intersection_ids) {
            double queue_sum = 0.0;
            if(const auto* inter = m_data_manager.getIntersection(id)) {
                for(auto const& ln : inter->getInboundLanes()) {
                    queue_sum += m_sumo_integration.getLaneQueueLength(ln);
                }
            }
            queue_lengths.push_back(queue_sum);
        }
        auto off = m_meta->calculateOffsets(
            corridor.intersection_ids,
            [&](auto const& f, auto const& t) {
                if(auto* a = m_data_manager.getIntersection(f))
                    for(auto const& [_ , rc] : a->getConnectedRoads())
                        if(rc.getConnectedIntersection().getId() == t)
                            return rc.getDistance();
                return
Stats::euclideanDistanceBetween(m_data_manager.getIntersection(f)->getPosition(),
                                m_data_manager.getIntersection(t)-
>getPosition());
            },
            m_current_sim_time,
            queue_lengths,
            m_current_flow);
        for(auto const& [tls, v] : off)
            m_offsets[tls] = v;
    }

    for(auto const& tls : m_data_manager.getRealTrafficLights())
        if(auto it = m_offsets.find(tls); it != m_offsets.end())
            applyOffset(tls, it->second);
}

void PulseTrafficAlgo::applyEmergencyOverride(const double current_sim_time) {
    for(const auto* inter : m_data_manager.getAllIntersections()) {
        if(!m_data_manager.isRealTrafficLight(inter->getId()))
            continue;
        bool em = false;
        auto posI = inter->getPosition();
        for(auto const& lane : inter->getInboundLanes()) {
            for(auto const& vid : m_sumo_integration.getLaneVehicles(lane)) {
                if(auto type = m_sumo_integration.getVehicleType(vid);

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				132
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        (type == "emergency" || type == "emergencyVeh") &&
        Stats::euclideanDistanceBetween(posI,

{m_sumo_integration.getVehiclePosition(vid).first,

m_sumo_integration.getVehiclePosition(vid).second}) <
        m_cfg.algo.emergency_thr) {
            em = true;
            break;
        }
    }
    if(em)
        break;
}
if(em) {
    const std::string id = inter->getId();

    if(!m_tls_restore_map.contains(id)) {
        m_tls_restore_map[id] =
{libsumo::TrafficLight::getRedYellowGreenState(id), current_sim_time + 10.0};
    }

    const std::string tmp1 =
libsumo::TrafficLight::getRedYellowGreenState(id);
    std::string allG(tmp1.size(), 'G');

    libsumo::TrafficLight::setRedYellowGreenState(id, allG);
}

for(auto it = m_tls_restore_map.begin(); it != m_tls_restore_map.end(); ) {
    if(current_sim_time >= it->second.second) {
        libsumo::TrafficLight::setRedYellowGreenState(it->first, it-
>second.first);
        it = m_tls_restore_map.erase(it);
    }
    else {
        ++it;
    }
}

double PulseTrafficAlgo::estimateTargetSpeed(const std::vector<std::string>&
intersection_ids) const {
    double num = 0, den = 0;
    for(auto const& id : intersection_ids) {
        for(const auto* inter = m_data_manager.getIntersection(id); auto const& ln
: inter->getInboundLanes()) {
            try {
                double sl = m_sumo_integration.getLaneSpeedLimit(ln);
                double L = m_sumo_integration.getLaneLength(ln);
                if(sl > 0.1) {
                    num += sl * L;
                    den += L;
                }
            }
            catch(...) {
            }
        }
    }
    return den > 0 ? (num / den) : m_cfg.algo.default_speed;
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				133
Змн.	Арк.	№ докум.	Підпис	Дата		

```

void PulseTrafficAlgo::applyOffset(const std::string& tl_id, const double
offset_sec) {
    const auto defs =
libsumo::TrafficLight::getCompleteRedYellowGreenDefinition(tl_id);

    if(defs.empty() || !std::isfinite(offset_sec))
        return;

    auto const& phases = defs.front().phases;

    if(std::ranges::any_of(phases, [](const auto& p) { return p->duration < 1.0;
}))
        return;

    double cycle = 0;
    for(const auto& phase : phases)
        cycle += phase->duration;

    if(cycle <= 0.0)
        return;

    double t = std::fmod(std::fmod(offset_sec, cycle) + cycle, cycle);
    std::size_t idx = 0;
    double left = 0.0;
    for(std::size_t i = 0; i < phases.size(); ++i) {
        const double dur = phases[i]->duration;

        if(dur < 1e-3) {
            t = std::fmod(t, cycle);
            continue;
        }

        if(t < dur || i == phases.size() - 1) {
            idx = i;
            left = (t < dur) ? (dur - t) : dur;
            break;
        }
        t -= dur;
    }

    if(left < 1.0)
        left = 1.0;

    try {
        libsumo::TrafficLight::setPhase(tl_id, idx);
        libsumo::TrafficLight::setPhaseDuration(tl_id, left);
    }
    catch(const std::exception& e) {
        spdlog::warn("TLS {}: {}", tl_id, e.what());
    }
}

bool PulseTrafficAlgo::isMajorIntersection(const PulseIntersection* inter) const {
    if(!inter)
        return false;

    if(!m_data_manager.isRealTrafficLight(inter->getTrafficLightId()))
        return false;

    size_t roads = 0;
    for(const auto& road_connection : inter->getConnectedRoads() |

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		134

```

std::views::values)
    if(road_connection.getDistance() > 1.0)
        ++roads;

    return roads >= 2;
}

void PulseTrafficAlgo::setControlSimulationLights(const bool enable) {
    m_control_simulation_lights = enable;
}

void PulseTrafficAlgo::setTrafficMode(const TrafficControlMode mode) {
    m_event_bus.dispatch(std::make_shared<TypedEvent<TrafficModeChangedEvent>>(Traffic
ModeChangedEvent{mode}));
    calculateGreenWaveSchedules();
}

double PulseTrafficAlgo::getCycleLength() const {
    return m_cfg.algo.max_cycle;
}

TrafficControlMode PulseTrafficAlgo::getMode() const {
    return m_meta->getCurrentMode();
}

double PulseTrafficAlgo::normaliseFlow(const double raw) const {
    return std::clamp(raw / std::max(1.0, m_cfg.flow.high), 0.0, 5.0);
}
} // namespace TrafficPulse

#include "core/SumoIntegration.h"
#include "constants/CMakeBinaryDir.h"
#include "core/PulseException.h"
#include "utils/File.h"
#include "utils/Log.h"

#include <algorithm>
#include <filesystem>
#include <libsumo/libsumo.h>
#include <pugixml.hpp>
#include <utility>

namespace TrafficPulse
{
SumoIntegration::SumoIntegration(const std::string& sumo_config,
                                const std::string& net_file,
                                const std::vector<SumoOutput>& extra,
                                const bool bypass_config_check,
                                const bool run_with_gui)
    : m_run_with_gui(run_with_gui)
    , m_extra(extra) {
    const auto sumo_config_path = File::resolvePath(sumo_config);
    const auto net_config_path = File::resolvePath(net_file);

    if(!bypass_config_check) {
        if(!std::filesystem::exists(sumo_config_path)) {
            throw PulseException("SUMO config file not found! Expected at: " +
sumo_config_path.string() +
                                "\nEnsure you have placed the correct file inside
'config/sumo/'.",
                                PulseErrorCode::InvalidFilePath);
        }
    }
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІЗ	Арк.
		Власенко О.В.				135
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    if(!std::filesystem::exists(net_config_path)) {
        throw PulseException("Net file does not exist! " +
net_config_path.string() +
                                "\nEnsure you have placed the correct file inside
'config/sumo/'.",
                                PulseErrorCode::InvalidFilePath);
    }

    m_sumo_config = std::move(sumo_config_path);
    m_net_config = std::move(net_config_path);
}
else {
    m_sumo_config = std::move(sumo_config_path);
    m_net_config = std::move(net_config_path);
}

spdlog::info("Using SUMO config file: {}", m_sumo_config);
}

void SumoIntegration::startSimulation() {
    if(m_started) {
        throw PulseException("SUMO simulation already running.",
PulseErrorCode::AlreadyRunning);
    }

    std::vector<std::string> args = {m_run_with_gui ? "sumo-gui" : "sumo", "-c",
m_sumo_config, "-n", m_net_config};

    for(auto& arg : m_extra) {
        args.push_back(arg.flag);
        if(arg.hasValue())
            args.push_back(arg.file);
    }

    if(std::ranges::none_of(m_extra, [](auto& o) { return o.flag == "--tripinfo-
output"; }))
        args.insert(args.end(), {"--tripinfo-output", "tripinfos.xml"});

    spdlog::info("Starting libsumo{} with config: {}", m_run_with_gui ? " (GUI)" :
"", m_sumo_config);
    libsumo::Simulation::start(args);
    m_started = true;
    spdlog::info("SUMO simulation started via libsumo.", m_sumo_config);
}

void SumoIntegration::stepSimulation() const {
    if(!m_started) {
        throw PulseException("Cannot step simulation: SUMO not running.",
PulseErrorCode::NotRunning);
    }
    libsumo::Simulation::step();
}

void SumoIntegration::stopSimulation() {
    if(!m_started) {
        throw PulseException("Cannot stop simulation: SUMO not running.",
PulseErrorCode::NotRunning);
    }
    spdlog::info("Stopping libsumo simulation...");
    libsumo::Simulation::close();
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				136
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        m_started = false;
        spdlog::info("SUMO simulation stopped.");
    }

    bool SumoIntegration::isRunning() const {
        if(!m_started)
            return false;

        return libsumo::Simulation::getMinExpectedNumber() > 0;
    }

    std::vector<std::string> SumoIntegration::getAllVehicles() const {
        if(!m_started) {
            throw PulseException("Cannot retrieve vehicles: SUMO not running.",
                PulseErrorCode::NotRunning);
        }
        return libsumo::Vehicle::getIDList();
    }

    std::pair<double, double> SumoIntegration::getVehiclePosition(const std::string&
        vehicle_id) const {
        if(!m_started) {
            throw PulseException("Cannot retrieve position: SUMO not running.",
                PulseErrorCode::NotRunning);
        }
        auto pos = libsumo::Vehicle::getPosition(vehicle_id);
        return {pos.x, pos.y};
    }

    std::vector<std::string> SumoIntegration::getAllTrafficLights() const {
        if(!m_started) {
            throw PulseException("Cannot retrieve traffic lights: SUMO not running.",
                PulseErrorCode::NotRunning);
        }
        return libsumo::TrafficLight::getIDList();
    }

    std::string SumoIntegration::getTrafficLightState(const std::string& tl_id) const
    {
        if(!m_started) {
            throw PulseException("Cannot retrieve traffic light state: SUMO not
                running.", PulseErrorCode::NotRunning);
        }
        return libsumo::TrafficLight::getRedYellowGreenState(tl_id);
    }

    void SumoIntegration::setTrafficLightState(const std::string& tl_id, const
        std::string& state) const {
        if(!m_started) {
            throw PulseException("Cannot set traffic light state: SUMO not running.",
                PulseErrorCode::NotRunning);
        }
        libsumo::TrafficLight::setRedYellowGreenState(tl_id, state);
    }

    std::string SumoIntegration::getVehicleType(const std::string& vehicle_id) const {
        if(!m_started) {
            throw PulseException("Cannot get vehicle type, SUMO not running.",
                PulseErrorCode::NotRunning);
        }

        try {

```

		Бабушко А.С.			ІІЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				137
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        // This returns e.g. "passenger", "truck", "bicycle", "emergency", etc.
        return libsumo::Vehicle::getTypeID(vehicle_id);
    }
    catch(const std::exception& ex) {
        throw PulseException(std::string("SumoIntegration::getVehicleType error: ")
+ ex.what(),
                                PulseErrorCode::ParsingError);
    }
}

void SumoIntegration::loadLaneData() {
    m_lane_ids.clear();

    pugi::xml_document doc;
    if(const pugi::xml_parse_result res = doc.load_file(m_net_config.c_str());
!res) {
        throw PulseException("Failed to parse net file: " + m_net_config + " - " +
res.description(),
                                PulseErrorCode::ParsingError);
    }

    const pugi::xml_node netRoot = doc.child("net");
    if(!netRoot) {
        throw PulseException("Missing <net> root in net file: " + m_net_config,
PulseErrorCode::ParsingError);
    }

    for(pugi::xml_node edgeNode = netRoot.child("edge"); edgeNode; edgeNode =
edgeNode.next_sibling("edge")) {
        for(pugi::xml_node laneNode = edgeNode.child("lane"); laneNode; laneNode =
laneNode.next_sibling("lane")) {
            if(const char* lid = laneNode.attribute("id").as_string(nullptr)) {
                m_lane_ids.emplace_back(lid);
            }
        }
    }

    spdlog::info("Loaded {} lane IDs from net file: {}", m_lane_ids.size(),
m_net_config);
}

std::vector<std::string> SumoIntegration::getAllLanes() const {
    if(!m_started) {
        throw PulseException("Cannot getAllLanes: SUMO not running.",
PulseErrorCode::NotRunning);
    }
    return m_lane_ids;
}

std::vector<std::string> SumoIntegration::getLaneVehicles(const std::string&
lane_id) const {
    if(!m_started) {
        throw PulseException("Cannot getLaneVehicles: SUMO not running.",
PulseErrorCode::NotRunning);
    }

    return libsumo::Lane::getLastStepVehicleIDs(lane_id);
}

double SumoIntegration::getLaneMeanSpeed(const std::string& lane_id) const {
    if(!m_started) {
        throw PulseException("Cannot getLaneMeanSpeed: SUMO not running.",

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				138
Змн.	Арк.	№ докум.	Підпис	Дата		

```

PulseErrorCode::NotRunning);
    }

    return libsumo::Lane::getLastStepMeanSpeed(lane_id);
}

double SumoIntegration::getLaneOccupancy(const std::string& lane_id) const {
    if(!m_started) {
        throw PulseException("Cannot getLaneOccupancy: SUMO not running.",
PulseErrorCode::NotRunning);
    }

    return libsumo::Lane::getLastStepOccupancy(lane_id);
}

double SumoIntegration::getLaneSpeedLimit(const std::string& lane_id) const {
    if(!m_started) {
        throw PulseException("Cannot getLaneSpeedLimit: SUMO not running.",
PulseErrorCode::NotRunning);
    }

    return libsumo::Lane::getMaxSpeed(lane_id);
}

double SumoIntegration::getLaneLength(const std::string& lane_id) const {
    if(!m_started) {
        throw PulseException("Cannot get lane length: SUMO not running.",
PulseErrorCode::NotRunning);
    }

    return libsumo::Lane::getLength(lane_id);
}

const std::string& SumoIntegration::getNetPath() const {
    return m_net_config;
}

const std::string& SumoIntegration::getConfigPath() const {
    return m_sumo_config;
}

double SumoIntegration::getVehicleDepartTime(const std::string& vehicle_id) const
{
    if(!m_started) {
        throw PulseException("Cannot get vehicle departure time: SUMO not
running.", PulseErrorCode::NotRunning);
    }

    return libsumo::Vehicle::getDeparture(vehicle_id);
}

double SumoIntegration::getLaneQueueLength(const std::string& lane_id) const {
    if(!m_started) {
        throw PulseException("Cannot get lane queue length: SUMO not running.",
PulseErrorCode::NotRunning);
    }

    return libsumo::Lane::getLastStepVehicleNumber(lane_id);
}

double SumoIntegration::getNumDeparted() const {
    if(!m_started) {

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				139
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        throw PulseException("Cannot get number of departed vehicles: SUMO not
running.", PulseErrorCode::NotRunning);
    }

    return libsumo::Simulation::getDepartedNumber();
}

double SumoIntegration::getDeltaT() {
    return libsumo::Simulation::getDeltaT();
}

double SumoIntegration::getTime() {
    return libsumo::Simulation::getTime();
}

} // namespace TrafficPulse

#include "core/TrafficSystem.h"
#include "core/PulseDataManager.h"
#include "core/PulseLoader.h"
#include "core/PulseStatsCollector.h"
#include "core/PulseTrafficAlgo.h"
#include "core/SumoIntegration.h"
#include "events/SimulationEndedEvent.h"
#include "events/SimulationStartedEvent.h"
#include "events/SimulationStepEvent.h"
#include "utils/Log.h"

#include <chrono>
#include <libsumo/TrafficLight.h>
#include <unordered_set>

namespace TrafficPulse
{
TrafficSystem::TrafficSystem(const double update_frequency,
                             const std::string& sumo_config_path,
                             const std::string& sumo_net_path,
                             EventBus& bus,
                             const bool algo_enabled,
                             const bool run_with_gui,
                             const std::vector<SumoOutput>& extra)
    : m_cfg(Settings::load("config/tp_config.yaml"))
    , m_data_manager(PulseDataManager::getInstance())
    , m_event_bus(bus)
    , m_sumo(std::make_unique<SumoIntegration>(sumo_config_path, sumo_net_path,
extra, false, run_with_gui))
    , m_algo(std::make_unique<PulseTrafficAlgo>(m_data_manager, *m_sumo,
m_event_bus, m_cfg))
    , m_update_frequency(update_frequency)
    , m_algo_enabled(algo_enabled) {
    m_data_manager.setEventBus(m_event_bus);

    m_algo->setControlSimulationLights(algo_enabled);

    m_stats_collector = std::make_unique<PulseStatsCollector>(m_data_manager,
*m_sumo, m_event_bus);
    m_loader = std::make_unique<PulseLoader>(m_data_manager, m_sumo->getNetPath(),
m_event_bus);
}

TrafficSystem::~TrafficSystem() {
    if(m_sumo->isRunning()) {

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				140
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        m_sumo->stopSimulation();
    }
}

void TrafficSystem::loadStaticData() const {
    m_loader->loadNetworkData();
}

PulseDataManager& TrafficSystem::getDataManager() const {
    return m_data_manager;
}

EventBus& TrafficSystem::getEventBus() const {
    return m_event_bus;
}

PulseTrafficAlgo& TrafficSystem::getAlgo() const {
    return *m_algo;
}

void TrafficSystem::run() {
    try {
        m_sumo->startSimulation();

        {
            std::vector<std::string> ids = m_sumo->getAllTrafficLights();
            m_data_manager.setRealTrafficLights(std::unordered_set(ids.begin(),
ids.end()));
        }

m_event_bus.dispatch(std::make_shared<TypedEvent<SimulationStartedEvent>>(Simulati
onStartedEvent{}));

        m_loader->loadNetworkData();

        if(m_algo_enabled) {
            m_algo->setControlSimulationLights(true);
            const double now = SumoIntegration::getTime();

            for(auto const& tls_id : m_data_manager.getRealTrafficLights()) {
                libsumo::TrafficLight::setProgram(tls_id, "0");
            }

            m_algo->initializeGreenWave(now);
        }

        while(m_run_state.load() != RunState::Stopped && m_sumo->isRunning()) {
            if(m_run_state.load() == RunState::Paused) {
                continue;
            }

            m_sumo->stepSimulation();

            m_data_manager.updateFromSumo(*m_sumo);

            const double now = SumoIntegration::getTime();
            m_algo->runOnce(now);
            m_stats_collector->collectStepData(now);

            m_event_bus.dispatch(

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		141

```

std::make_shared<TypedEvent<SimulationStepEvent>>(SimulationStepEvent{.sim_time =
now}));
    }

    m_sumo->stopSimulation();
    requestStop();

m_event_bus.dispatch(std::make_shared<TypedEvent<SimulationEndedEvent>>(Simulation
EndedEvent{}));
    }
    catch(const std::exception& ex) {

m_event_bus.dispatch(std::make_shared<TypedEvent<SimulationEndedEvent>>(Simulation
EndedEvent{}));
        throw;
    }
}

void TrafficSystem::pause() {
    m_run_state.store(RunState::Paused);
}

void TrafficSystem::resume() {
    m_run_state.store(RunState::Running);
}

bool TrafficSystem::isPaused() const {
    return m_run_state.load() == RunState::Paused;
}

bool TrafficSystem::isRunning() const {
    return m_run_state.load() == RunState::Running;
}

void TrafficSystem::requestStop() {
    m_run_state.store(RunState::Stopped);
}

} // namespace TrafficPulse

#include "entities/PulseIntersection.h"
#include "core/PulseException.h"
#include "types/PulseErrorCode.h"

#include <stdexcept>
#include <utility>

namespace TrafficPulse
{
PulseIntersection::PulseIntersection(std::string intersection_id, const
PulsePosition& position, std::string tls_id)
    : m_intersection_id(std::move(intersection_id))
    , m_position(position)
    , m_tls_id(std::move(tls_id)) { }

std::string PulseIntersection::getId() const {
    return m_intersection_id;
}

PulsePosition PulseIntersection::getPosition() const {
    return m_position;
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				142
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}

const std::string& PulseIntersection::getTrafficLightId() const {
    if(!m_tls_id.empty())
        return m_tls_id;

    return m_intersection_id;
}

void PulseIntersection::addInboundLane(const std::string& lane_id) {
    m_inbound_lanes.push_back(lane_id);
}

const std::vector<std::string>& PulseIntersection::getInboundLanes() const {
    return m_inbound_lanes;
}

void PulseIntersection::addRoadConnection(int road_id,
                                           PulseIntersection* intersection,
                                           PulseTrafficLight* traffic_light,
                                           const double distance) {
    if(!intersection) {
        throw std::invalid_argument("Invalid intersection pointer.");
    }

    if(m_connected_roads.contains(road_id)) {
        throw PulseException("Road connection with this ID already exists: " +
std::to_string(road_id),
                           PulseErrorCode::DuplicateEntity);
    }

    m_connected_roads.emplace(road_id, PulseRoadConnection(*intersection,
traffic_light, distance));
}

const std::unordered_map<int, PulseRoadConnection>&
PulseIntersection::getConnectedRoads() const {
    return m_connected_roads;
}
} // namespace TrafficPulse

#include "entities/PulseRoadConnection.h"

namespace TrafficPulse
{
PulseRoadConnection::PulseRoadConnection(PulseIntersection& intersection,
                                           PulseTrafficLight* traffic_light,
                                           const double distance)
    : m_connected_intersection(intersection)
    , m_traffic_light(traffic_light)
    , m_distance(distance) { }

PulseIntersection& PulseRoadConnection::getConnectedIntersection() const {
    return m_connected_intersection;
}

PulseTrafficLight* PulseRoadConnection::getTrafficLight() const {
    return m_traffic_light;
}

double PulseRoadConnection::getDistance() const {
    return m_distance;
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				143
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}

void PulseRoadConnection::setTrafficLight(PulseTrafficLight* traffic_light) {
    m_traffic_light = traffic_light;
}

void PulseRoadConnection::setDistance(double distance) {
    m_distance = distance;
}
} // namespace TrafficPulse

#include "entities/PulseTrafficLight.h"
#include <utility>

namespace TrafficPulse
{
PulseTrafficLight::PulseTrafficLight(std::string traffic_light_id, const
TrafficLightDurations& durations)
    : m_traffic_light_id(std::move(traffic_light_id))
    , m_current_state(TrafficLightState::RED)
    , m_durations(durations) { }

std::string PulseTrafficLight::getId() const {
    return m_traffic_light_id;
}

void PulseTrafficLight::setState(const TrafficLightState state) {
    m_current_state = state;
}

TrafficLightState PulseTrafficLight::getState() const {
    return m_current_state;
}

void PulseTrafficLight::setDurations(const TrafficLightDurations& durations) {
    m_durations = durations;
}

TrafficLightDurations PulseTrafficLight::getDurations() const {
    return m_durations;
}

void PulseTrafficLight::addControlledLane(const std::string& lane_id) {
    m_controlled_lanes.push_back(lane_id);
}

const std::vector<std::string>& PulseTrafficLight::getControlledLanes() const {
    return m_controlled_lanes;
}
} // namespace TrafficPulse

#include "entities/PulseVehicle.h"
#include <utility>

namespace TrafficPulse
{
PulseVehicle::PulseVehicle(std::string vehicle_id,
                           const PulseVehicleType type,
                           const PulseVehicleRole role,
                           const PulsePosition& position)
    : m_vehicle_id(std::move(vehicle_id))

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				144
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        , m_type(type)
        , m_role(role)
        , m_position(position) { }

std::string PulseVehicle::getId() const {
    return m_vehicle_id;
}

PulseVehicleType PulseVehicle::getType() const {
    return m_type;
}

PulseVehicleRole PulseVehicle::getRole() const {
    return m_role;
}

PulsePosition PulseVehicle::getPosition() const {
    return m_position;
}

void PulseVehicle::updatePosition(const PulsePosition& new_position) {
    m_position = new_position;
}

double PulseVehicle::getSpeed() const {
    return m_speed;
}

void PulseVehicle::setSpeed(double speedVal) {
    m_speed = speedVal;
}
} // namespace TrafficPulse

#include "services/GreenWaveManager.h"

#include <numeric>
#include <ranges>

namespace TrafficPulse
{
GreenWaveManager::GreenWaveManager(std::shared_ptr<IGreenWaveStrategy> strategy)
    : m_strategy(std::move(strategy)) { }

void GreenWaveManager::setStrategy(std::shared_ptr<IGreenWaveStrategy> strategy) {
    m_strategy = std::move(strategy);
}

std::unordered_map<std::string, double>
GreenWaveManager::buildOffsets(const std::vector<std::string>& intersection_ids,
                               const std::function<double(const std::string&, const
                               std::string&)>& get_distance,
                               const double current_time,
                               const std::vector<double>& queue_lengths,
                               const double total_flow) const {
    // 1) detect maximum distance between consecutive intersections
    m_last_max_dist = 0.0;
    for(size_t i = 1; i < intersection_ids.size(); ++i)
        m_last_max_dist = std::max(m_last_max_dist, get_distance(intersection_ids[i]
- 1], intersection_ids[i]));

    // 2) delegate to strategy -> raw metre offsets
    auto offsets =

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				145
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        m_strategy->calculateOffsets(intersection_ids, get_distance, current_time,
queue_lengths, total_flow);

        // 3) convert metres -> seconds using observed_speed or implied speed
        if(const double speed = (m_observed_speed > 0.1) ? m_observed_speed :
(m_last_max_dist / m_last_cycle);
            speed > 0.1) {
            for(auto& off : offsets | std::views::values) {
                off = off / speed;
            }
        }

        m_last_cycle = std::max(1.0, m_strategy->getCycleLength());
        m_observed_speed = (m_last_max_dist > 0.1 ? m_last_max_dist / m_last_cycle :
0.0);

        return offsets;
    }

void GreenWaveManager::registerVehicleSample(const double distance, const double
travel_time) const {
    if(travel_time <= 0.1)
        return;
    const double vehicle_speed = distance / travel_time;

    constexpr double alpha = 0.1;
    m_observed_speed =
        (m_observed_speed <= 0.0) ? vehicle_speed : alpha * vehicle_speed + (1.0 -
alpha) * m_observed_speed;
}

void GreenWaveManager::registerQueueSample(const double queue_length) {
    m_q_samples.push_back(queue_length);

    if(m_q_samples.size() > 300)
        m_q_samples.pop_front();
}

double GreenWaveManager::getAvgQueue() const {
    if(m_q_samples.empty())
        return 0.0;

    const double sum = std::accumulate(m_q_samples.begin(), m_q_samples.end(),
0.0);
    return sum / static_cast<double>(m_q_samples.size());
}

double GreenWaveManager::getObservedSpeed() const {
    return m_observed_speed;
}

double GreenWaveManager::getLastMaxDist() const {
    return m_last_max_dist;
}

} // namespace TrafficPulse

#include "strategies/BaselineWaveStrategy.h"

namespace TrafficPulse
{

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				146
Змн.	Арк.	№ докум.	Підпис	Дата		

```

BaselineWaveStrategy::BaselineWaveStrategy(const double target_speed_m_s, const
double cycle_length)
    : m_speed(target_speed_m_s)
    , m_cycle(cycle_length) { }

double BaselineWaveStrategy::getCycleLength() const {
    return m_cycle;
}

std::unordered_map<std::string, double> BaselineWaveStrategy::calculateOffsets(
    // Offset equals cumulative distance; later converted to time by caller.

    const std::vector<std::string>& intersection_ids,
    const std::function<double(const std::string&, const std::string&)>&
get_distance,
    const double current_time,
    const std::vector<double>& queue_lengths,
    const double total_flow) const {
    std::unordered_map<std::string, double> offs;

    double acc = 0.0;
    offs[intersection_ids.front()] = 0.0;

    for(size_t i = 1; i < intersection_ids.size(); ++i) {
        acc += get_distance(intersection_ids[i - 1], intersection_ids[i]);
        offs[intersection_ids[i]] = acc;
    }
    return offs;
}

} // namespace TrafficPulse

#include "strategies/HybridWaveStrategy.h"
#include "utils/Log.h"

#include <algorithm>
#include <cmath>
#include <numeric>

namespace TrafficPulse
{
    HybridWaveStrategy::HybridWaveStrategy(const double inter_green,
                                            const int phases,
                                            const double max_cycle,
                                            const Settings::Adaptive& cfg)
        : m_inter_green(inter_green)
        , m_phases(phases)
        , m_max_cycle(max_cycle)
        , m_pid(cfg)
        , m_enabled(cfg.enabled)
        , m_check_every(cfg.check_every)
        , m_min_hold(cfg.pid.min_hold)
        , m_cap_move(cfg.pid.cap_move)
        , m_avg_queue_weight(cfg.pid.avg_q_w)
        , m_warmup(cfg.warmup) {
        m_last_cycle = std::clamp(cfg.pid.init_set_pt * m_max_cycle, 30.0,
m_max_cycle);
        m_pid.setTarget(m_last_cycle);
    }

    std::unordered_map<std::string, double>

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				147
Змн.	Арк.	№ докум.	Підпис	Дата		

```

HybridWaveStrategy::calculateOffsets(const std::vector<std::string>&
intersection_ids,
                                const std::function<double(const std::string&, const
std::string&)>& get_distance,
                                const double current_time,
                                const std::vector<double>& queue_lengths,
                                const double total_flow) const {
    // computeWebsterCycle():  $C = (1.5 * L + 5) / (1 - \sum(y))$ 
    // PID adjusts cycle +/- cap_move using average queue as bias.

    // Step 1: Webster baseline
    const double sum_y = computeSumY(total_flow, intersection_ids.size(),
queue_lengths);
    const double c_web = computeWebsterCycle(sum_y);

    // Step 2: PID adaptation of cycle
    double c_target = c_web;
    if(m_enabled && current_time >= m_warmup && (current_time - m_last_adapt_time)
>= m_min_hold) {
        double avg_queue = 0.0;
        for(const double queue : queue_lengths)
            avg_queue += queue;
        avg_queue = queue_lengths.empty() ? 0.0 : avg_queue / queue_lengths.size();

        const double dt = current_time - m_last_adapt_time;
        const double weighted_queue = avg_queue * m_avg_queue_weight;
        const double delta_pid = m_pid.update(c_web, dt, weighted_queue);
        c_target = std::clamp(m_last_cycle + delta_pid, 30.0, m_max_cycle);
        m_last_adapt_time = current_time;
    }

    if(!std::isfinite(c_target)) {
        c_target = m_last_cycle;
    }

    // Step 3: Clamp delta and update cycle
    double delta = c_target - m_last_cycle;
    delta = std::clamp(delta, -m_cap_move, m_cap_move);
    m_last_cycle += delta;
    m_pid.setTarget(m_last_cycle);

    // Step 4: Build metre offsets and convert to phase units
    std::unordered_map<std::string, double> offs;
    double acc = 0.0;
    offs[intersection_ids.front()] = 0.0;
    for(size_t i = 1; i < intersection_ids.size(); ++i) {
        acc += get_distance(intersection_ids[i - 1], intersection_ids[i]);
        offs[intersection_ids[i]] = acc / (m_last_cycle / m_phases);
    }

    spdlog::debug("Cycle now {} s (Webster={}, PID Δ={})", m_last_cycle, c_web,
delta);

    return offs;
}

double HybridWaveStrategy::computeSumY(const double flow_rate,
                                const size_t N,
                                const std::vector<double>& queue_lengths) const {
    // computeSumY():  $y_{flow} + y_{queue} blend$ 

    constexpr double s_i = 0.5;

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				148
Змн.	Арк.	№ докум.	Підпис	Дата		

```

constexpr double q_i = 30.0;
const double v_i = flow_rate / std::max<size_t>(1, N);
const double y_flow = std::clamp(N * (v_i / s_i), 0.0, 0.9);
const double y_queue = std::clamp(
    std::accumulate(
        queue_lengths.begin(), queue_lengths.end(), 0.0, [&](double acc, double
q) { return acc + (q / q_i); }) /
    static_cast<double>(queue_lengths.size()),
    0.0,
    0.9);

return (1 - m_avg_queue_weight) * y_flow + m_avg_queue_weight * y_queue;
}

double HybridWaveStrategy::computeWebsterCycle(const double sum_y) const {
    // computeWebsterCycle(): classic formula with clamp

    const double L = m_phases * m_inter_green;
    const double den = std::max(0.1, 1.0 - std::clamp(sum_y, 0.0, 0.99));
    return std::clamp((1.5 * L + 5.0) / den, 30.0, m_max_cycle);
}

PIDController& HybridWaveStrategy::pid() {
    return m_pid;
}

const PIDController& HybridWaveStrategy::pid() const {
    return m_pid;
}

} // namespace TrafficPulse

#include "strategies/MaxPressureWaveStrategy.h"

#include <numeric>
#include <unordered_map>

namespace TrafficPulse
{
    MaxPressureWaveStrategy::MaxPressureWaveStrategy(const double cycle_length)
        : m_cycle(cycle_length) {}

    double MaxPressureWaveStrategy::getCycleLength() const {
        return m_cycle;
    }

    std::unordered_map<std::string, double> MaxPressureWaveStrategy::calculateOffsets(
        const std::vector<std::string>& intersection_ids,
        const std::function<double(const std::string&, const std::string&)>&
get_distance,
        double /*current_time*/,
        const std::vector<double>& queue_lengths,
        double /*total_flow*/) const {
        // Offset (1 - queue_i / sum(queue)) * cycle -> longer queue -> earlier green
        meaning the most-loaded node gets offset ≈ 0.

        const size_t N = intersection_ids.size();

        std::vector<size_t> idx(N);
        std::iota(idx.begin(), idx.end(), 0);

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				149
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    const double sum_q = std::accumulate(queue_lengths.begin(),
queue_lengths.end(), 0.0);
    std::unordered_map<std::string, double> offs;
    for(size_t rank = 0; rank < N; ++rank) {
        const size_t i = idx[rank];
        const double frac = (sum_q > 0.0) ? (1.0 - queue_lengths[i] / sum_q) : 1.0;
        offs[intersection_ids[i]] = frac * m_cycle;
    }

    return offs;
}

} // namespace TrafficPulse

#include "utils/Log.h"

namespace TrafficPulse
{
void Log::init(std::string_view level, bool to_file) {
    static bool once = ([level, to_file] {
        const auto console =
std::make_shared<spdlog::sinks::stdout_color_sink_mt>();
        console->set_pattern("[%H:%M:%S.%e] [%^%l%$] %v");

        std::vector<spdlog::sink_ptr> sinks{console};
        if(to_file) {
            const auto file =
std::make_shared<spdlog::sinks::rotating_file_sink_mt>("logs/traffic-
pulse.log", 5 * 1024 * 1024, 3);
            file->set_pattern("[%Y-%m-%d %H:%M:%S] [%l] %v");
            sinks.push_back(file);
        }

        const auto logger = std::make_shared<spdlog::logger>("tp", sinks.begin(),
sinks.end());
        spdlog::register_logger(logger);
        spdlog::set_default_logger(logger);

        const char* env = std::getenv("TP_LOG");
        const auto lvl = spdlog::level::from_str(env ? env : std::string(level));
        spdlog::set_level(lvl);
        spdlog::flush_on(spdlog::level::err);

        return true;
    })();
    (void)once;
}

} // namespace TrafficPulse

#include "utils/Metrics.h"
#include "types/Metrics.h"

#include <fstream>
#include <iomanip>
#include <regex>

namespace TrafficPulse
{
static double asDouble(const std::vector<std::string>& v, const size_t i, double

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІЗ	Арк.
		Власенко О.В.				150
Змн.	Арк.	№ докум.	Підпис	Дата		

```

def = 0) {
    return (i < v.size() && !v[i].empty()) ? std::stod(v[i]) : def;
}

static int asInt(const std::vector<std::string>& v, const size_t i, int def = 0) {
    return (i < v.size() && !v[i].empty()) ? std::stoi(v[i]) : def;
}

MetricsMap MetricsUtils::readCsv(const std::string& csv_path) {
    std::ifstream fi(csv_path);
    if(!fi)
        throw std::runtime_error("Cannot open " + csv_path);

    std::string line;
    std::getline(fi, line);
    MetricsMap out;

    while(std::getline(fi, line)) {
        auto p = split(line, ',');
        if(p.size() < 9)
            continue;

        bool is_custom = (p[2] == "custom");
        Metrics m{};
        m.avg_travel_time = asDouble(p, 3);
        m.avg_waiting_time = asDouble(p, 4);
        m.avg_speed = asDouble(p, 5);
        m.throughput = asInt(p, 6);
        m.num_stops = asInt(p, 7);
        m.avg_queue_length = asDouble(p, 8);

        m.time_loss = asDouble(p, 9);
        m.halted_peak = asDouble(p, 10);
        m.density_peak = asDouble(p, 11);
        m.queue_peak = asDouble(p, 12);

        if(auto& slot = is_custom ? out[canonical(p[1])].custom :
out[canonical(p[1])].base;
            slot.throughput == 0 || m.avg_travel_time < slot.avg_travel_time)
            slot = m;
    }
    return out;
}

void MetricsUtils::printTable(const MetricsMap& mm, std::ostream& os) {
    auto cell = [&](double v) { os << std::setw(9) << std::fixed <<
std::setprecision(2) << v; };
    os << "\nScenario          | base | custom | Δ% travel\n"
    << "-----+-----+-----+-----\n";
    for(auto& [sc, p] : mm) {
        if(!p.base.throughput || !p.custom.throughput)
            continue;
        const double delta = (p.base.avg_travel_time - p.custom.avg_travel_time) /
p.base.avg_travel_time * 100;
        os << std::left << std::setw(25) << sc << "|";
        cell(p.base.avg_travel_time);
        os << " |";
        cell(p.custom.avg_travel_time);
        os << " |";
        cell(delta);
        os << " %\n";
    }
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				151
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    os << std::endl;
}
} // namespace TrafficPulse

#include "utils/Settings.h"

#include <spdlog/spdlog.h>

namespace TrafficPulse
{

static void read_if(YAML::Node n, auto& ref) {
    if(n)
        ref = n.as<std::decay_t<decltype(ref)>>(ref);
}

Settings Settings::load(const std::string& path) {
    Settings s;
    YAML::Node root = YAML::LoadFile(path);

    auto A = root["algo"];
    read_if(A["max_cycle"], s.algo.max_cycle);
    read_if(A["max_corridors"], s.algo.max_corr);
    read_if(A["emergency_threshold"], s.algo.emergency_thr);
    read_if(A["bfs_depth"], s.algo.bfs_depth);
    read_if(A["default_target_speed"], s.algo.default_speed);
    read_if(A["inter_green"], s.algo.inter_green);
    read_if(A["phases"], s.algo.phases);
    read_if(A["force_mode"], s.algo.force_mode);

    auto F = root["flow"];
    read_if(F["low"], s.flow.low);
    read_if(F["high"], s.flow.high);
    read_if(F["min_mode_hold"], s.flow.min_hold);
    read_if(F["queue"]["threshold"], s.flow.queue.threshold);
    read_if(F["queue"]["per_inter"], s.flow.queue.per_inter);
    read_if(F["dwell"]["max_pressure"], s.flow.dwell.max_pressure);
    read_if(F["dwell"]["other"], s.flow.dwell.other);

    auto AD = root["adaptive"];
    read_if(AD["enabled"], s.adaptive.enabled);
    read_if(AD["check_every"], s.adaptive.check_every);
    read_if(AD["warmup"], s.adaptive.warmup);

    auto P = AD["pid"];
    read_if(P["Kp"], s.adaptive.pid.Kp);
    read_if(P["Ki"], s.adaptive.pid.Ki);
    read_if(P["Kd"], s.adaptive.pid.Kd);
    read_if(P["cap_move"], s.adaptive.pid.cap_move);
    read_if(P["min_hold"], s.adaptive.pid.min_hold);
    read_if(P["avg_queue_w"], s.adaptive.pid.avg_q_w);
    read_if(P["init_set_pt"], s.adaptive.pid.init_set_pt);

    auto T = AD["tune"];
    read_if(T["enabled"], s.adaptive.tune.enabled);
    read_if(T["retune_every_s"], s.adaptive.tune.retune_every);
    read_if(T["min_hold_s"], s.adaptive.tune.min_hold);
    read_if(T["queue_weight"], s.adaptive.tune.queue_w);
    read_if(T["pid_limits"]["Kp_max"], s.adaptive.tune.lim.Kp);
    read_if(T["pid_limits"]["Ki_max"], s.adaptive.tune.lim.Ki);
    read_if(T["pid_limits"]["Kd_max"], s.adaptive.tune.lim.Kd);

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				152
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        spdlog::debug("Settings loaded from {}", path);

        spdlog::debug("[algo] max_cycle={}, max_corridors={}, emergency_threshold={},
bfs_depth={}, "
            "default_target_speed={}, inter_green={}, phases=[{}],
force_mode={},
            s.algo.max_cycle,
            s.algo.max_corr,
            s.algo.emergency_thr,
            s.algo.bfs_depth,
            s.algo.default_speed,
            s.algo.inter_green,
            s.algo.phases,
            static_cast<int>(s.algo.force_mode));

        spdlog::debug("[flow] low={}, high={}, min_mode_hold={}", s.flow.low,
s.flow.high, s.flow.min_hold);

        spdlog::debug("    L queue.threshold={}, queue.per_inter=",
s.flow.queue.threshold, s.flow.queue.per_inter);

        spdlog::debug("    L dwell.max_pressure={}, dwell.other={}",
s.flow.dwell.max_pressure, s.flow.dwell.other);

        spdlog::debug("[adaptive] enabled={}, check_every={}, warmup=",
            s.adaptive.enabled,
            s.adaptive.check_every,
            s.adaptive.warmup);

        spdlog::debug("    L pid: Kp={}, Ki={}, Kd={}, cap_move={}, min_hold={},
avg_queue_w={}, init_set_pt=",
            s.adaptive.pid.Kp,
            s.adaptive.pid.Ki,
            s.adaptive.pid.Kd,
            s.adaptive.pid.cap_move,
            s.adaptive.pid.min_hold,
            s.adaptive.pid.avg_q_w,
            s.adaptive.pid.init_set_pt);

        spdlog::debug("    L tune: enabled={}, retune_every={}, min_hold={},
queue_weight=",
            s.adaptive.tune.enabled,
            s.adaptive.tune.retune_every,
            s.adaptive.tune.min_hold,
            s.adaptive.tune.queue_w);

        spdlog::debug("            limits Kp_max={}, Ki_max={}, Kd_max=",
            s.adaptive.tune.lim.Kp,
            s.adaptive.tune.lim.Ki,
            s.adaptive.tune.lim.Kd);

        return s;
    }
} // namespace TrafficPulse

#include "utils/XML.h"
#include <filesystem>
#include <spdlog/spdlog.h>

using namespace TrafficPulse;

/* internal helper */

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				153
Змн.	Арк.	№ докум.	Підпис	Дата		

```

bool XML::loadDoc(const std::string& fn, pugi::xml_document& doc) {
    if(!std::filesystem::exists(fn)) {
        spdlog::warn("[XML] '{}' not found - skipped", fn);
        return false;
    }
    if(!doc.load_file(fn.c_str())) {
        spdlog::warn("[XML] cannot parse '{}'", fn);
        return false;
    }
    return true;
}

/* tripinfo parsing */
Metrics XML::parseTripInfo(const std::string& file) {
    pugi::xml_document doc;
    if(!loadDoc(file, doc))
        throw std::runtime_error("tripinfo file missing: " + file);

    Metrics m{};
    double sumDur = 0, sumWait = 0, sumLen = 0;
    int cnt = 0, stops = 0;

    for(auto ti : doc.child("tripinfos").children("tripinfo")) {
        double dur = ti.attribute("duration").as_double();
        double wait = ti.attribute("waitingTime").as_double();
        double len = ti.attribute("routeLength").as_double();

        sumDur += dur;
        sumWait += wait;
        sumLen += len;
        ++cnt;
        if(wait > 0)
            ++stops;
    }
    if(cnt) {
        m.avg_travel_time = sumDur / cnt;
        m.avg_waiting_time = sumWait / cnt;
        m.avg_speed = sumLen / sumDur;
        m.throughput = cnt;
        m.num_stops = stops;
    }
    return m;
}

/* summary.xml */
void XML::enrichFromSummary(const std::string& file, Metrics& m) {
    pugi::xml_document doc;
    if(!loadDoc(file, doc))
        return;

    for(auto s : doc.child("summary").children("step"))
        m.halted_peak = std::max(m.halted_peak,
s.attribute("halting").as_double(0.0));
}

/* statistics.xml */
void XML::enrichFromStats(const std::string& fn, Metrics& m) {
    pugi::xml_document doc;
    if(!loadDoc(fn, doc))
        return;

    if(auto st = doc.child("statistics").child("vehicleTripStatistics"))

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				154
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        m.time_loss = st.attribute("timeLoss").as_double(m.time_loss);
    }

    /* queue.xml */
    void XML::enrichFromQueue(const std::string& file, Metrics& m) {
        pugi::xml_document doc;
        if(!loadDoc(file, doc))
            return;

        double sumQ = 0.0;
        int cntQ = 0;

        for(auto lane : doc.select_nodes("//lane")) {
            const double len = lane.node().attribute("queueing_length").as_double(0.0);
            m.queue_peak = std::max(m.queue_peak, len);
            sumQ += len;
            ++cntQ;
        }

        if(cntQ > 0)
            m.avg_queue_length = sumQ / static_cast<double>(cntQ);
    }

    /* edge.xml */
    void XML::enrichFromEdge(const std::string& file, Metrics& m) {
        pugi::xml_document doc;
        if(!loadDoc(file, doc))
            return;

        for(auto edge : doc.select_nodes("//edge")) {
            m.density_peak = std::max(m.density_peak,
            edge.node().attribute("density").as_double(0.0));
        }
    }

#include "TrafficPulseLibrary.h"

#include <TrafficPulseLibrary>
#include <chrono>
#include <iostream>
#include <memory>
#include <thread>

int main() {
    using namespace TrafficPulse;
    try {
        EventBus bus;
        TrafficSystem system(5.0, "config.sumocfg", "network.net.xml", bus);

        struct ObserverImpl : public IObservable {
            void update(const PulseEvent& ev) override {
                spdlog::info("Observer event: {}", ev.message);
            }
        };

        auto observer = std::make_shared<ObserverImpl>();
        system.attach(observer);

        std::thread sim_thread([&]() {
            system.run();
        });
    }
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				155
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        std::this_thread::sleep_for(std::chrono::seconds(30));
        system.requestStop();
        sim_thread.join();

        std::cout << "Insersections: "
                    << system.getDataManager().getAllIntersections().size() <<
std::endl;
    } catch (const std::exception& ex) {
        std::cerr << "Error: " << ex.what() << std::endl;
        return 1;
    }
    return 0;
}

cmake_minimum_required(VERSION 3.15)
project(PackageTest CXX)

find_library(LIBSUMOCPP sumocpp HINTS /usr/lib /usr/local/lib)
if(NOT LIBSUMOCPP)
    message(FATAL_ERROR "libsumocpp not found! Please install SUMO with libsumo-
cpp support.")
endif()

find_package(PugixML REQUIRED)
find_package(TrafficPulseLibrary CONFIG REQUIRED)

set(EXAMPLES_DIR "examples")

add_executable(PackageTest ${EXAMPLES_DIR}/application_example.cpp)
target_link_libraries(PackageTest PRIVATE TrafficPulseLibrary::TrafficPulseLibrary
${LIBSUMOCPP} pugixml)

import os

from conan import ConanFile
from conan.tools.cmake import CMake, cmake_layout
from conan.tools.build import can_run

class traffic_pulse_libraryTestConan(ConanFile):
    settings = "os", "compiler", "build_type", "arch"
    generators = "CMakeDeps", "CMakeToolchain"

    def requirements(self):
        self.requires(self.tested_reference_str)

    def build(self):
        cmake = CMake(self)
        cmake.configure()
        cmake.build()

    def layout(self):
        cmake_layout(self)

    def test(self):
        if can_run(self):
            cmd = os.path.join(self.cpp.build.bindir, "PackageTest")
            self.run(cmd, env="conanrun")

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				156
Змн.	Арк.	№ докум.	Підпис	Дата		

```

#pragma once

#ifndef PULSELOADERFIXTURE_H
#define PULSELOADERFIXTURE_H

#include <TrafficPulseLibrary>

#include <gtest/gtest.h>

using namespace TrafficPulse;

class PulseLoaderFixture : public ::testing::Test {
protected:
    void SetUp() override {
        auto& manager = PulseDataManager::getInstance();
        manager.clearAll();
    }

    static std::optional<std::filesystem::path> locateNet() {
        const std::filesystem::path cwd = std::filesystem::current_path();
        const std::filesystem::path repo = cwd / ".." / "..";

        for(constexpr const char* NET_REL =
"config/sumo/scenarios/xs/low_traffic/osm.net.xml";
            const std::filesystem::path p : {cwd / NET_REL, repo / NET_REL})
            if(std::filesystem::exists(p))
                return p;
        return std::nullopt;
    }
};

#endif // PULSELOADERFIXTURE_H

#pragma once

#ifndef PULSESTATSFIXTURE_H
#define PULSESTATSFIXTURE_H

#include <gtest/gtest.h>

#include <TrafficPulseLibrary>

using namespace TrafficPulse;

class PulseStatsFixture : public ::testing::Test {
protected:
    static inline std::unique_ptr<SumoIntegration> s_sumo;
    static inline bool s_skip_all = false;

    static std::optional<std::pair<std::string, std::string>> locateScenario() {
        using std::filesystem::path;
        const path cwd = std::filesystem::current_path();
        const path repo = cwd / ".." / "..";

        constexpr auto CFG = "config/sumo/test/test_config.sumocfg";
        constexpr auto NET = "config/sumo/test/test_net.net.xml";

        for(const path p : {cwd / CFG, repo / CFG}) {
            if(std::filesystem::exists(p) && std::filesystem::exists(repo / NET))
                return std::pair<std::string, std::string>{p.string(),
(p.parent_path() / "test_net.net.xml").string()};
        }
    }
};

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				157
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return std::nullopt;
    }

    static void SetUpTestSuite() {
        if(const auto p = locateScenario()) {
            s_sumo = std::make_unique<SumoIntegration>(p->first, p->second);
            s_sumo->startSimulation();
        }
        else {
            s_skip_all = true;
        }
    }

    static void TearDownTestSuite() {
        if(s_sumo && s_sumo->isRunning())
            s_sumo->stopSimulation();
        s_sumo.reset();
    }

    static void restartScenarioIfEnded() {
        if(!s_sumo || s_sumo->isRunning())
            return;

        if(const auto p = locateScenario()) {
            s_sumo = std::make_unique<SumoIntegration>(p->first, p->second);
            s_sumo->startSimulation();
        }
        else
            GTEST_SKIP() << "Scenario finished and files disappeared - skipping.";
    }

    void SetUp() override {
        if(s_skip_all)
            GTEST_SKIP() << "SUMO test scenario not found.";

        restartScenarioIfEnded();

        m_data_manager = &PulseDataManager::getInstance();
        m_data_manager->clearAll();

        m_stats = std::make_unique<PulseStatsCollector>(*m_data_manager, *s_sumo,
m_bus);

        auto inter = std::make_unique<PulseIntersection>("JunctionA",
PulsePosition{10, 20});
        inter->addInboundLane("edgeA_0");
        m_data_manager->addIntersection(std::move(inter));

        auto tl = std::make_unique<PulseTrafficLight>("JunctionA");
        tl->addControlledLane("edgeA_0");
        m_data_manager->addTrafficLight(std::move(tl));
    }

    PulseDataManager* m_data_manager = nullptr;
    std::unique_ptr<PulseStatsCollector> m_stats;
    EventBus m_bus;
};

#endif // PULSESTATSFIXTURE_H

#pragma once

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				158
Змн.	Арк.	№ докум.	Підпис	Дата		

```

#ifdef PULSESUMOINTEGRATIONFIXTURE_H
#define PULSESUMOINTEGRATIONFIXTURE_H

#include <TrafficPulseLibrary>
#include <filesystem>
#include <gtest/gtest.h>

using namespace TrafficPulse;

class PulseSumoIntegrationFixture : public ::testing::Test {
protected:
    static inline std::unique_ptr<SumoIntegration> s_sumo;
    static inline bool s_skip_all = false;

    static std::optional<std::pair<std::string, std::string>> locateScenario() {
        using std::filesystem::path;
        const path cwd = std::filesystem::current_path();
        const path repo = cwd / ".." / "..";

        constexpr auto CFG = "config/sumo/test/test_config.sumocfg";
        constexpr auto NET = "config/sumo/test/test_net.net.xml";

        for(const path base : {cwd, repo}) {
            path cfg = base / CFG;
            path net = base / NET;
            if(std::filesystem::exists(cfg) && std::filesystem::exists(net))
                return {{cfg.string(), net.string()}};
        }
        return std::nullopt;
    }

    static void SetUpTestSuite() {
        if(const auto sc = locateScenario()) {
            s_sumo = std::make_unique<SumoIntegration>(sc->first, sc->second);
            s_sumo->startSimulation();
        }
        else
            s_skip_all = true;
    }

    static void TearDownTestSuite() {
        if(s_sumo && s_sumo->isRunning())
            s_sumo->stopSimulation();
        s_sumo.reset();
    }

    static void ensureRunning() {
        if(s_sumo && s_sumo->isRunning())
            return;
        const auto sc = locateScenario();
        if(!sc) {
            s_skip_all = true;
            return;
        }
        s_sumo = std::make_unique<SumoIntegration>(sc->first, sc->second);
        s_sumo->startSimulation();
    }

    void SetUp() override {
        if(s_skip_all)
            GTEST_SKIP() << "SUMO test scenario not found - all tests skipped.";
        ensureRunning();
    }
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				159
Змн.	Арк.	№ докум.	Підпис	Дата		

```

};

#endif // PULSESUMOINTEGRATIONFIXTURE_H

#pragma once

#ifndef PULSETRAFFICALGOFIXTURE_H
#define PULSETRAFFICALGOFIXTURE_H

#include <TrafficPulseLibrary>
#include <filesystem>
#include <gtest/gtest.h>

using namespace TrafficPulse;

class PulseTrafficAlgoFixture : public ::testing::Test {
protected:
    static inline std::unique_ptr<SumoIntegration> s_sumo;
    static inline bool s_skip_all = false;

    static std::optional<std::pair<std::string, std::string>> locateScenario() {
        using std::filesystem::path;
        const path cwd = std::filesystem::current_path();
        const path repo = cwd / ".." / "..";

        for(constexpr auto CFG = "config/sumo/test/test_config.sumocfg"; const path
p_cfg : {cwd / CFG, repo / CFG}) {
            const path p_net = p_cfg.parent_path() / "test_net.net.xml";
            if(std::filesystem::exists(p_cfg) && std::filesystem::exists(p_net))
                return std::pair{p_cfg.string(), p_net.string()};
        }
        return std::nullopt;
    }

    static std::optional<std::string> locateYaml() {
        using std::filesystem::path;
        const path cwd = std::filesystem::current_path();

        for(const path repo = cwd / ".." / "..";
            const path p : {cwd / "config/tp_config.yaml", repo /
"config/tp_config.yaml"})
            if(std::filesystem::exists(p))
                return p.string();
        return std::nullopt;
    }

    static void SetUpTestSuite() {
        if(const auto p = locateScenario()) {
            s_sumo = std::make_unique<SumoIntegration>(p->first, p->second);
            s_sumo->startSimulation();
        }
        else {
            s_skip_all = true;
        }
    }

    static void TearDownTestSuite() {
        if(s_sumo && s_sumo->isRunning())
            s_sumo->stopSimulation();
        s_sumo.reset();
    }
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				160
Змн.	Арк.	№ докум.	Підпис	Дата		

```

void SetUp() override {
    if(s_skip_all)
        GTEST_SKIP() << "SUMO test scenario not found.";

    restartSumoIfEnded();

    data_manager = &PulseDataManager::getInstance();
    data_manager->clearAll();

    auto A = std::make_unique<PulseIntersection>("JunctionA", PulsePosition{0,
0});
    A->addInboundLane("edgeA_0");
    A->addInboundLane("edgeA_1");
    data_manager->addIntersection(std::move(A));

    auto B = std::make_unique<PulseIntersection>("JunctionB",
PulsePosition{100, 0});
    B->addInboundLane("edgeB_0");
    B->addInboundLane("edgeB_1");
    data_manager->addIntersection(std::move(B));

    data_manager-
>addTrafficLight(std::make_unique<PulseTrafficLight>("JunctionA"));
    data_manager-
>addTrafficLight(std::make_unique<PulseTrafficLight>("JunctionB"));

    stats = std::make_unique<PulseStatsCollector>(*data_manager, *s_sumo,
event_bus);

    Settings cfg;
    if(const auto yml = locateYaml())
        cfg = Settings::load(*yml);

    algo = std::make_unique<PulseTrafficAlgo>(*data_manager, *s_sumo,
event_bus, cfg);
    algo->setControlSimulationLights(true);
}

void TearDown() override {
    data_manager->clearAll();
}

static void restartSumoIfEnded() {
    if(s_sumo && s_sumo->isRunning())
        return;
    const auto sc = locateScenario();
    if(!sc) {
        s_skip_all = true;
        return;
    }
    s_sumo = std::make_unique<SumoIntegration>(sc->first, sc->second);
    s_sumo->startSimulation();
}

PulseDataManager* data_manager = nullptr;
std::unique_ptr<PulseStatsCollector> stats;
std::unique_ptr<PulseTrafficAlgo> algo;
EventBus event_bus;
};

#endif // PULSETRAFFICALGOFIXTURE_H

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІЗ	Арк.
		Власенко О.В.				161
Змн.	Арк.	№ докум.	Підпис	Дата		

```

#pragma once

#ifndef PUSEDATAMANAGERFIXTURE_H
#define PUSEDATAMANAGERFIXTURE_H

#include <TrafficPulseLibrary>
#include <gtest/gtest.h>

using namespace TrafficPulse;

class PulseDataManagerFixture : public ::testing::Test {
protected:
    static inline SumoIntegration* s_sumo = nullptr;
    static inline bool s_skip_all = false;

    static void SetUpTestSuite() {
        const auto paths = findTestScenario();
        if(!paths) {
            s_skip_all = true;
            return;
        }

        s_sumo = new SumoIntegration(paths->first, paths->second);
        s_sumo->startSimulation();
    }

    static void TearDownTestSuite() {
        if(s_sumo && s_sumo->isRunning()) {
            s_sumo->stopSimulation();
        }
        delete s_sumo;
        s_sumo = nullptr;
    }

    void SetUp() override {
        if(s_skip_all)
            GTEST_SKIP() << "SUMO test scenario files are missing - fixture
skipped.";

        PulseDataManager::getInstance().clearAll();
        ASSERT_NE(s_sumo, nullptr);
        ensureRunning();
        ASSERT_TRUE(s_sumo->isRunning());
    }

    static std::optional<std::pair<std::string, std::string>> findTestScenario() {
        constexpr auto CFG = "config/sumo/test/test_config.sumocfg";

        const std::filesystem::path cwd = std::filesystem::current_path();
        const std::filesystem::path repo = cwd / ".." / "..";

        for(const std::filesystem::path paths[2] = {cwd / CFG, repo / CFG}; const
auto& p_cfg : paths) {
            if(auto p_net = p_cfg.parent_path() / "test_net.net.xml";
                std::filesystem::exists(p_cfg) && std::filesystem::exists(p_net))
                return std::pair<std::string, std::string>{p_cfg.string(),
p_net.string()};
        }
        return std::nullopt;
    }

    static void ensureRunning() {

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІІЗ	Арк.
		Власенко О.В.				162
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if(!s_sumo)
            return;

        if(!s_sumo->isRunning())
            restartScenario();
    }

    static void restartScenario() {
        try {
            if(s_sumo)
                s_sumo->stopSimulation();
        }
        catch(...) {
        }
        delete s_sumo;

        const auto paths = findTestScenario();
        if(!paths) {
            s_skip_all = true;
            return;
        }

        s_sumo = new SumoIntegration(paths->first, paths->second);
        s_sumo->startSimulation();
    }
};

#endif // PUSEDATAMANAGERFIXTURE_H

#pragma once

#ifndef TRAFFICSYSTEMFIXTURE_H
#define TRAFFICSYSTEMFIXTURE_H

#include <TrafficPulseLibrary>

#include <gtest/gtest.h>
#include <memory>
#include <thread>

using namespace TrafficPulse;

class TrafficSystemFixture : public ::testing::Test {
protected:
    static inline bool s_cwd_fixed = false;

    static std::optional<std::pair<std::string, std::string>> locateScenario()
    {
        namespace fs = std::filesystem;
        const fs::path cwd = fs::current_path();

        for (const fs::path repo = cwd / ".." / ".."; const fs::path cfg : { cwd /
SUMO_CONFIG_PATH, repo / SUMO_CONFIG_PATH }) {
            if (const fs::path net = cfg.parent_path() / "osm.net.xml";
fs::exists(cfg) && fs::exists(net))
                return { { cfg.string(), net.string() } };
        }
        return std::nullopt;
    }

    static void normaliseCwd()
    {

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				163
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (s_cwd_fixed) return;
        const std::filesystem::path cwd = std::filesystem::current_path();
        if (cwd.filename() == "library")
            std::filesystem::current_path(cwd.parent_path());
        s_cwd_fixed = true;
    }

    void SetUp() override
    {
        normaliseCwd();

        const auto scenario = locateScenario();
        if (!scenario) {
            GTEST_SKIP() << "SUMO micro-scenario not found - TrafficSystem tests
skipped.";
        }

        if (!std::filesystem::exists("config/tp_config.yaml"))
            ASSERT_TRUE(std::filesystem::exists("build/config/tp_config.yaml"))
                << "tp_config.yaml not found.";

        event_bus = std::make_unique<EventBus>();
        traffic_system = std::make_unique<TrafficSystem>(
            /* step */ 5.0,
            scenario->first,
            scenario->second,
            *event_bus);
    }

    void TearDown() override {
        if(traffic_system) {
            traffic_system->requestStop();
        }

        if(sim_thread.joinable()) {
            sim_thread.join();
        }

        PulseDataManager::getInstance().clearAll();
    }

    std::unique_ptr<EventBus> event_bus;
    std::unique_ptr<TrafficSystem> traffic_system;
    std::thread sim_thread;
};

#endif // TRAFFICSYSTEMFIXTURE_H

#include <gtest/gtest.h>

#include <TrafficPulseLibrary>

using namespace TrafficPulse;

TEST(TrafficModeController, SwitchesModes) {
    auto c = TrafficModeController{Settings::FlowThresh{.low = 0.4,
                                                         .high = 0.6,
                                                         .min_hold = 0.5,
                                                         .queue = {.threshold = 10.0, .per_inter
= 5.0},
                                                         .dwell = {.max_pressure = 1.0, .other =
2.0}}}};

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				164
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    EXPECT_EQ(c.evaluate(0.7, {0.0}, 1.0), TrafficControlMode::Baseline);
    EXPECT_EQ(c.evaluate(0.7, {11.0}, 2.0), TrafficControlMode::MaxPressure);
}

#include <TrafficPulseLibrary>
#include <fixtures/PuseDataManagerFixture.h>

#include <chrono>
#include <gtest/gtest.h>
#include <thread>

using namespace TrafficPulse;

#define SKIP_IF_NO_SCENARIO()
\
    if(PulseDataManagerFixture::s_skip_all)
\
        GTEST_SKIP();

/**
 * @test Adds two intersections and ensures a duplicate ID throws
 *       `PulseException`.
 */
TEST_F(PulseDataManagerFixture, AddIntersectionNoDup) {
    SKIP_IF_NO_SCENARIO();
    auto& m = PulseDataManager::getInstance();

    m.addIntersection(std::make_unique<PulseIntersection>("intA", PulsePosition{1,
2}));
    m.addIntersection(std::make_unique<PulseIntersection>("intB", PulsePosition{3,
4}));

    EXPECT_EQ(m.getAllIntersections().size(), 2u);
    EXPECT_THROW(m.addIntersection(std::make_unique<PulseIntersection>("intA",
PulsePosition{9, 9})), PulseException);
}

/**
 * @test Duplicate-ID protection for traffic lights.
 */
TEST_F(PulseDataManagerFixture, AddTrafficLightNoDup) {
    SKIP_IF_NO_SCENARIO();
    auto& m = PulseDataManager::getInstance();

    m.addTrafficLight(std::make_unique<PulseTrafficLight>("TL_A"));
    m.addTrafficLight(std::make_unique<PulseTrafficLight>("TL_B"));

    EXPECT_EQ(m.getAllTrafficLights().size(), 2u);
    EXPECT_THROW(m.addTrafficLight(std::make_unique<PulseTrafficLight>("TL_A")),
PulseException);
}

/**
 * @test Duplicate-ID protection for vehicles.
 */
TEST_F(PulseDataManagerFixture, AddVehicleNoDup) {
    SKIP_IF_NO_SCENARIO();
    auto& m = PulseDataManager::getInstance();

    m.addVehicle(
        std::make_unique<PulseVehicle>("veh1", PulseVehicleType::CAR,

```

		Бабушко А.С.			ІІЗ.КР.Б.-121-25-ІІЗ	Арк.
		Власенко О.В.				165
Змн.	Арк.	№ докум.	Підпис	Дата		

```

PulseVehicleRole::NORMAL, PulsePosition{5, 5}));
    m.addVehicle(std::make_unique<PulseVehicle>(
        "veh2", PulseVehicleType::TRUCK, PulseVehicleRole::NORMAL,
        PulsePosition{10, 20}));

    EXPECT_EQ(m.getAllVehicles().size(), 2u);
    EXPECT_THROW(m.addVehicle(std::make_unique<PulseVehicle>(
        "veh1", PulseVehicleType::CAR, PulseVehicleRole::NORMAL,
        PulsePosition{0, 0})),
        PulseException);
}

/**
 * @test Passing a nullptr must raise `std::invalid_argument`.
 */
TEST_F(PulseDataManagerFixture, NullPointerThrowsInvalidArgument) {
    SKIP_IF_NO_SCENARIO();
    auto& m = PulseDataManager::getInstance();
    EXPECT_THROW(m.addIntersection(nullptr), std::invalid_argument);
    EXPECT_THROW(m.addTrafficLight(nullptr), std::invalid_argument);
    EXPECT_THROW(m.addVehicle(nullptr), std::invalid_argument);
}

#include <filesystem>
#include <gtest/gtest.h>
#include <ranges>

#include "core/PulseLoader.h"
#include "fixtures/PulseLoaderFixture.h"
#include <TrafficPulseLibrary>

using namespace TrafficPulse;

TEST_F(PulseLoaderFixture, LoaderFillsDataManager) {
    auto& manager = PulseDataManager::getInstance();

    const auto net_path = locateNet();
    if(!net_path)
        GTEST_SKIP() << "XS test-scenario network not found - loader test
skipped.";

    EventBus event_bus;
    bool started = false;
    bool completed = false;
    float last_progress = 0.0f;

    event_bus.subscribe<NetworkLoadingStartedEvent>([&](const auto&) { started =
true; });
    event_bus.subscribe<NetworkLoadingProgressEvent>([&](const auto& ev) {
last_progress = ev.progress; });
    event_bus.subscribe<NetworkLoadingCompletedEvent>([&](const auto&) { completed
= true; });

    const PulseLoader loader(manager, *net_path, event_bus);
    EXPECT_NO_THROW(loader.loadNetworkData());

    EXPECT_TRUE(started);
    EXPECT_TRUE(completed);
    EXPECT_GE(last_progress, 0.95f);

    const auto intersections = manager.getAllIntersections();
    EXPECT_GT(intersections.size(), 0u);

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІПЗ	Арк.
		Власенко О.В.				166
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const auto trafficLights = manager.getAllTrafficLights();
EXPECT_GT(trafficLights.size(), 0u);

for(const auto* inter : intersections) {
    static bool found_signalised = false;
    for(const auto& conn : inter->getConnectedRoads() | std::views::values)
        if(conn.getTrafficLight() != nullptr)
            found_signalised = true;

    if(inter == intersections.back())
        EXPECT_TRUE(found_signalised) << "Loader did not assign any traffic-
light pointers - "
                                "expected at least one signalised connection.";
}
}

#include <gtest/gtest.h>

#include "fixtures/PulseStatsFixture.h"
#include <TrafficPulseLibrary>

using namespace TrafficPulse;

#define SKIP_IF_NO_STATS()
\
    if(!m_stats)
\
    GTEST_SKIP()

/**
 * @test Verifies that the collector publishes a StatsUpdatedEvent and that
 *         the basic aggregates (average speed, vehicles per type) are
 *         non-negative after 10 s of simulation time.
 */
TEST_F(PulseStatsFixture, CheckLaneStats) {
    SKIP_IF_NO_STATS();

    bool event_received = false;
    StatsUpdatedEvent e{};

    m_bus.subscribe<StatsUpdatedEvent>([&](const StatsUpdatedEvent& ev) {
        event_received = true;
        e = ev;
    });

    for(int i = 0; i < 10; ++i)
        s_sumo->stepSimulation();
    m_stats->collectStepData(10.0);

    EXPECT_TRUE(event_received);
    EXPECT_GE(e.avg_speed, 0.0);
    EXPECT_GE(e.vehicle_type_counts[PulseVehicleType::CAR], 0);
}

/**
 * @test Ensures the intersection counters stay non-negative after 15 s of
 *         network activity.
 */
TEST_F(PulseStatsFixture, CheckIntersectionStats) {
    SKIP_IF_NO_STATS();
    for(int i = 0; i < 15; ++i)

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				167
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        s_sumo->stepSimulation();
        m_stats->collectStepData(15.0);

        const auto s = m_stats->getIntersectionStats("JunctionA");
        EXPECT_GE(s.total_wait_time, 0.0);
        EXPECT_GE(s.vehicles_passed, 0);
    }

    /**
     * @test Stress test with twenty simulation steps: waiting-count on the
     *         first incoming lane must never be negative, and total wait time is
     *         recorded.
     */
    TEST_F(PulseStatsFixture, MultipleVehiclesLaneSaturation) {
        SKIP_IF_NO_STATS();

        for(int i = 0; i < 20; ++i)
            s_sumo->stepSimulation();
        m_stats->collectStepData(20.0);

        EXPECT_GE(m_stats->getLaneWaitingCount("edgeA_0"), 0);
        EXPECT_GE(m_stats->getIntersectionStats("JunctionA").total_wait_time, 0.0);
    }

    /**
     * @test Checks monotonic growth of the 'vehicles_passed' counter.
     */
    TEST_F(PulseStatsFixture, VehiclesPassedMonotone) {
        SKIP_IF_NO_STATS();

        /* first snapshot */
        for(int i = 0; i < 20; ++i)
            s_sumo->stepSimulation();
        m_stats->collectStepData(20.0);
        const int passed_20 = m_stats->getIntersectionStats("JunctionA").vehicles_passed;

        /* later snapshot */
        for(int i = 0; i < 30; ++i)
            s_sumo->stepSimulation();
        m_stats->collectStepData(50.0);
        const int passed_50 = m_stats->getIntersectionStats("JunctionA").vehicles_passed;

        EXPECT_GE(passed_50, passed_20);
    }

    /**
     * @test After one simulated minute, the collector must have accumulated
     *         non-negative green/red time and a count of vehicles that stopped
     *         at red on the same junction.
     */
    TEST_F(PulseStatsFixture, LightTimingCountersExist) {
        SKIP_IF_NO_STATS();
        for(int i = 0; i < 60; ++i)
            s_sumo->stepSimulation();
        m_stats->collectStepData(60.0);

        EXPECT_GE(m_stats->getTimeSpentGreen("JunctionA"), 0.0);
        EXPECT_GE(m_stats->getTimeSpentRed("JunctionA"), 0.0);
        EXPECT_GE(m_stats->getVehiclesStoppedAtRed("JunctionA"), 0);
    }

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				168
Змн.	Арк.	№ докум.	Підпис	Дата		

```

/**
 * @test Emulates staggered vehicle departure times (0 s and 15 s) and
 *        confirms that lane waiting counts and intersection wait time are
 *        still reported without exceptions.
 */
TEST_F(PulseStatsFixture, StaggeredVehicleDepartures) {
    SKIP_IF_NO_STATS();

    for(int i = 0; i < 10; ++i)
        s_sumo->stepSimulation();
    m_stats->collectStepData(10.0);

    for(int i = 10; i < 30; ++i)
        s_sumo->stepSimulation();
    m_stats->collectStepData(30.0);

    EXPECT_GE(m_stats->getLaneWaitingCount("edgeA_0"), 0);
    EXPECT_GE(m_stats->getIntersectionStats("JunctionA").total_wait_time, 0.0);
}

/**
 * @test Ensures the waiting counters on a multi-lane edge remain
 *        non-negative when vehicles are allowed to change lanes.
 */
TEST_F(PulseStatsFixture, MultiLaneEdge) {
    SKIP_IF_NO_STATS();
    for(int i = 0; i < 40; ++i)
        s_sumo->stepSimulation();
    m_stats->collectStepData(40.0);

    EXPECT_GE(m_stats->getLaneWaitingCount("edgeA_0"), 0);
    EXPECT_GE(m_stats->getLaneWaitingCount("edgeA_1"), 0);
}

#include "fixtures/PulseTrafficAlgoFixture.h"
#include <TrafficPulseLibrary>
#include <gtest/gtest.h>
#include <thread>

using namespace TrafficPulse;

#define SKIP_IF_NO_SUMO()
\
    if(PulseTrafficAlgoFixture::s_skip_all)
\
    GTEST_SKIP()

/**
 * @test Green-wave scheduling runs without exceptions; both signals
 *        end in a valid RED/GREEN state after 10 simulation steps.
 */
TEST_F(PulseTrafficAlgoFixture, NormalGreenWaveScheduling) {
    SKIP_IF_NO_SUMO();
    for(int i = 0; i < 10; ++i)
        s_sumo->stepSimulation();

    algo->runOnce(10.0);

    ASSERT_NE(data_manager->getTrafficLight("JunctionA"), nullptr);
    ASSERT_NE(data_manager->getTrafficLight("JunctionB"), nullptr);
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				169
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    EXPECT_NE(data_manager->getTrafficLight("JunctionA")->getState(),
TrafficLightState::YELLOW);
    EXPECT_NE(data_manager->getTrafficLight("JunctionB")->getState(),
TrafficLightState::YELLOW);
}

/**
 * @test After 30 steps, the algorithm can still handle a two-junction
 *        corridor, and every light is RED or GREEN.
 */
TEST_F(PulseTrafficAlgoFixture, MultiIntersectionCorridor) {
    SKIP_IF_NO_SUMO();
    for(int i = 0; i < 30; ++i)
        s_sumo->stepSimulation();
    algo->runOnce(30.0);

    EXPECT_NE(data_manager->getTrafficLight("JunctionA")->getState(),
TrafficLightState::YELLOW);
    EXPECT_NE(data_manager->getTrafficLight("JunctionB")->getState(),
TrafficLightState::YELLOW);
}

/** @test runOnce() must not throw when the network is empty. */
TEST_F(PulseTrafficAlgoFixture, NoIntersectionsCase) {
    SKIP_IF_NO_SUMO();
    data_manager->clearAll();
    EXPECT_NO_THROW(algo->runOnce(5.0));
}

/** @test `TrafficModeChangedEvent` is emitted when switching modes. */
TEST_F(PulseTrafficAlgoFixture, TrafficModeChangeDispatch) {
    SKIP_IF_NO_SUMO();
    bool triggered = false;
    auto new_mode = TrafficControlMode::Baseline;

    event_bus.subscribe<TrafficModeChangedEvent>([&](const auto& ev) {
        triggered = true;
        new_mode = ev.new_mode;
    });

    algo->setTrafficMode(TrafficControlMode::Hybrid);

    EXPECT_TRUE(triggered);
    EXPECT_EQ(new_mode, TrafficControlMode::Hybrid);
}

#include "fixtures/PulseSumoIntegrationFixture.h"
#include <TrafficPulseLibrary>
#include <chrono>
#include <gtest/gtest.h>
#include <thread>

using namespace TrafficPulse;

#define SKIP_IF_NO_SUMO()
\
    if(PulseSumoIntegrationFixture::s_skip_all)
\
    GTEST_SKIP()

/** @test The simulation can be stopped and restarted without throwing. */
TEST_F(PulseSumoIntegrationFixture, StartAndStopSimulation) {

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				170
Змн.	Арк.	№ докум.	Підпис	Дата		

```

SKIP_IF_NO_SUMO();
ASSERT_TRUE(s_sumo->isRunning());

s_sumo->stopSimulation();
EXPECT_FALSE(s_sumo->isRunning());

EXPECT_NO_THROW(s_sumo->startSimulation());
EXPECT_TRUE(s_sumo->isRunning());
}

/** @test Stepping the simulation ten times takes less than 6 s wall-time. */
TEST_F(PulseSumoIntegrationFixture, StepSimulationAndTime) {
    SKIP_IF_NO_SUMO();
    constexpr int kSteps = 10;

    const auto t0 = std::chrono::steady_clock::now();
    for(int i = 0; i < kSteps; ++i) {
        EXPECT_NO_THROW(s_sumo->stepSimulation());
        std::this_thread::sleep_for(std::chrono::milliseconds(200));
    }
    const auto dt =

std::chrono::duration_cast<std::chrono::milliseconds>(std::chrono::steady_clock::now() - t0).count();
    EXPECT_LT(dt, 6000) << "Ten steps took " << dt << " ms";
}

/** @test At least one vehicle appears within 20 simulation steps. */
TEST_F(PulseSumoIntegrationFixture, VehiclesEventuallyAppear) {
    SKIP_IF_NO_SUMO();
    bool found = false;

    for(int i = 0; i < 20 && s_sumo->isRunning(); ++i) {
        s_sumo->stepSimulation();
        if(!s_sumo->getAllVehicles().empty()) {
            found = true;
            break;
        }
        std::this_thread::sleep_for(std::chrono::milliseconds(200));
    }
    EXPECT_TRUE(found);

    if(found) {
        const auto veh_id = s_sumo->getAllVehicles().front();
        auto pos = s_sumo->getVehiclePosition(veh_id);
        EXPECT_TRUE(std::isfinite(pos.first));
        EXPECT_TRUE(std::isfinite(pos.second));
    }
}

/** @test Reading and toggling the first traffic light's state works. */
TEST_F(PulseSumoIntegrationFixture, TrafficLightsCheck) {
    SKIP_IF_NO_SUMO();
    const auto lights = s_sumo->getAllTrafficLights();
    if(lights.empty())
        GTEST_SKIP() << "Scenario has no signals.";

    const auto& id = lights.front();
    const std::string original = s_sumo->getTrafficLightState(id);
    ASSERT_FALSE(original.empty());

    std::string flipped = original;

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				171
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    for(char& c : flipped)
        c = (c == 'r' ? 'g' : c == 'g' ? 'r' : c);
    EXPECT_NO_THROW(s_sumo->setTrafficLightState(id, flipped));
    EXPECT_EQ(s_sumo->getTrafficLightState(id), flipped);

    EXPECT_NO_THROW(s_sumo->setTrafficLightState(id, original));
}

/** @test 100 simulation steps complete in under 8 s (debug build). */
TEST_F(PulseSumoIntegrationFixture, PerformanceTest) {
    SKIP_IF_NO_SUMO();
    constexpr int kSteps = 100;
    const auto t0 = std::chrono::steady_clock::now();

    for(int i = 0; i < kSteps && s_sumo->isRunning(); ++i)
        s_sumo->stepSimulation();

    const auto dt =

std::chrono::duration_cast<std::chrono::milliseconds>(std::chrono::steady_clock::now() - t0).count();
    EXPECT_LT(dt, 8000) << kSteps << " steps took " << dt << " ms";
}

#include <any>
#include <chrono>
#include <gtest/gtest.h>
#include <memory>
#include <thread>
#include <vector>

#include "fixtures/TrafficSystemFixture.h"
#include <TrafficPulseLibrary>

using namespace TrafficPulse;

/**
 * @test Ensures that traffic light states are changed during simulation.
 */
TEST_F(TrafficSystemFixture, TrafficLightStateChangesAreReflected) {
    sim_thread = std::thread([&]() { traffic_system->run(); });

    std::this_thread::sleep_for(std::chrono::seconds(3));
    traffic_system->requestStop();
    sim_thread.join();

    const auto& data_manager = traffic_system->getDataManager();
    const auto lights = data_manager.getAllTrafficLights();

    ASSERT_GT(lights.size(), 0u) << "No traffic lights were loaded from SUMO.";

    bool has_green = false, has_red = false, has_yellow = false;
    for(const auto* light : lights) {
        switch(light->getState()) {
            case TrafficLightState::GREEN:
                has_green = true;
                break;
            case TrafficLightState::RED:
                has_red = true;
                break;
            case TrafficLightState::YELLOW:
                has_yellow = true;
                break;
        }
    }
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				172
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        break;
    }
}

EXPECT_TRUE(has_green || has_red || has_yellow);
}

/**
 * @test Ensures that system resources are properly cleaned up on stop.
 */
TEST_F(TrafficSystemFixture, ResourceCleanupOnStop) {
    sim_thread = std::thread([&]() { traffic_system->run(); });

    std::this_thread::sleep_for(std::chrono::seconds(3));
    traffic_system->requestStop();
    sim_thread.join();

    auto& data_manager = traffic_system->getDataManager();
    EXPECT_NO_THROW({ data_manager.clearAll(); });
}

/**
 * @test Ensures TrafficSystem can be started again after being stopped.
 */
TEST_F(TrafficSystemFixture, RestartAfterStop) {
    sim_thread = std::thread([&]() { traffic_system->run(); });
    std::this_thread::sleep_for(std::chrono::seconds(2));
    traffic_system->requestStop();
    sim_thread.join();

    sim_thread = std::thread([&]() { traffic_system->run(); });
    std::this_thread::sleep_for(std::chrono::seconds(2));
    traffic_system->requestStop();
    sim_thread.join();

    SUCCEED();
}

add_executable(LibraryTests
    src/controllers/TrafficModeController.cpp

    src/core/PulseDataManagerTest.cpp
    src/core/PulseLoaderTest.cpp
    src/core/PulseStatsCollectorTest.cpp
    src/core/PulseTrafficAlgoTest.cpp
    src/core/SumoIntegrationTest.cpp
    src/core/TrafficSystemTest.cpp
)

target_include_directories(LibraryTests
    PRIVATE
    ${CMAKE_CURRENT_LIST_DIR}/include
)

get_filename_component(TESTS_PARENT_DIR "${CMAKE_CURRENT_BINARY_DIR}/.." REALPATH)

target_compile_definitions(TrafficPulseLibrary PUBLIC
    TRAFFIC_PULSE_TESTS_BUILD_PARENT="${TESTS_PARENT_DIR}"
)

target_compile_definitions(LibraryTests PRIVATE
    TRAFFIC_PULSE_TESTS_BUILD_PARENT="${TESTS_PARENT_DIR}"
)

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		173

```

)

target_link_libraries(LibraryTests
    PRIVATE
    TrafficPulseLibrary
    gtest_main
)

include(GoogleTest)
gtest_discover_tests(LibraryTests PROPERTIES LABELS "library")

cmake_minimum_required(VERSION 3.18)
project(TrafficPulseLibrary VERSION 1.0 LANGUAGES CXX)

set(CMAKE_CXX_STANDARD 20)
set(CMAKE_CXX_STANDARD_REQUIRED ON)
set(PROJECT_ROOT ${CMAKE_SOURCE_DIR})

include_directories(${CMAKE_CURRENT_SOURCE_DIR}/include)

file(MAKE_DIRECTORY ${PROJECT_ROOT}/config/sumo)
execute_process(COMMAND ${CMAKE_COMMAND} -E copy_directory
    ${PROJECT_ROOT}/config/sumo
    ${CMAKE_CURRENT_BINARY_DIR}/config/sumo
)
configure_file(${PROJECT_ROOT}/config/tp_config.yaml
    ${CMAKE_BINARY_DIR}/config/tp_config.yaml COPYONLY)

file(GLOB_RECURSE LIBRARY_SOURCES ${CMAKE_CURRENT_SOURCE_DIR}/src/*.cpp)

add_definitions(-DCMAKE_BINARY_DIR="${CMAKE_BINARY_DIR}")

find_package(PugixML REQUIRED)
find_package(spdlog CONFIG REQUIRED)
find_package(yaml-cpp REQUIRED)
add_library(TrafficPulseLibrary STATIC ${LIBRARY_SOURCES})

target_include_directories(TrafficPulseLibrary PUBLIC
    ${BUILD_INTERFACE:${CMAKE_CURRENT_SOURCE_DIR}/include}
    ${INSTALL_INTERFACE:include}
)

find_library(LIBSUMOCPP sumocpp HINTS /usr/lib /usr/local/lib)
if(NOT LIBSUMOCPP)
    message(FATAL_ERROR "libsumocpp not found! Please install SUMO with libsumo-
cpp support.")
endif()

target_link_libraries(TrafficPulseLibrary PRIVATE ${LIBSUMOCPP} pugixml PUBLIC
    spdlog::spdlog PUBLIC yaml-cpp::yaml-cpp PUBLIC fmt::fmt)

include(FetchContent)
FetchContent_Declare(
    googletest
    GIT_REPOSITORY https://github.com/google/googletest.git
    GIT_TAG v1.15.2
)
FetchContent_MakeAvailable(googletest)
option(BUILD_TESTING "Build unit tests" ON)
if(BUILD_TESTING)
    enable_testing()

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				174
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if(EXISTS "${CMAKE_CURRENT_SOURCE_DIR}/tests")
            add_subdirectory(tests)
        endif()
    endif()

install(TARGETS TrafficPulseLibrary
        EXPORT TrafficPulseLibraryTargets
        DESTINATION lib
)
install(DIRECTORY include/ DESTINATION include)
install(EXPORT TrafficPulseLibraryTargets
        FILE TrafficPulseLibraryConfig.cmake
        DESTINATION lib/cmake/TrafficPulseLibrary
)

from conan import ConanFile
from conan.tools.cmake import CMakeToolchain, CMake, cmake_layout, CMakeDeps

class traffic_pulse_libraryRecipe(ConanFile):
    name = "traffic_pulse_library"
    version = "1.0.0"
    package_type = "library"

    # TODO
    # Optional metadata
    license = "<Put the package license here>"
    author = "<Put your name here> <And your email here>"
    url = "<Package recipe repository url here, for issues about the package>"
    description = "<Description of traffic_pulse_library package here>"
    topics = ("<Put some tag here>", "<here>", "<and here>")

    # Binary configuration
    settings = "os", "compiler", "build_type", "arch"
    options = {"shared": [True, False], "fPIC": [True, False]}
    default_options = {"shared": False, "fPIC": True}

    # Sources are located in the same place as this recipe, copy them to the
    recipe
    exports_sources = "CMakeLists.txt", "src/*", "include/*", "config/*"

    def config_options(self):
        if self.settings.os == "Windows":
            self.options.rm_safe("fPIC")

    def configure(self):
        if self.options.shared:
            self.options.rm_safe("fPIC")

    def layout(self):
        cmake_layout(self)

    def generate(self):
        deps = CMakeDeps(self)
        deps.generate()
        tc = CMakeToolchain(self)
        tc.generate()

    def build(self):
        cmake = CMake(self)
        cmake.configure()
        cmake.build()

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				175
Змн.	Арк.	№ докум.	Підпис	Дата		

```

def package(self):
    cmake = CMake(self)
    cmake.install()

def package_info(self):
    self.cpp_info.libs = ["traffic_pulse_library"]

# Doxyfile 1.13.2

# This file describes the settings to be used by the documentation system
# Doxygen (www.doxygen.org) for a project.
#
# All text after a double hash (##) is considered a comment and is placed in
# front of the TAG it is preceding.
#
# All text after a single hash (#) is considered a comment and will be ignored.
# The format is:
# TAG = value [value, ...]
# For lists, items can also be appended using:
# TAG += value [value, ...]
# Values that contain spaces should be placed between quotes (\ " \").
#
# Note:
#
# Use Doxygen to compare the used configuration file with the template
# configuration file:
# doxygen -x [configFile]
# Use Doxygen to compare the used configuration file with the template
# configuration file without replacing the environment variables or CMake type
# replacement variables:
# doxygen -x_noenv [configFile]

#-----
# Project related configuration options
#-----

# This tag specifies the encoding used for all characters in the configuration
# file that follow. The default is UTF-8 which is also the encoding used for all
# text before the first occurrence of this tag. Doxygen uses libiconv (or the
# iconv built into libc) for the transcoding. See
# https://www.gnu.org/software/libiconv/ for the list of possible encodings.
# The default value is: UTF-8.

DOXYFILE_ENCODING      = UTF-8

# The PROJECT_NAME tag is a single word (or a sequence of words surrounded by
# double-quotes, unless you are using Doxywizard) that should identify the
# project for which the documentation is generated. This name is used in the
# title of most generated pages and in a few other places.
# The default value is: My Project.

PROJECT_NAME           = "PulseTrafficLibrary"

# The PROJECT_NUMBER tag can be used to enter a project or revision number. This
# could be handy for archiving the generated documentation or if some version
# control system is used.

PROJECT_NUMBER         =

# Using the PROJECT_BRIEF tag one can provide an optional one line description
# for a project that appears at the top of each page and should give viewers a

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		176

```

# quick idea about the purpose of the project. Keep the description short.

PROJECT_BRIEF          =

# With the PROJECT_LOGO tag one can specify a logo or an icon that is included
# in the documentation. The maximum height of the logo should not exceed 55
# pixels and the maximum width should not exceed 200 pixels. Doxygen will copy
# the logo to the output directory.

PROJECT_LOGO           =

# With the PROJECT_ICON tag one can specify an icon that is included in the tabs
# when the HTML document is shown. Doxygen will copy the logo to the output
# directory.

PROJECT_ICON           =

# The OUTPUT_DIRECTORY tag is used to specify the (relative or absolute) path
# into which the generated documentation will be written. If a relative path is
# entered, it will be relative to the location where Doxygen was started. If
# left blank the current directory will be used.

OUTPUT_DIRECTORY       =

# If the CREATE_SUBDIRS tag is set to YES then Doxygen will create up to 4096
# sub-directories (in 2 levels) under the output directory of each output format
# and will distribute the generated files over these directories. Enabling this
# option can be useful when feeding Doxygen a huge amount of source files, where
# putting all generated files in the same directory would otherwise cause
# performance problems for the file system. Adapt CREATE_SUBDIRS_LEVEL to
# control the number of sub-directories.
# The default value is: NO.

CREATE_SUBDIRS         = NO

# Controls the number of sub-directories that will be created when
# CREATE_SUBDIRS tag is set to YES. Level 0 represents 16 directories, and every
# level increment doubles the number of directories, resulting in 4096
# directories at level 8 which is the default and also the maximum value. The
# sub-directories are organized in 2 levels, the first level always has a fixed
# number of 16 directories.
# Minimum value: 0, maximum value: 8, default value: 8.
# This tag requires that the tag CREATE_SUBDIRS is set to YES.

CREATE_SUBDIRS_LEVEL   = 8

# If the ALLOW_UNICODE_NAMES tag is set to YES, Doxygen will allow non-ASCII
# characters to appear in the names of generated files. If set to NO, non-ASCII
# characters will be escaped, for example _xE3_x81_x84 will be used for Unicode
# U+3044.
# The default value is: NO.

ALLOW_UNICODE_NAMES    = NO

# The OUTPUT_LANGUAGE tag is used to specify the language in which all
# documentation generated by Doxygen is written. Doxygen will use this
# information to generate all constant output in the proper language.
# Possible values are: Afrikaans, Arabic, Armenian, Brazilian, Bulgarian,
# Catalan, Chinese, Chinese-Traditional, Croatian, Czech, Danish, Dutch, English
# (United States), Esperanto, Farsi (Persian), Finnish, French, German, Greek,
# Hindi, Hungarian, Indonesian, Italian, Japanese, Japanese-en (Japanese with
# English messages), Korean, Korean-en (Korean with English messages), Latvian,

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		177

```

# Lithuanian, Macedonian, Norwegian, Persian (Farsi), Polish, Portuguese,
# Romanian, Russian, Serbian, Serbian-Cyrillic, Slovak, Slovene, Spanish,
# Swedish, Turkish, Ukrainian and Vietnamese.
# The default value is: English.

OUTPUT_LANGUAGE          = English

# If the BRIEF_MEMBER_DESC tag is set to YES, Doxygen will include brief member
# descriptions after the members that are listed in the file and class
# documentation (similar to Javadoc). Set to NO to disable this.
# The default value is: YES.

BRIEF_MEMBER_DESC        = YES

# If the REPEAT_BRIEF tag is set to YES, Doxygen will prepend the brief
# description of a member or function before the detailed description
#
# Note: If both HIDE_UNDOC_MEMBERS and BRIEF_MEMBER_DESC are set to NO, the
# brief descriptions will be completely suppressed.
# The default value is: YES.

REPEAT_BRIEF              = YES

# This tag implements a quasi-intelligent brief description abbreviator that is
# used to form the text in various listings. Each string in this list, if found
# as the leading text of the brief description, will be stripped from the text
# and the result, after processing the whole list, is used as the annotated
# text. Otherwise, the brief description is used as-is. If left blank, the
# following values are used ($name is automatically replaced with the name of
# the entity):The $name class, The $name widget, The $name file, is, provides,
# specifies, contains, represents, a, an and the.

ABBREVIATE_BRIEF          = "The $name class" \
                             "The $name widget" \
                             "The $name file" \
                             is \
                             provides \
                             specifies \
                             contains \
                             represents \
                             a \
                             an \
                             the

# If the ALWAYS_DETAILED_SEC and REPEAT_BRIEF tags are both set to YES then
# Doxygen will generate a detailed section even if there is only a brief
# description.
# The default value is: NO.

ALWAYS_DETAILED_SEC       = NO

# If the INLINE_INHERITED_MEMB tag is set to YES, Doxygen will show all
# inherited members of a class in the documentation of that class as if those
# members were ordinary class members. Constructors, destructors and assignment
# operators of the base classes will not be shown.
# The default value is: NO.

INLINE_INHERITED_MEMB     = NO

# If the FULL_PATH_NAMES tag is set to YES, Doxygen will prepend the full path
# before files name in the file list and in the header files. If set to NO the
# shortest path that makes the file name unique will be used

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		178

```

# The default value is: YES.

FULL_PATH_NAMES          = YES

# The STRIP_FROM_PATH tag can be used to strip a user-defined part of the path.
# Stripping is only done if one of the specified strings matches the left-hand
# part of the path. The tag can be used to show relative paths in the file list.
# If left blank the directory from which Doxygen is run is used as the path to
# strip.
#
# Note that you can specify absolute paths here, but also relative paths, which
# will be relative from the directory where Doxygen is started.
# This tag requires that the tag FULL_PATH_NAMES is set to YES.

STRIP_FROM_PATH          =

# The STRIP_FROM_INC_PATH tag can be used to strip a user-defined part of the
# path mentioned in the documentation of a class, which tells the reader which
# header file to include in order to use a class. If left blank only the name of
# the header file containing the class definition is used. Otherwise one should
# specify the list of include paths that are normally passed to the compiler
# using the -I flag.

STRIP_FROM_INC_PATH      =

# If the SHORT_NAMES tag is set to YES, Doxygen will generate much shorter (but
# less readable) file names. This can be useful if your file system doesn't
# support long names like on DOS, Mac, or CD-ROM.
# The default value is: NO.

SHORT_NAMES              = NO

# If the JAVADOC_AUTOBRIEF tag is set to YES then Doxygen will interpret the
# first line (until the first dot, question mark or exclamation mark) of a
# Javadoc-style comment as the brief description. If set to NO, the Javadoc-
# style will behave just like regular Qt-style comments (thus requiring an
# explicit @brief command for a brief description.)
# The default value is: NO.

JAVADOC_AUTOBRIEF        = NO

# If the JAVADOC_BANNER tag is set to YES then Doxygen will interpret a line
# such as
# /*****
# as being the beginning of a Javadoc-style comment "banner". If set to NO, the
# Javadoc-style will behave just like regular comments and it will not be
# interpreted by Doxygen.
# The default value is: NO.

JAVADOC_BANNER           = NO

# If the QT_AUTOBRIEF tag is set to YES then Doxygen will interpret the first
# line (until the first dot, question mark or exclamation mark) of a Qt-style
# comment as the brief description. If set to NO, the Qt-style will behave just
# like regular Qt-style comments (thus requiring an explicit \brief command for
# a brief description.)
# The default value is: NO.

QT_AUTOBRIEF             = NO

# The MULTILINE_CPP_IS_BRIEF tag can be set to YES to make Doxygen treat a
# multi-line C++ special comment block (i.e. a block of /// or ///< comments) as

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		179

```
# a brief description. This used to be the default behavior. The new default is
# to treat a multi-line C++ comment block as a detailed description. Set this
# tag to YES if you prefer the old behavior instead.
#
# Note that setting this tag to YES also means that rational rose comments are
# not recognized any more.
# The default value is: NO.
```

```
MULTILINE_CPP_IS_BRIEF = NO
```

```
# By default Python docstrings are displayed as preformatted text and Doxygen's
# special commands cannot be used. By setting PYTHON_DOCSTRING to NO the
# Doxygen's special commands can be used and the contents of the docstring
# documentation blocks is shown as Doxygen documentation.
# The default value is: YES.
```

```
PYTHON_DOCSTRING = YES
```

```
# If the INHERIT_DOCS tag is set to YES then an undocumented member inherits the
# documentation from any documented member that it re-implements.
# The default value is: YES.
```

```
INHERIT_DOCS = YES
```

```
# If the SEPARATE_MEMBER_PAGES tag is set to YES then Doxygen will produce a new
# page for each member. If set to NO, the documentation of a member will be part
# of the file/class/namespace that contains it.
# The default value is: NO.
```

```
SEPARATE_MEMBER_PAGES = NO
```

```
# The TAB_SIZE tag can be used to set the number of spaces in a tab. Doxygen
# uses this value to replace tabs by spaces in code fragments.
# Minimum value: 1, maximum value: 16, default value: 4.
```

```
TAB_SIZE = 4
```

```
# This tag can be used to specify a number of aliases that act as commands in
# the documentation. An alias has the form:
# name=value
# For example adding
# "sideeffect=@par Side Effects:^^"
# will allow you to put the command \sideeffect (or @sideeffect) in the
# documentation, which will result in a user-defined paragraph with heading
# "Side Effects:". Note that you cannot put \n's in the value part of an alias
# to insert newlines (in the resulting output). You can put ^^ in the value part
# of an alias to insert a newline as if a physical newline was in the original
# file. When you need a literal { or } or , in the value part of an alias you
# have to escape them by means of a backslash (\), this can lead to conflicts
# with the commands \{ and \} for these it is advised to use the version @{ and
# @} or use a double escape (\{ and \})
```

```
ALIASES =
```

```
# Set the OPTIMIZE_OUTPUT_FOR_C tag to YES if your project consists of C sources
# only. Doxygen will then generate output that is more tailored for C. For
# instance, some of the names that are used will be different. The list of all
# members will be omitted, etc.
# The default value is: NO.
```

```
OPTIMIZE_OUTPUT_FOR_C = NO
```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІПЗ	Арк.
		Власенко О.В.				180
Змн.	Арк.	№ докум.	Підпис	Дата		

```

# Set the OPTIMIZE_OUTPUT_JAVA tag to YES if your project consists of Java or
# Python sources only. Doxygen will then generate output that is more tailored
# for that language. For instance, namespaces will be presented as packages,
# qualified scopes will look different, etc.
# The default value is: NO.

OPTIMIZE_OUTPUT_JAVA    = NO

# Set the OPTIMIZE_FOR_FORTRAN tag to YES if your project consists of Fortran
# sources. Doxygen will then generate output that is tailored for Fortran.
# The default value is: NO.

OPTIMIZE_FOR_FORTRAN    = NO

# Set the OPTIMIZE_OUTPUT_VHDL tag to YES if your project consists of VHDL
# sources. Doxygen will then generate output that is tailored for VHDL.
# The default value is: NO.

OPTIMIZE_OUTPUT_VHDL    = NO

# Set the OPTIMIZE_OUTPUT_SLICE tag to YES if your project consists of Slice
# sources only. Doxygen will then generate output that is more tailored for that
# language. For instance, namespaces will be presented as modules, types will be
# separated into more groups, etc.
# The default value is: NO.

OPTIMIZE_OUTPUT_SLICE   = NO

# Doxygen selects the parser to use depending on the extension of the files it
# parses. With this tag you can assign which parser to use for a given
# extension. Doxygen has a built-in mapping, but you can override or extend it
# using this tag. The format is ext=language, where ext is a file extension, and
# language is one of the parsers supported by Doxygen: IDL, Java, JavaScript,
# Csharp (C#), C, C++, Lex, D, PHP, md (Markdown), Objective-C, Python, Slice,
# VHDL, Fortran (fixed format Fortran: FortranFixed, free formatted Fortran:
# FortranFree, unknown formatted Fortran: Fortran. In the later case the parser
# tries to guess whether the code is fixed or free formatted code, this is the
# default for Fortran type files). For instance to make Doxygen treat .inc files
# as Fortran files (default is PHP), and .f files as C (default is Fortran),
# use: inc=Fortran f=C.
#
# Note: For files without extension you can use no_extension as a placeholder.
#
# Note that for custom extensions you also need to set FILE_PATTERNS otherwise
# the files are not read by Doxygen. When specifying no_extension you should add
# * to the FILE_PATTERNS.
#
# Note see also the list of default file extension mappings.

EXTENSION_MAPPING        =

# If the MARKDOWN_SUPPORT tag is enabled then Doxygen pre-processes all comments
# according to the Markdown format, which allows for more readable
# documentation. See https://daringfireball.net/projects/markdown/ for details.
# The output of markdown processing is further processed by Doxygen, so you can
# mix Doxygen, HTML, and XML commands with Markdown formatting. Disable only in
# case of backward compatibilities issues.
# The default value is: YES.

MARKDOWN_SUPPORT         = YES

# When the TOC_INCLUDE_HEADINGS tag is set to a non-zero value, all headings up

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		181

```

# to that level are automatically included in the table of contents, even if
# they do not have an id attribute.
# Note: This feature currently applies only to Markdown headings.
# Minimum value: 0, maximum value: 99, default value: 6.
# This tag requires that the tag MARKDOWN_SUPPORT is set to YES.

TOC_INCLUDE_HEADINGS    = 6

# The MARKDOWN_ID_STYLE tag can be used to specify the algorithm used to
# generate identifiers for the Markdown headings. Note: Every identifier is
# unique.
# Possible values are: DOXYGEN use a fixed 'autotoc_md' string followed by a
# sequence number starting at 0 and GITHUB use the lower case version of title
# with any whitespace replaced by '-' and punctuation characters removed.
# The default value is: DOXYGEN.
# This tag requires that the tag MARKDOWN_SUPPORT is set to YES.

MARKDOWN_ID_STYLE       = DOXYGEN

# When enabled Doxygen tries to link words that correspond to documented
# classes, or namespaces to their corresponding documentation. Such a link can
# be prevented in individual cases by putting a % sign in front of the word or
# globally by setting AUTOLINK_SUPPORT to NO. Words listed in the
# AUTOLINK_IGNORE_WORDS tag are excluded from automatic linking.
# The default value is: YES.

AUTOLINK_SUPPORT        = YES

# This tag specifies a list of words that, when matching the start of a word in
# the documentation, will suppress auto links generation, if it is enabled via
# AUTOLINK_SUPPORT. This list does not affect affect links explicitly created
# using \# or the \link or commands.
# This tag requires that the tag AUTOLINK_SUPPORT is set to YES.

AUTOLINK_IGNORE_WORDS   =

# If you use STL classes (i.e. std::string, std::vector, etc.) but do not want
# to include (a tag file for) the STL sources as input, then you should set this
# tag to YES in order to let Doxygen match functions declarations and
# definitions whose arguments contain STL classes (e.g. func(std::string);
# versus func(std::string) {}). This also makes the inheritance and
# collaboration diagrams that involve STL classes more complete and accurate.
# The default value is: NO.

BUILTIN_STL_SUPPORT     = NO

# If you use Microsoft's C++/CLI language, you should set this option to YES to
# enable parsing support.
# The default value is: NO.

CPP_CLI_SUPPORT         = NO

# Set the SIP_SUPPORT tag to YES if your project consists of sip (see:
# https://www.riverbankcomputing.com/software) sources only. Doxygen will parse
# them like normal C++ but will assume all classes use public instead of private
# inheritance when no explicit protection keyword is present.
# The default value is: NO.

SIP_SUPPORT             = NO

# For Microsoft's IDL there are propget and propput attributes to indicate
# getter and setter methods for a property. Setting this option to YES will make

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		182

```
# Doxygen to replace the get and set methods by a property in the documentation.
# This will only work if the methods are indeed getting or setting a simple
# type. If this is not the case, or you want to show the methods anyway, you
# should set this option to NO.
# The default value is: YES.
```

```
IDL_PROPERTY_SUPPORT    = YES
```

```
# If member grouping is used in the documentation and the DISTRIBUTE_GROUP_DOC
# tag is set to YES then Doxygen will reuse the documentation of the first
# member in the group (if any) for the other members of the group. By default
# all members of a group must be documented explicitly.
# The default value is: NO.
```

```
DISTRIBUTE_GROUP_DOC    = NO
```

```
# If one adds a struct or class to a group and this option is enabled, then also
# any nested class or struct is added to the same group. By default this option
# is disabled and one has to add nested compounds explicitly via \ingroup.
# The default value is: NO.
```

```
GROUP_NESTED_COMPOUNDS = NO
```

```
# Set the SUBGROUPING tag to YES to allow class member groups of the same type
# (for instance a group of public functions) to be put as a subgroup of that
# type (e.g. under the Public Functions section). Set it to NO to prevent
# subgrouping. Alternatively, this can be done per class using the
# \nosubgrouping command.
# The default value is: YES.
```

```
SUBGROUPING             = YES
```

```
# When the INLINE_GROUPED_CLASSES tag is set to YES, classes, structs and unions
# are shown inside the group in which they are included (e.g. using \ingroup)
# instead of on a separate page (for HTML and Man pages) or section (for LaTeX
# and RTF).
#
# Note that this feature does not work in combination with
# SEPARATE_MEMBER_PAGES.
# The default value is: NO.
```

```
INLINE_GROUPED_CLASSES = NO
```

```
# When the INLINE_SIMPLE_STRUCTS tag is set to YES, structs, classes, and unions
# with only public data fields or simple typedef fields will be shown inline in
# the documentation of the scope in which they are defined (i.e. file,
# namespace, or group documentation), provided this scope is documented. If set
# to NO, structs, classes, and unions are shown on a separate page (for HTML and
# Man pages) or section (for LaTeX and RTF).
# The default value is: NO.
```

```
INLINE_SIMPLE_STRUCTS   = NO
```

```
# When TYPEDEF_HIDES_STRUCT tag is enabled, a typedef of a struct, union, or
# enum is documented as struct, union, or enum with the name of the typedef. So
# typedef struct TypeS {} TypeT, will appear in the documentation as a struct
# with name TypeT. When disabled the typedef will appear as a member of a file,
# namespace, or class. And the struct will be named TypeS. This can typically be
# useful for C code in case the coding convention dictates that all compound
# types are typedef'ed and only the typedef is referenced, never the tag name.
# The default value is: NO.
```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		183

```

TYPEDEF_HIDES_STRUCT    = NO

# The size of the symbol lookup cache can be set using LOOKUP_CACHE_SIZE. This
# cache is used to resolve symbols given their name and scope. Since this can be
# an expensive process and often the same symbol appears multiple times in the
# code, Doxygen keeps a cache of pre-resolved symbols. If the cache is too small
# Doxygen will become slower. If the cache is too large, memory is wasted. The
# cache size is given by this formula: 2^(16+LOOKUP_CACHE_SIZE). The valid range
# is 0..9, the default is 0, corresponding to a cache size of 2^16=65536
# symbols. At the end of a run Doxygen will report the cache usage and suggest
# the optimal cache size from a speed point of view.
# Minimum value: 0, maximum value: 9, default value: 0.

LOOKUP_CACHE_SIZE       = 0

# The NUM_PROC_THREADS specifies the number of threads Doxygen is allowed to use
# during processing. When set to 0 Doxygen will based this on the number of
# cores available in the system. You can set it explicitly to a value larger
# than 0 to get more control over the balance between CPU load and processing
# speed. At this moment only the input processing can be done using multiple
# threads. Since this is still an experimental feature the default is set to 1,
# which effectively disables parallel processing. Please report any issues you
# encounter. Generating dot graphs in parallel is controlled by the
# DOT_NUM_THREADS setting.
# Minimum value: 0, maximum value: 32, default value: 1.

NUM_PROC_THREADS        = 1

# If the TIMESTAMP tag is set different from NO then each generated page will
# contain the date or date and time when the page was generated. Setting this to
# NO can help when comparing the output of multiple runs.
# Possible values are: YES, NO, DATETIME and DATE.
# The default value is: NO.

TIMESTAMP               = NO

#-----
# Build related configuration options
#-----

# If the EXTRACT_ALL tag is set to YES, Doxygen will assume all entities in
# documentation are documented, even if no documentation was available. Private
# class members and static file members will be hidden unless the
# EXTRACT_PRIVATE respectively EXTRACT_STATIC tags are set to YES.
# Note: This will also disable the warnings about undocumented members that are
# normally produced when WARNINGS is set to YES.
# The default value is: NO.

EXTRACT_ALL             = NO

# If the EXTRACT_PRIVATE tag is set to YES, all private members of a class will
# be included in the documentation.
# The default value is: NO.

EXTRACT_PRIVATE         = NO

# If the EXTRACT_PRIV_VIRTUAL tag is set to YES, documented private virtual
# methods of a class will be included in the documentation.
# The default value is: NO.

EXTRACT_PRIV_VIRTUAL    = NO

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		184

```

# If the EXTRACT_PACKAGE tag is set to YES, all members with package or internal
# scope will be included in the documentation.
# The default value is: NO.

EXTRACT_PACKAGE          = NO

# If the EXTRACT_STATIC tag is set to YES, all static members of a file will be
# included in the documentation.
# The default value is: NO.

EXTRACT_STATIC           = NO

# If the EXTRACT_LOCAL_CLASSES tag is set to YES, classes (and structs) defined
# locally in source files will be included in the documentation. If set to NO,
# only classes defined in header files are included. Does not have any effect
# for Java sources.
# The default value is: YES.

EXTRACT_LOCAL_CLASSES    = YES

# This flag is only useful for Objective-C code. If set to YES, local methods,
# which are defined in the implementation section but not in the interface are
# included in the documentation. If set to NO, only methods in the interface are
# included.
# The default value is: NO.

EXTRACT_LOCAL_METHODS    = NO

# If this flag is set to YES, the members of anonymous namespaces will be
# extracted and appear in the documentation as a namespace called
# 'anonymous_namespace{file}', where file will be replaced with the base name of
# the file that contains the anonymous namespace. By default anonymous namespace
# are hidden.
# The default value is: NO.

EXTRACT_ANON_NSPPACES    = NO

# If this flag is set to YES, the name of an unnamed parameter in a declaration
# will be determined by the corresponding definition. By default unnamed
# parameters remain unnamed in the output.
# The default value is: YES.

RESOLVE_UNNAMED_PARAMS   = YES

# If the HIDE_UNDOC_MEMBERS tag is set to YES, Doxygen will hide all
# undocumented members inside documented classes or files. If set to NO these
# members will be included in the various overviews, but no documentation
# section is generated. This option has no effect if EXTRACT_ALL is enabled.
# The default value is: NO.

HIDE_UNDOC_MEMBERS       = NO

# If the HIDE_UNDOC_CLASSES tag is set to YES, Doxygen will hide all
# undocumented classes that are normally visible in the class hierarchy. If set
# to NO, these classes will be included in the various overviews. This option
# will also hide undocumented C++ concepts if enabled. This option has no effect
# if EXTRACT_ALL is enabled.
# The default value is: NO.

HIDE_UNDOC_CLASSES       = NO

# If the HIDE_UNDOC_NAMESPACES tag is set to YES, Doxygen will hide all

```

		Бабушко А.С.			ІІЗ.КР.Б.-121-25-ІІЗ	Арк.
		Власенко О.В.				185
Змн.	Арк.	№ докум.	Підпис	Дата		

```
# undocumented namespaces that are normally visible in the namespace hierarchy.
# If set to NO, these namespaces will be included in the various overviews. This
# option has no effect if EXTRACT_ALL is enabled.
# The default value is: YES.
```

```
HIDE_UNDOC_NAMESPACES = YES
```

```
# If the HIDE_FRIEND_COMPOUNDS tag is set to YES, Doxygen will hide all friend
# declarations. If set to NO, these declarations will be included in the
# documentation.
# The default value is: NO.
```

```
HIDE_FRIEND_COMPOUNDS = NO
```

```
# If the HIDE_IN_BODY_DOCS tag is set to YES, Doxygen will hide any
# documentation blocks found inside the body of a function. If set to NO, these
# blocks will be appended to the function's detailed documentation block.
# The default value is: NO.
```

```
HIDE_IN_BODY_DOCS = NO
```

```
# The INTERNAL_DOCS tag determines if documentation that is typed after a
# \internal command is included. If the tag is set to NO then the documentation
# will be excluded. Set it to YES to include the internal documentation.
# The default value is: NO.
```

```
INTERNAL_DOCS = NO
```

```
# With the correct setting of option CASE_SENSE_NAMES Doxygen will better be
# able to match the capabilities of the underlying filesystem. In case the
# filesystem is case sensitive (i.e. it supports files in the same directory
# whose names only differ in casing), the option must be set to YES to properly
# deal with such files in case they appear in the input. For filesystems that
# are not case sensitive the option should be set to NO to properly deal with
# output files written for symbols that only differ in casing, such as for two
# classes, one named CLASS and the other named Class, and to also support
# references to files without having to specify the exact matching casing. On
# Windows (including Cygwin) and macOS, users should typically set this option
# to NO, whereas on Linux or other Unix flavors it should typically be set to
# YES.
# Possible values are: SYSTEM, NO and YES.
# The default value is: SYSTEM.
```

```
CASE_SENSE_NAMES = SYSTEM
```

```
# If the HIDE_SCOPE_NAMES tag is set to NO then Doxygen will show members with
# their full class and namespace scopes in the documentation. If set to YES, the
# scope will be hidden.
# The default value is: NO.
```

```
HIDE_SCOPE_NAMES = NO
```

```
# If the HIDE_COMPOUND_REFERENCE tag is set to NO (default) then Doxygen will
# append additional text to a page's title, such as Class Reference. If set to
# YES the compound reference will be hidden.
# The default value is: NO.
```

```
HIDE_COMPOUND_REFERENCE= NO
```

```
# If the SHOW_HEADERFILE tag is set to YES then the documentation for a class
# will show which file needs to be included to use the class.
# The default value is: YES.
```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		186

SHOW_HEADERFILE = YES

If the SHOW_INCLUDE_FILES tag is set to YES then Doxygen will put a list of
the files that are included by a file in the documentation of that file.
The default value is: YES.

SHOW_INCLUDE_FILES = YES

If the SHOW_GROUPED_MEMB_INC tag is set to YES then Doxygen will add for each
grouped member an include statement to the documentation, telling the reader
which file to include in order to use the member.
The default value is: NO.

SHOW_GROUPED_MEMB_INC = NO

If the FORCE_LOCAL_INCLUDES tag is set to YES then Doxygen will list include
files with double quotes in the documentation rather than with sharp brackets.
The default value is: NO.

FORCE_LOCAL_INCLUDES = NO

If the INLINE_INFO tag is set to YES then a tag [inline] is inserted in the
documentation for inline members.
The default value is: YES.

INLINE_INFO = YES

If the SORT_MEMBER_DOCS tag is set to YES then Doxygen will sort the
(detailed) documentation of file and class members alphabetically by member
name. If set to NO, the members will appear in declaration order.
The default value is: YES.

SORT_MEMBER_DOCS = YES

If the SORT_BRIEF_DOCS tag is set to YES then Doxygen will sort the brief
descriptions of file, namespace and class members alphabetically by member
name. If set to NO, the members will appear in declaration order. Note that
this will also influence the order of the classes in the class list.
The default value is: NO.

SORT_BRIEF_DOCS = NO

If the SORT_MEMBERS_CTORS_1ST tag is set to YES then Doxygen will sort the
(brief and detailed) documentation of class members so that constructors and
destructors are listed first. If set to NO the constructors will appear in the
respective orders defined by SORT_BRIEF_DOCS and SORT_MEMBER_DOCS.
Note: If SORT_BRIEF_DOCS is set to NO this option is ignored for sorting brief
member documentation.
Note: If SORT_MEMBER_DOCS is set to NO this option is ignored for sorting
detailed member documentation.
The default value is: NO.

SORT_MEMBERS_CTORS_1ST = NO

If the SORT_GROUP_NAMES tag is set to YES then Doxygen will sort the hierarchy
of group names into alphabetical order. If set to NO the group names will
appear in their defined order.
The default value is: NO.

SORT_GROUP_NAMES = NO

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		187

If the SORT_BY_SCOPE_NAME tag is set to YES, the class list will be sorted by
 # fully-qualified names, including namespaces. If set to NO, the class list will
 # be sorted only by class name, not including the namespace part.
 # Note: This option is not very useful if HIDE_SCOPE_NAMES is set to YES.
 # Note: This option applies only to the class list, not to the alphabetical
 # list.
 # The default value is: NO.

SORT_BY_SCOPE_NAME = NO

If the STRICT_PROTO_MATCHING option is enabled and Doxygen fails to do proper
 # type resolution of all parameters of a function it will reject a match between
 # the prototype and the implementation of a member function even if there is
 # only one candidate or it is obvious which candidate to choose by doing a
 # simple string match. By disabling STRICT_PROTO_MATCHING Doxygen will still
 # accept a match between prototype and implementation in such cases.
 # The default value is: NO.

STRICT_PROTO_MATCHING = NO

The GENERATE_TODOLIST tag can be used to enable (YES) or disable (NO) the todo
 # list. This list is created by putting \todo commands in the documentation.
 # The default value is: YES.

GENERATE_TODOLIST = YES

The GENERATE_TESTLIST tag can be used to enable (YES) or disable (NO) the test
 # list. This list is created by putting \test commands in the documentation.
 # The default value is: YES.

GENERATE_TESTLIST = YES

The GENERATE_BUGLIST tag can be used to enable (YES) or disable (NO) the bug
 # list. This list is created by putting \bug commands in the documentation.
 # The default value is: YES.

GENERATE_BUGLIST = YES

The GENERATE_DEPRECATEDLIST tag can be used to enable (YES) or disable (NO)
 # the deprecated list. This list is created by putting \deprecated commands in
 # the documentation.
 # The default value is: YES.

GENERATE_DEPRECATEDLIST= YES

The ENABLED_SECTIONS tag can be used to enable conditional documentation
 # sections, marked by \if <section_label> ... \endif and \cond <section_label>
 # ... \endcond blocks.

ENABLED_SECTIONS =

The MAX_INITIALIZER_LINES tag determines the maximum number of lines that the
 # initial value of a variable or macro / define can have for it to appear in the
 # documentation. If the initializer consists of more lines than specified here
 # it will be hidden. Use a value of 0 to hide initializers completely. The
 # appearance of the value of individual variables and macros / defines can be
 # controlled using \showinitializer or \hideinitializer command in the
 # documentation regardless of this setting.
 # Minimum value: 0, maximum value: 10000, default value: 30.

MAX_INITIALIZER_LINES = 30

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		188

```

# Set the SHOW_USED_FILES tag to NO to disable the list of files generated at
# the bottom of the documentation of classes and structs. If set to YES, the
# list will mention the files that were used to generate the documentation.
# The default value is: YES.

SHOW_USED_FILES          = YES

# Set the SHOW_FILES tag to NO to disable the generation of the Files page. This
# will remove the Files entry from the Quick Index and from the Folder Tree View
# (if specified).
# The default value is: YES.

SHOW_FILES                = YES

# Set the SHOW_NAMESPACES tag to NO to disable the generation of the Namespaces
# page. This will remove the Namespaces entry from the Quick Index and from the
# Folder Tree View (if specified).
# The default value is: YES.

SHOW_NAMESPACES          = YES

# The FILE_VERSION_FILTER tag can be used to specify a program or script that
# Doxygen should invoke to get the current version for each file (typically from
# the version control system). Doxygen will invoke the program by executing (via
# popen()) the command command input-file, where command is the value of the
# FILE_VERSION_FILTER tag, and input-file is the name of an input file provided
# by Doxygen. Whatever the program writes to standard output is used as the file
# version. For an example see the documentation.

FILE_VERSION_FILTER      =

# The LAYOUT_FILE tag can be used to specify a layout file which will be parsed
# by Doxygen. The layout file controls the global structure of the generated
# output files in an output format independent way. To create the layout file
# that represents Doxygen's defaults, run Doxygen with the -l option. You can
# optionally specify a file name after the option, if omitted DoxygenLayout.xml
# will be used as the name of the layout file. See also section "Changing the
# layout of pages" for information.
#
# Note that if you run Doxygen from a directory containing a file called
# DoxygenLayout.xml, Doxygen will parse it automatically even if the LAYOUT_FILE
# tag is left empty.

LAYOUT_FILE               =

# The CITE_BIB_FILES tag can be used to specify one or more bib files containing
# the reference definitions. This must be a list of .bib files. The .bib
# extension is automatically appended if omitted. This requires the bibtex tool
# to be installed. See also https://en.wikipedia.org/wiki/BibTeX for more info.
# For LaTeX the style of the bibliography can be controlled using
# LATEX_BIB_STYLE. To use this feature you need bibtex and perl available in the
# search path. See also \cite for info how to create references.

CITE_BIB_FILES            =

# The EXTERNAL_TOOL_PATH tag can be used to extend the search path (PATH
# environment variable) so that external tools such as latex and gs can be
# found.
# Note: Directories specified with EXTERNAL_TOOL_PATH are added in front of the
# path already specified by the PATH variable, and are added in the order
# specified.
# Note: This option is particularly useful for macOS version 14 (Sonoma) and

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				189
Змн.	Арк.	№ докум.	Підпис	Дата		

```
# higher, when running Doxygen from Doxywizard, because in this case any user-
# defined changes to the PATH are ignored. A typical example on macOS is to set
# EXTERNAL_TOOL_PATH = /Library/TeX/texbin /usr/local/bin
# together with the standard path, the full search path used by doxygen when
# launching external tools will then become
# PATH=/Library/TeX/texbin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin
```

```
EXTERNAL_TOOL_PATH      =
```

```
#-----
# Configuration options related to warning and progress messages
#-----
```

```
# The QUIET tag can be used to turn on/off the messages that are generated to
# standard output by Doxygen. If QUIET is set to YES this implies that the
# messages are off.
# The default value is: NO.
```

```
QUIET                    = NO
```

```
# The WARNINGS tag can be used to turn on/off the warning messages that are
# generated to standard error (stderr) by Doxygen. If WARNINGS is set to YES
# this implies that the warnings are on.
```

```
#
# Tip: Turn warnings on while writing the documentation.
# The default value is: YES.
```

```
WARNINGS                 = YES
```

```
# If the WARN_IF_UNDOCUMENTED tag is set to YES then Doxygen will generate
# warnings for undocumented members. If EXTRACT_ALL is set to YES then this flag
# will automatically be disabled.
# The default value is: YES.
```

```
WARN_IF_UNDOCUMENTED    = YES
```

```
# If the WARN_IF_DOC_ERROR tag is set to YES, Doxygen will generate warnings for
# potential errors in the documentation, such as documenting some parameters in
# a documented function twice, or documenting parameters that don't exist or
# using markup commands wrongly.
# The default value is: YES.
```

```
WARN_IF_DOC_ERROR       = YES
```

```
# If WARN_IF_INCOMPLETE_DOC is set to YES, Doxygen will warn about incomplete
# function parameter documentation. If set to NO, Doxygen will accept that some
# parameters have no documentation without warning.
# The default value is: YES.
```

```
WARN_IF_INCOMPLETE_DOC = YES
```

```
# This WARN_NO_PARAMDOC option can be enabled to get warnings for functions that
# are documented, but have no documentation for their parameters or return
# value. If set to NO, Doxygen will only warn about wrong parameter
# documentation, but not about the absence of documentation. If EXTRACT_ALL is
# set to YES then this flag will automatically be disabled. See also
# WARN_IF_INCOMPLETE_DOC
# The default value is: NO.
```

```
WARN_NO_PARAMDOC        = NO
```

```
# If WARN_IF_UNDOC_ENUM_VAL option is set to YES, Doxygen will warn about
```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		190

```

# undocumented enumeration values. If set to NO, Doxygen will accept
# undocumented enumeration values. If EXTRACT_ALL is set to YES then this flag
# will automatically be disabled.
# The default value is: NO.

WARN_IF_UNDOC_ENUM_VAL = NO

# If WARN_LAYOUT_FILE option is set to YES, Doxygen will warn about issues found
# while parsing the user defined layout file, such as missing or wrong elements.
# See also LAYOUT_FILE for details. If set to NO, problems with the layout file
# will be suppressed.
# The default value is: YES.

WARN_LAYOUT_FILE          = YES

# If the WARN_AS_ERROR tag is set to YES then Doxygen will immediately stop when
# a warning is encountered. If the WARN_AS_ERROR tag is set to FAIL_ON_WARNINGS
# then Doxygen will continue running as if WARN_AS_ERROR tag is set to NO, but
# at the end of the Doxygen process Doxygen will return with a non-zero status.
# If the WARN_AS_ERROR tag is set to FAIL_ON_WARNINGS_PRINT then Doxygen behaves
# like FAIL_ON_WARNINGS but in case no WARN_LOGFILE is defined Doxygen will not
# write the warning messages in between other messages but write them at the end
# of a run, in case a WARN_LOGFILE is defined the warning messages will be
# besides being in the defined file also be shown at the end of a run, unless
# the WARN_LOGFILE is defined as - i.e. standard output (stdout) in that case
# the behavior will remain as with the setting FAIL_ON_WARNINGS.
# Possible values are: NO, YES, FAIL_ON_WARNINGS and FAIL_ON_WARNINGS_PRINT.
# The default value is: NO.

WARN_AS_ERROR              = NO

# The WARN_FORMAT tag determines the format of the warning messages that Doxygen
# can produce. The string should contain the $file, $line, and $text tags, which
# will be replaced by the file and line number from which the warning originated
# and the warning text. Optionally the format may contain $version, which will
# be replaced by the version of the file (if it could be obtained via
# FILE_VERSION_FILTER)
# See also: WARN_LINE_FORMAT
# The default value is: $file:$line: $text.

WARN_FORMAT                = "$file:$line: $text"

# In the $text part of the WARN_FORMAT command it is possible that a reference
# to a more specific place is given. To make it easier to jump to this place
# (outside of Doxygen) the user can define a custom "cut" / "paste" string.
# Example:
# WARN_LINE_FORMAT = "'vi $file +$line'"
# See also: WARN_FORMAT
# The default value is: at line $line of file $file.

WARN_LINE_FORMAT           = "at line $line of file $file"

# The WARN_LOGFILE tag can be used to specify a file to which warning and error
# messages should be written. If left blank the output is written to standard
# error (stderr). In case the file specified cannot be opened for writing the
# warning and error messages are written to standard error. When as file - is
# specified the warning and error messages are written to standard output
# (stdout).

WARN_LOGFILE               =

#-----

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІІЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		191

```

# Configuration options related to the input files
#-----

# The INPUT tag is used to specify the files and/or directories that contain
# documented source files. You may enter file names like myfile.cpp or
# directories like /usr/src/myproject. Separate the files or directories with
# spaces. See also FILE_PATTERNS and EXTENSION_MAPPING
# Note: If this tag is empty the current directory is searched.

INPUT                                = include src

# This tag can be used to specify the character encoding of the source files
# that Doxygen parses. Internally Doxygen uses the UTF-8 encoding. Doxygen uses
# libiconv (or the iconv built into libc) for the transcoding. See the libiconv
# documentation (see:
# https://www.gnu.org/software/libiconv/) for the list of possible encodings.
# See also: INPUT_FILE_ENCODING
# The default value is: UTF-8.

INPUT_ENCODING                      = UTF-8

# This tag can be used to specify the character encoding of the source files
# that Doxygen parses. The INPUT_FILE_ENCODING tag can be used to specify
# character encoding on a per file pattern basis. Doxygen will compare the file
# name with each pattern and apply the encoding instead of the default
# INPUT_ENCODING if there is a match. The character encodings are a list of the
# form: pattern=encoding (like *.php=ISO-8859-1).
# See also: INPUT_ENCODING for further information on supported encodings.

INPUT_FILE_ENCODING                 =

# If the value of the INPUT tag contains directories, you can use the
# FILE_PATTERNS tag to specify one or more wildcard patterns (like *.cpp and
# *.h) to filter out the source-files in the directories.
#
# Note that for custom extensions or not directly supported extensions you also
# need to set EXTENSION_MAPPING for the extension otherwise the files are not
# read by Doxygen.
#
# Note the list of default checked file patterns might differ from the list of
# default file extension mappings.
#
# If left blank the following patterns are tested:*.c, *.cc, *.cxx, *.cxxm,
# *.cpp, *.cppm, *.ccm, *.c++, *.c++m, *.java, *.ii, *.ixx, *.ipp, *.i++, *.inl,
# *.idl, *.ddl, *.odl, *.h, *.hh, *.hxx, *.hpp, *.h++, *.ixx, *.l, *.cs, *.d,
# *.php, *.php4, *.php5, *.phtml, *.inc, *.m, *.markdown, *.md, *.mm, *.dox (to
# be provided as Doxygen C comment), *.py, *.pyw, *.f90, *.f95, *.f03, *.f08,
# *.f18, *.f, *.for, *.vhd, *.vhdl, *.ucf, *.qsf and *.ice.

FILE_PATTERNS                       = *.c \
                                      *.cc \
                                      *.cxx \
                                      *.cxxm \
                                      *.cpp \
                                      *.cppm \
                                      *.ccm \
                                      *.c++ \
                                      *.c++m \
                                      *.java \
                                      *.ii \
                                      *.ixx \
                                      *.ipp \

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		192

```

*.i++ \
*.inl \
*.idl \
*.ddl \
*.odl \
*.h \
*.hh \
*.hxx \
*.hpp \
*.h++ \
*.ixx \
*.l \
*.cs \
*.d \
*.php \
*.php4 \
*.php5 \
*.phtml \
*.inc \
*.m \
*.markdown \
*.md \
*.mm \
*.dox \
*.py \
*.pyw \
*.f90 \
*.f95 \
*.f03 \
*.f08 \
*.f18 \
*.f \
*.for \
*.vhd \
*.vhdl \
*.ucf \
*.qsf \
*.ice

```

```

# The RECURSIVE tag can be used to specify whether or not subdirectories should
# be searched for input files as well.
# The default value is: NO.

```

```

RECURSIVE          = YES

```

```

# The EXCLUDE tag can be used to specify files and/or directories that should be
# excluded from the INPUT source files. This way you can easily exclude a
# subdirectory from a directory tree whose root is specified with the INPUT tag.
#
# Note that relative paths are relative to the directory from which Doxygen is
# run.

```

```

EXCLUDE            =

```

```

# The EXCLUDE_SYMLINKS tag can be used to select whether or not files or
# directories that are symbolic links (a Unix file system feature) are excluded
# from the input.
# The default value is: NO.

```

```

EXCLUDE_SYMLINKS   = NO

```

```

# If the value of the INPUT tag contains directories, you can use the

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				193
Змн.	Арк.	№ докум.	Підпис	Дата		

```

# EXCLUDE_PATTERNS tag to specify one or more wildcard patterns to exclude
# certain files from those directories.
#
# Note that the wildcards are matched against the file with absolute path, so to
# exclude all test directories for example use the pattern */test/*

EXCLUDE_PATTERNS      =

# The EXCLUDE_SYMBOLS tag can be used to specify one or more symbol names
# (namespaces, classes, functions, etc.) that should be excluded from the
# output. The symbol name can be a fully qualified name, a word, or if the
# wildcard * is used, a substring. Examples: ANamespace, AClass,
# ANamespace::AClass, ANamespace::*Test

EXCLUDE_SYMBOLS       =

# The EXAMPLE_PATH tag can be used to specify one or more files or directories
# that contain example code fragments that are included (see the \include
# command).

EXAMPLE_PATH          =

# If the value of the EXAMPLE_PATH tag contains directories, you can use the
# EXAMPLE_PATTERNS tag to specify one or more wildcard pattern (like *.cpp and
# *.h) to filter out the source-files in the directories. If left blank all
# files are included.

EXAMPLE_PATTERNS      = *

# If the EXAMPLE_RECURSIVE tag is set to YES then subdirectories will be
# searched for input files to be used with the \include or \dontinclude commands
# irrespective of the value of the RECURSIVE tag.
# The default value is: NO.

EXAMPLE_RECURSIVE     = NO

# The IMAGE_PATH tag can be used to specify one or more files or directories
# that contain images that are to be included in the documentation (see the
# \image command).

IMAGE_PATH            =

# The INPUT_FILTER tag can be used to specify a program that Doxygen should
# invoke to filter for each input file. Doxygen will invoke the filter program
# by executing (via popen()) the command:
#
# <filter> <input-file>
#
# where <filter> is the value of the INPUT_FILTER tag, and <input-file> is the
# name of an input file. Doxygen will then use the output that the filter
# program writes to standard output. If FILTER_PATTERNS is specified, this tag
# will be ignored.
#
# Note that the filter must not add or remove lines; it is applied before the
# code is scanned, but not when the output code is generated. If lines are added
# or removed, the anchors will not be placed correctly.
#
# Note that Doxygen will use the data processed and written to standard output
# for further processing, therefore nothing else, like debug statements or used
# commands (so in case of a Windows batch file always use @echo OFF), should be
# written to standard output.
#

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІПЗ	Арк.
		Власенко О.В.				194
Змн.	Арк.	№ докум.	Підпис	Дата		

```
# Note that for custom extensions or not directly supported extensions you also
# need to set EXTENSION_MAPPING for the extension otherwise the files are not
# properly processed by Doxygen.
```

```
INPUT_FILTER          =
```

```
# The FILTER_PATTERNS tag can be used to specify filters on a per file pattern
# basis. Doxygen will compare the file name with each pattern and apply the
# filter if there is a match. The filters are a list of the form: pattern=filter
# (like *.cpp=my_cpp_filter). See INPUT_FILTER for further information on how
# filters are used. If the FILTER_PATTERNS tag is empty or if none of the
# patterns match the file name, INPUT_FILTER is applied.
```

```
#
# Note that for custom extensions or not directly supported extensions you also
# need to set EXTENSION_MAPPING for the extension otherwise the files are not
# properly processed by Doxygen.
```

```
FILTER_PATTERNS      =
```

```
# If the FILTER_SOURCE_FILES tag is set to YES, the input filter (if set using
# INPUT_FILTER) will also be used to filter the input files that are used for
# producing the source files to browse (i.e. when SOURCE_BROWSER is set to YES).
# The default value is: NO.
```

```
FILTER_SOURCE_FILES  = NO
```

```
# The FILTER_SOURCE_PATTERNS tag can be used to specify source filters per file
# pattern. A pattern will override the setting for FILTER_PATTERN (if any) and
# it is also possible to disable source filtering for a specific pattern using
# *.ext= (so without naming a filter).
# This tag requires that the tag FILTER_SOURCE_FILES is set to YES.
```

```
FILTER_SOURCE_PATTERNS =
```

```
# If the USE_MDFILE_AS_MAINPAGE tag refers to the name of a markdown file that
# is part of the input, its contents will be placed on the main page
# (index.html). This can be useful if you have a project on for instance GitHub
# and want to reuse the introduction page also for the Doxygen output.
```

```
USE_MDFILE_AS_MAINPAGE =
```

```
# If the IMPLICIT_DIR_DOCS tag is set to YES, any README.md file found in sub-
# directories of the project's root, is used as the documentation for that sub-
# directory, except when the README.md starts with a \dir, \page or \mainpage
# command. If set to NO, the README.md file needs to start with an explicit \dir
# command in order to be used as directory documentation.
# The default value is: YES.
```

```
IMPLICIT_DIR_DOCS    = YES
```

```
# The Fortran standard specifies that for fixed formatted Fortran code all
# characters from position 72 are to be considered as comment. A common
# extension is to allow longer lines before the automatic comment starts. The
# setting FORTRAN_COMMENT_AFTER will also make it possible that longer lines can
# be processed before the automatic comment starts.
# Minimum value: 7, maximum value: 10000, default value: 72.
```

```
FORTRAN_COMMENT_AFTER = 72
```

```
#-----
# Configuration options related to source browsing
#-----
```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				195
Змн.	Арк.	№ докум.	Підпис	Дата		

```

# If the SOURCE_BROWSER tag is set to YES then a list of source files will be
# generated. Documented entities will be cross-referenced with these sources.
#
# Note: To get rid of all source code in the generated output, make sure that
# also VERBATIM_HEADERS is set to NO.
# The default value is: NO.

SOURCE_BROWSER                = NO

# Setting the INLINE_SOURCES tag to YES will include the body of functions,
# multi-line macros, enums or list initialized variables directly into the
# documentation.
# The default value is: NO.

INLINE_SOURCES                = NO

# Setting the STRIP_CODE_COMMENTS tag to YES will instruct Doxygen to hide any
# special comment blocks from generated source code fragments. Normal C, C++ and
# Fortran comments will always remain visible.
# The default value is: YES.

STRIP_CODE_COMMENTS          = YES

# If the REFERENCED_BY_RELATION tag is set to YES then for each documented
# entity all documented functions referencing it will be listed.
# The default value is: NO.

REFERENCED_BY_RELATION       = NO

# If the REFERENCES_RELATION tag is set to YES then for each documented function
# all documented entities called/used by that function will be listed.
# The default value is: NO.

REFERENCES_RELATION          = NO

# If the REFERENCES_LINK_SOURCE tag is set to YES and SOURCE_BROWSER tag is set
# to YES then the hyperlinks from functions in REFERENCES_RELATION and
# REFERENCED_BY_RELATION lists will link to the source code. Otherwise they will
# link to the documentation.
# The default value is: YES.

REFERENCES_LINK_SOURCE       = YES

# If SOURCE_TOOLTIPS is enabled (the default) then hovering a hyperlink in the
# source code will show a tooltip with additional information such as prototype,
# brief description and links to the definition and documentation. Since this
# will make the HTML file larger and loading of large files a bit slower, you
# can opt to disable this feature.
# The default value is: YES.
# This tag requires that the tag SOURCE_BROWSER is set to YES.

SOURCE_TOOLTIPS              = YES

# If the USE_HTAGS tag is set to YES then the references to source code will
# point to the HTML generated by the htags(1) tool instead of Doxygen built-in
# source browser. The htags tool is part of GNU's global source tagging system
# (see https://www.gnu.org/software/global/global.html). You will need version
# 4.8.6 or higher.
#
# To use it do the following:
# - Install the latest version of global

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		196

```
# - Enable SOURCE_BROWSER and USE_HTAGS in the configuration file
# - Make sure the INPUT points to the root of the source tree
# - Run doxygen as normal
#
# Doxygen will invoke htags (and that will in turn invoke gtags), so these
# tools must be available from the command line (i.e. in the search path).
#
# The result: instead of the source browser generated by Doxygen, the links to
# source code will now point to the output of htags.
# The default value is: NO.
# This tag requires that the tag SOURCE_BROWSER is set to YES.
```

```
USE_HTAGS = NO
```

```
# If the VERBATIM_HEADERS tag is set the YES then Doxygen will generate a
# verbatim copy of the header file for each class for which an include is
# specified. Set to NO to disable this.
# See also: Section \class.
# The default value is: YES.
```

```
VERBATIM_HEADERS = YES
```

```
# If the CLANG_ASSISTED_PARSING tag is set to YES then Doxygen will use the
# clang parser (see:
# http://clang.llvm.org/) for more accurate parsing at the cost of reduced
# performance. This can be particularly helpful with template rich C++ code for
# which Doxygen's built-in parser lacks the necessary type information.
# Note: The availability of this option depends on whether or not Doxygen was
# generated with the -Duse_libclang=ON option for CMake.
# The default value is: NO.
```

```
CLANG_ASSISTED_PARSING = NO
```

```
# If the CLANG_ASSISTED_PARSING tag is set to YES and the CLANG_ADD_INC_PATHS
# tag is set to YES then Doxygen will add the directory of each input to the
# include path.
# The default value is: YES.
# This tag requires that the tag CLANG_ASSISTED_PARSING is set to YES.
```

```
CLANG_ADD_INC_PATHS = YES
```

```
# If clang assisted parsing is enabled you can provide the compiler with command
# line options that you would normally use when invoking the compiler. Note that
# the include paths will already be set by Doxygen for the files and directories
# specified with INPUT and INCLUDE_PATH.
# This tag requires that the tag CLANG_ASSISTED_PARSING is set to YES.
```

```
CLANG_OPTIONS =
```

```
# If clang assisted parsing is enabled you can provide the clang parser with the
# path to the directory containing a file called compile_commands.json. This
# file is the compilation database (see:
# http://clang.llvm.org/docs/HowToSetupToolingForLLVM.html) containing the
# options used when the source files were built. This is equivalent to
# specifying the -p option to a clang tool, such as clang-check. These options
# will then be passed to the parser. Any options specified with CLANG_OPTIONS
# will be added as well.
# Note: The availability of this option depends on whether or not Doxygen was
# generated with the -Duse_libclang=ON option for CMake.
```

```
CLANG_DATABASE_PATH =
```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		197

```

#-----
# Configuration options related to the alphabetical class index
#-----

# If the ALPHABETICAL_INDEX tag is set to YES, an alphabetical index of all
# compounds will be generated. Enable this if the project contains a lot of
# classes, structs, unions or interfaces.
# The default value is: YES.

ALPHABETICAL_INDEX      = YES

# The IGNORE_PREFIX tag can be used to specify a prefix (or a list of prefixes)
# that should be ignored while generating the index headers. The IGNORE_PREFIX
# tag works for classes, function and member names. The entity will be placed in
# the alphabetical list under the first letter of the entity name that remains
# after removing the prefix.
# This tag requires that the tag ALPHABETICAL_INDEX is set to YES.

IGNORE_PREFIX           =

#-----
# Configuration options related to the HTML output
#-----

# If the GENERATE_HTML tag is set to YES, Doxygen will generate HTML output
# The default value is: YES.

GENERATE_HTML           = YES

# The HTML_OUTPUT tag is used to specify where the HTML docs will be put. If a
# relative path is entered the value of OUTPUT_DIRECTORY will be put in front of
# it.
# The default directory is: html.
# This tag requires that the tag GENERATE_HTML is set to YES.

HTML_OUTPUT             = html

# The HTML_FILE_EXTENSION tag can be used to specify the file extension for each
# generated HTML page (for example: .htm, .php, .asp).
# The default value is: .html.
# This tag requires that the tag GENERATE_HTML is set to YES.

HTML_FILE_EXTENSION     = .html

# The HTML_HEADER tag can be used to specify a user-defined HTML header file for
# each generated HTML page. If the tag is left blank Doxygen will generate a
# standard header.
#
# To get valid HTML the header file that includes any scripts and style sheets
# that Doxygen needs, which is dependent on the configuration options used (e.g.
# the setting GENERATE_TREEVIEW). It is highly recommended to start with a
# default header using
# doxygen -w html new_header.html new_footer.html new_stylesheet.css
# YourConfigFile
# and then modify the file new_header.html. See also section "Doxygen usage"
# for information on how to generate the default header that Doxygen normally
# uses.
# Note: The header is subject to change so you typically have to regenerate the
# default header when upgrading to a newer version of Doxygen. For a description
# of the possible markers and block names see the documentation.
# This tag requires that the tag GENERATE_HTML is set to YES.

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				198
Змн.	Арк.	№ докум.	Підпис	Дата		

HTML_HEADER =

The HTML_HEADER tag can be used to specify a user-defined HTML header for each generated HTML page. If the tag is left blank Doxygen will generate a standard header. See HTML_HEADER for more information on how to generate a default header and what special commands can be used inside the header. See also section "Doxygen usage" for information on how to generate the default header that Doxygen normally uses.
This tag requires that the tag GENERATE_HTML is set to YES.

HTML_FOOTER =

The HTML_FOOTER tag can be used to specify a user-defined cascading style sheet that is used by each HTML page. It can be used to fine-tune the look of the HTML output. If left blank Doxygen will generate a default style sheet. See also section "Doxygen usage" for information on how to generate the style sheet that Doxygen normally uses.
Note: It is recommended to use HTML_EXTRA_STYLESHEET instead of this tag, as it is more robust and this tag (HTML_FOOTER) will in the future become obsolete.
This tag requires that the tag GENERATE_HTML is set to YES.

HTML_STYLESHEET =

The HTML_EXTRA_STYLESHEET tag can be used to specify additional user-defined cascading style sheets that are included after the standard style sheets created by Doxygen. Using this option one can overrule certain style aspects. This is preferred over using HTML_STYLESHEET since it does not replace the standard style sheet and is therefore more robust against future updates. Doxygen will copy the style sheet files to the output directory.
Note: The order of the extra style sheet files is of importance (e.g. the last style sheet in the list overrules the setting of the previous ones in the list).
Note: Since the styling of scrollbars can currently not be overruled in Webkit/Chromium, the styling will be left out of the default doxygen.css if one or more extra stylesheets have been specified. So if scrollbar customization is desired it has to be added explicitly. For an example see the documentation.
This tag requires that the tag GENERATE_HTML is set to YES.

HTML_EXTRA_STYLESHEET =

The HTML_EXTRA_FILES tag can be used to specify one or more extra images or other source files which should be copied to the HTML output directory. Note that these files will be copied to the base HTML output directory. Use the \$relpath^ marker in the HTML_HEADER and/or HTML_FOOTER files to load these files. In the HTML_STYLESHEET file, use the file name only. Also note that the files will be copied as-is; there are no commands or markers available.
This tag requires that the tag GENERATE_HTML is set to YES.

HTML_EXTRA_FILES =

The HTML_COLORSTYLE tag can be used to specify if the generated HTML output should be rendered with a dark or light theme.
Possible values are: LIGHT always generates light mode output, DARK always generates dark mode output, AUTO_LIGHT automatically sets the mode according to the user preference, uses light mode if no preference is set (the default), AUTO_DARK automatically sets the mode according to the user preference, uses dark mode if no preference is set and TOGGLE allows a user to switch between light and dark mode via a button.
The default value is: AUTO_LIGHT.
This tag requires that the tag GENERATE_HTML is set to YES.

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				199
Змн.	Арк.	№ докум.	Підпис	Дата		

```

HTML_COLORSTYLE          = AUTO_LIGHT

# The HTML_COLORSTYLE_HUE tag controls the color of the HTML output. Doxygen
# will adjust the colors in the style sheet and background images according to
# this color. Hue is specified as an angle on a color-wheel, see
# https://en.wikipedia.org/wiki/Hue for more information. For instance the value
# 0 represents red, 60 is yellow, 120 is green, 180 is cyan, 240 is blue, 300
# purple, and 360 is red again.
# Minimum value: 0, maximum value: 359, default value: 220.
# This tag requires that the tag GENERATE_HTML is set to YES.

HTML_COLORSTYLE_HUE      = 220

# The HTML_COLORSTYLE_SAT tag controls the purity (or saturation) of the colors
# in the HTML output. For a value of 0 the output will use gray-scales only. A
# value of 255 will produce the most vivid colors.
# Minimum value: 0, maximum value: 255, default value: 100.
# This tag requires that the tag GENERATE_HTML is set to YES.

HTML_COLORSTYLE_SAT      = 100

# The HTML_COLORSTYLE_GAMMA tag controls the gamma correction applied to the
# luminance component of the colors in the HTML output. Values below 100
# gradually make the output lighter, whereas values above 100 make the output
# darker. The value divided by 100 is the actual gamma applied, so 80 represents
# a gamma of 0.8, The value 220 represents a gamma of 2.2, and 100 does not
# change the gamma.
# Minimum value: 40, maximum value: 240, default value: 80.
# This tag requires that the tag GENERATE_HTML is set to YES.

HTML_COLORSTYLE_GAMMA    = 80

# If the HTML_DYNAMIC_MENUS tag is set to YES then the generated HTML
# documentation will contain a main index with vertical navigation menus that
# are dynamically created via JavaScript. If disabled, the navigation index will
# consists of multiple levels of tabs that are statically embedded in every HTML
# page. Disable this option to support browsers that do not have JavaScript,
# like the Qt help browser.
# The default value is: YES.
# This tag requires that the tag GENERATE_HTML is set to YES.

HTML_DYNAMIC_MENUS       = YES

# If the HTML_DYNAMIC_SECTIONS tag is set to YES then the generated HTML
# documentation will contain sections that can be hidden and shown after the
# page has loaded.
# The default value is: NO.
# This tag requires that the tag GENERATE_HTML is set to YES.

HTML_DYNAMIC_SECTIONS    = NO

# If the HTML_CODE_FOLDING tag is set to YES then classes and functions can be
# dynamically folded and expanded in the generated HTML source code.
# The default value is: YES.
# This tag requires that the tag GENERATE_HTML is set to YES.

HTML_CODE_FOLDING        = YES

# If the HTML_COPY_CLIPBOARD tag is set to YES then Doxygen will show an icon in
# the top right corner of code and text fragments that allows the user to copy
# its content to the clipboard. Note this only works if supported by the browser

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				200
Змн.	Арк.	№ докум.	Підпис	Дата		

```
# and the web page is served via a secure context (see:
# https://www.w3.org/TR/secure-contexts/), i.e. using the https: or file:
# protocol.
# The default value is: YES.
# This tag requires that the tag GENERATE_HTML is set to YES.

HTML_COPY_CLIPBOARD      = YES

# Doxygen stores a couple of settings persistently in the browser (via e.g.
# cookies). By default these settings apply to all HTML pages generated by
# Doxygen across all projects. The HTML_PROJECT_COOKIE tag can be used to store
# the settings under a project specific key, such that the user preferences will
# be stored separately.
# This tag requires that the tag GENERATE_HTML is set to YES.

HTML_PROJECT_COOKIE      =

# With HTML_INDEX_NUM_ENTRIES one can control the preferred number of entries
# shown in the various tree structured indices initially; the user can expand
# and collapse entries dynamically later on. Doxygen will expand the tree to
# such a level that at most the specified number of entries are visible (unless
# a fully collapsed tree already exceeds this amount). So setting the number of
# entries 1 will produce a full collapsed tree by default. 0 is a special value
# representing an infinite number of entries and will result in a full expanded
# tree by default.
# Minimum value: 0, maximum value: 9999, default value: 100.
# This tag requires that the tag GENERATE_HTML is set to YES.

HTML_INDEX_NUM_ENTRIES = 100

# If the GENERATE_DOCSET tag is set to YES, additional index files will be
# generated that can be used as input for Apple's Xcode 3 integrated development
# environment (see:
# https://developer.apple.com/xcode/), introduced with OSX 10.5 (Leopard). To
# create a documentation set, Doxygen will generate a Makefile in the HTML
# output directory. Running make will produce the docset in that directory and
# running make install will install the docset in
# ~/Library/Developer/Shared/Documentation/DocSets so that Xcode will find it at
# startup. See https://developer.apple.com/library/archive/featuredarticles/Doxy
# genXcode/_index.html for more information.
# The default value is: NO.
# This tag requires that the tag GENERATE_HTML is set to YES.

GENERATE_DOCSET           = NO

# This tag determines the name of the docset feed. A documentation feed provides
# an umbrella under which multiple documentation sets from a single provider
# (such as a company or product suite) can be grouped.
# The default value is: Doxygen generated docs.
# This tag requires that the tag GENERATE_DOCSET is set to YES.

DOCSET_FEEDNAME           = "Doxygen generated docs"

# This tag determines the URL of the docset feed. A documentation feed provides
# an umbrella under which multiple documentation sets from a single provider
# (such as a company or product suite) can be grouped.
# This tag requires that the tag GENERATE_DOCSET is set to YES.

DOCSET_FEEDURL            =

# This tag specifies a string that should uniquely identify the documentation
# set bundle. This should be a reverse domain-name style string, e.g.
```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		201

```

# com.mycompany.MyDocSet. Doxygen will append .docset to the name.
# The default value is: org.doxygen.Project.
# This tag requires that the tag GENERATE_DOCSET is set to YES.

DOCSET_BUNDLE_ID      = org.doxygen.Project

# The DOCSET_PUBLISHER_ID tag specifies a string that should uniquely identify
# the documentation publisher. This should be a reverse domain-name style
# string, e.g. com.mycompany.MyDocSet.documentation.
# The default value is: org.doxygen.Publisher.
# This tag requires that the tag GENERATE_DOCSET is set to YES.

DOCSET_PUBLISHER_ID   = org.doxygen.Publisher

# The DOCSET_PUBLISHER_NAME tag identifies the documentation publisher.
# The default value is: Publisher.
# This tag requires that the tag GENERATE_DOCSET is set to YES.

DOCSET_PUBLISHER_NAME = Publisher

# If the GENERATE_HTMLHELP tag is set to YES then Doxygen generates three
# additional HTML index files: index.hpp, index.hhc, and index.hhk. The
# index.hpp is a project file that can be read by Microsoft's HTML Help Workshop
# on Windows. In the beginning of 2021 Microsoft took the original page, with
# a.o. the download links, offline (the HTML help workshop was already many
# years in maintenance mode). You can download the HTML help workshop from the
# web archives at Installation executable (see:
# http://web.archive.org/web/20160201063255/http://download.microsoft.com/download/ad/0/A/9/0A939EF6-E31C-430F-A3DF-DFAE7960D564/htmlhelp.exe).
#
# The HTML Help Workshop contains a compiler that can convert all HTML output
# generated by Doxygen into a single compiled HTML file (.chm). Compiled HTML
# files are now used as the Windows 98 help format, and will replace the old
# Windows help format (.hlp) on all Windows platforms in the future. Compressed
# HTML files also contain an index, a table of contents, and you can search for
# words in the documentation. The HTML workshop also contains a viewer for
# compressed HTML files.
# The default value is: NO.
# This tag requires that the tag GENERATE_HTML is set to YES.

GENERATE_HTMLHELP      = NO

# The CHM_FILE tag can be used to specify the file name of the resulting .chm
# file. You can add a path in front of the file if the result should not be
# written to the html output directory.
# This tag requires that the tag GENERATE_HTMLHELP is set to YES.

CHM_FILE               =

# The HHC_LOCATION tag can be used to specify the location (absolute path
# including file name) of the HTML help compiler (hhc.exe). If non-empty,
# Doxygen will try to run the HTML help compiler on the generated index.hpp.
# The file has to be specified with full path.
# This tag requires that the tag GENERATE_HTMLHELP is set to YES.

HHC_LOCATION           =

# The GENERATE_CHI flag controls if a separate .chi index file is generated
# (YES) or that it should be included in the main .chm file (NO).
# The default value is: NO.
# This tag requires that the tag GENERATE_HTMLHELP is set to YES.

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		202

```

GENERATE_CHI                = NO

# The CHM_INDEX_ENCODING is used to encode HtmlHelp index (hhk), content (hhc)
# and project file content.
# This tag requires that the tag GENERATE_HTMLHELP is set to YES.

CHM_INDEX_ENCODING          =

# The BINARY_TOC flag controls whether a binary table of contents is generated
# (YES) or a normal table of contents (NO) in the .chm file. Furthermore it
# enables the Previous and Next buttons.
# The default value is: NO.
# This tag requires that the tag GENERATE_HTMLHELP is set to YES.

BINARY_TOC                  = NO

# The TOC_EXPAND flag can be set to YES to add extra items for group members to
# the table of contents of the HTML help documentation and to the tree view.
# The default value is: NO.
# This tag requires that the tag GENERATE_HTMLHELP is set to YES.

TOC_EXPAND                  = NO

# The SITEMAP_URL tag is used to specify the full URL of the place where the
# generated documentation will be placed on the server by the user during the
# deployment of the documentation. The generated sitemap is called sitemap.xml
# and placed on the directory specified by HTML_OUTPUT. In case no SITEMAP_URL
# is specified no sitemap is generated. For information about the sitemap
# protocol see https://www.sitemaps.org
# This tag requires that the tag GENERATE_HTML is set to YES.

SITEMAP_URL                 =

# If the GENERATE_QHP tag is set to YES and both QHP_NAMESPACE and
# QHP_VIRTUAL_FOLDER are set, an additional index file will be generated that
# can be used as input for Qt's qhelpgenerator to generate a Qt Compressed Help
# (.qch) of the generated HTML documentation.
# The default value is: NO.
# This tag requires that the tag GENERATE_HTML is set to YES.

GENERATE_QHP                = NO

# If the QHG_LOCATION tag is specified, the QCH_FILE tag can be used to specify
# the file name of the resulting .qch file. The path specified is relative to
# the HTML output folder.
# This tag requires that the tag GENERATE_QHP is set to YES.

QCH_FILE                    =

# The QHP_NAMESPACE tag specifies the namespace to use when generating Qt Help
# Project output. For more information please see Qt Help Project / Namespace
# (see:
# https://doc.qt.io/archives/qt-4.8/qthelpproject.html#namespace).
# The default value is: org.doxygen.Project.
# This tag requires that the tag GENERATE_QHP is set to YES.

QHP_NAMESPACE               = org.doxygen.Project

# The QHP_VIRTUAL_FOLDER tag specifies the namespace to use when generating Qt
# Help Project output. For more information please see Qt Help Project / Virtual
# Folders (see:
# https://doc.qt.io/archives/qt-4.8/qthelpproject.html#virtual-folders).

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІІЗ	Арк.
		Власенко О.В.				203
Змн.	Арк.	№ докум.	Підпис	Дата		

```

# The default value is: doc.
# This tag requires that the tag GENERATE_QHP is set to YES.

QHP_VIRTUAL_FOLDER      = doc

# If the QHP_CUST_FILTER_NAME tag is set, it specifies the name of a custom
# filter to add. For more information please see Qt Help Project / Custom
# Filters (see:
# https://doc.qt.io/archives/qt-4.8/qthelpproject.html#custom-filters).
# This tag requires that the tag GENERATE_QHP is set to YES.

QHP_CUST_FILTER_NAME     =

# The QHP_CUST_FILTER_ATTRS tag specifies the list of the attributes of the
# custom filter to add. For more information please see Qt Help Project / Custom
# Filters (see:
# https://doc.qt.io/archives/qt-4.8/qthelpproject.html#custom-filters).
# This tag requires that the tag GENERATE_QHP is set to YES.

QHP_CUST_FILTER_ATTRS    =

# The QHP_SECT_FILTER_ATTRS tag specifies the list of the attributes this
# project's filter section matches. Qt Help Project / Filter Attributes (see:
# https://doc.qt.io/archives/qt-4.8/qthelpproject.html#filter-attributes).
# This tag requires that the tag GENERATE_QHP is set to YES.

QHP_SECT_FILTER_ATTRS    =

# The QHG_LOCATION tag can be used to specify the location (absolute path
# including file name) of Qt's qhelpgenerator. If non-empty Doxygen will try to
# run qhelpgenerator on the generated .qhp file.
# This tag requires that the tag GENERATE_QHP is set to YES.

QHG_LOCATION             =

# If the GENERATE_ECLIPSEHELP tag is set to YES, additional index files will be
# generated, together with the HTML files, they form an Eclipse help plugin. To
# install this plugin and make it available under the help contents menu in
# Eclipse, the contents of the directory containing the HTML and XML files needs
# to be copied into the plugins directory of eclipse. The name of the directory
# within the plugins directory should be the same as the ECLIPSE_DOC_ID value.
# After copying Eclipse needs to be restarted before the help appears.
# The default value is: NO.
# This tag requires that the tag GENERATE_HTML is set to YES.

GENERATE_ECLIPSEHELP      = NO

# A unique identifier for the Eclipse help plugin. When installing the plugin
# the directory name containing the HTML and XML files should also have this
# name. Each documentation set should have its own identifier.
# The default value is: org.doxygen.Project.
# This tag requires that the tag GENERATE_ECLIPSEHELP is set to YES.

ECLIPSE_DOC_ID           = org.doxygen.Project

# If you want full control over the layout of the generated HTML pages it might
# be necessary to disable the index and replace it with your own. The
# DISABLE_INDEX tag can be used to turn on/off the condensed index (tabs) at top
# of each HTML page. A value of NO enables the index and the value YES disables
# it. Since the tabs in the index contain the same information as the navigation
# tree, you can set this option to YES if you also set GENERATE_TREEVIEW to YES.
# The default value is: YES.

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		204

```

# This tag requires that the tag GENERATE_HTML is set to YES.

DISABLE_INDEX                = YES

# The GENERATE_TREEVIEW tag is used to specify whether a tree-like index
# structure should be generated to display hierarchical information. If the tag
# value is set to YES, a side panel will be generated containing a tree-like
# index structure (just like the one that is generated for HTML Help). For this
# to work a browser that supports JavaScript, DHTML, CSS and frames is required
# (i.e. any modern browser). Windows users are probably better off using the
# HTML help feature. Via custom style sheets (see HTML_EXTRA_STYLESHEET) one can
# further fine tune the look of the index (see "Fine-tuning the output"). As an
# example, the default style sheet generated by Doxygen has an example that
# shows how to put an image at the root of the tree instead of the PROJECT_NAME.
# Since the tree basically has the same information as the tab index, you could
# consider setting DISABLE_INDEX to YES when enabling this option.
# The default value is: YES.
# This tag requires that the tag GENERATE_HTML is set to YES.

GENERATE_TREEVIEW            = YES

# When both GENERATE_TREEVIEW and DISABLE_INDEX are set to YES, then the
# FULL_SIDEBAR option determines if the side bar is limited to only the treeview
# area (value NO) or if it should extend to the full height of the window (value
# YES). Setting this to YES gives a layout similar to
# https://docs.readthedocs.io with more room for contents, but less room for the
# project logo, title, and description. If either GENERATE_TREEVIEW or
# DISABLE_INDEX is set to NO, this option has no effect.
# The default value is: NO.
# This tag requires that the tag GENERATE_HTML is set to YES.

FULL_SIDEBAR                 = NO

# The ENUM_VALUES_PER_LINE tag can be used to set the number of enum values that
# Doxygen will group on one line in the generated HTML documentation.
#
# Note that a value of 0 will completely suppress the enum values from appearing
# in the overview section.
# Minimum value: 0, maximum value: 20, default value: 4.
# This tag requires that the tag GENERATE_HTML is set to YES.

ENUM_VALUES_PER_LINE         = 4

# When the SHOW_ENUM_VALUES tag is set doxygen will show the specified
# enumeration values besides the enumeration mnemonics.
# The default value is: NO.

SHOW_ENUM_VALUES              = NO

# If the treeview is enabled (see GENERATE_TREEVIEW) then this tag can be used
# to set the initial width (in pixels) of the frame in which the tree is shown.
# Minimum value: 0, maximum value: 1500, default value: 250.
# This tag requires that the tag GENERATE_HTML is set to YES.

TREEVIEW_WIDTH                = 250

# If the EXT_LINKS_IN_WINDOW option is set to YES, Doxygen will open links to
# external symbols imported via tag files in a separate window.
# The default value is: NO.
# This tag requires that the tag GENERATE_HTML is set to YES.

EXT_LINKS_IN_WINDOW           = NO

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				205
Змн.	Арк.	№ докум.	Підпис	Дата		

```

# If the OBFUSCATE_EMAILS tag is set to YES, Doxygen will obfuscate email
# addresses.
# The default value is: YES.
# This tag requires that the tag GENERATE_HTML is set to YES.

OBFUSCATE_EMAILS          = YES

# If the HTML_FORMULA_FORMAT option is set to svg, Doxygen will use the pdf2svg
# tool (see https://github.com/dawbarton/pdf2svg) or inkscape (see
# https://inkscape.org) to generate formulas as SVG images instead of PNGs for
# the HTML output. These images will generally look nicer at scaled resolutions.
# Possible values are: png (the default) and svg (looks nicer but requires the
# pdf2svg or inkscape tool).
# The default value is: png.
# This tag requires that the tag GENERATE_HTML is set to YES.

HTML_FORMULA_FORMAT       = png

# Use this tag to change the font size of LaTeX formulas included as images in
# the HTML documentation. When you change the font size after a successful
# Doxygen run you need to manually remove any form_*.png images from the HTML
# output directory to force them to be regenerated.
# Minimum value: 8, maximum value: 50, default value: 10.
# This tag requires that the tag GENERATE_HTML is set to YES.

FORMULA_FONTSIZE          = 10

# The FORMULA_MACROFILE can contain LaTeX \newcommand and \renewcommand commands
# to create new LaTeX commands to be used in formulas as building blocks. See
# the section "Including formulas" for details.

FORMULA_MACROFILE         =

# Enable the USE_MATHJAX option to render LaTeX formulas using MathJax (see
# https://www.mathjax.org) which uses client side JavaScript for the rendering
# instead of using pre-rendered bitmaps. Use this if you do not have LaTeX
# installed or if you want to formulas look prettier in the HTML output. When
# enabled you may also need to install MathJax separately and configure the path
# to it using the MATHJAX_RELPATH option.
# The default value is: NO.
# This tag requires that the tag GENERATE_HTML is set to YES.

USE_MATHJAX               = NO

# With MATHJAX_VERSION it is possible to specify the MathJax version to be used.
# Note that the different versions of MathJax have different requirements with
# regards to the different settings, so it is possible that also other MathJax
# settings have to be changed when switching between the different MathJax
# versions.
# Possible values are: MathJax_2 and MathJax_3.
# The default value is: MathJax_2.
# This tag requires that the tag USE_MATHJAX is set to YES.

MATHJAX_VERSION           = MathJax_2

# When MathJax is enabled you can set the default output format to be used for
# the MathJax output. For more details about the output format see MathJax
# version 2 (see:
# http://docs.mathjax.org/en/v2.7-latest/output.html) and MathJax version 3
# (see:
# http://docs.mathjax.org/en/latest/web/components/output.html).

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				206
Змн.	Арк.	№ докум.	Підпис	Дата		

Possible values are: HTML-CSS (which is slower, but has the best compatibility. This is the name for MathJax version 2, for MathJax version 3 this will be translated into chtml), NativeMML (i.e. MathML. Only supported for MathJax 2. For MathJax version 3 chtml will be used instead.), chtml (This is the name for MathJax version 3, for MathJax version 2 this will be translated into HTML-CSS) and SVG.
 # The default value is: HTML-CSS.
 # This tag requires that the tag USE_MATHJAX is set to YES.

MATHJAX_FORMAT = HTML-CSS

When MathJax is enabled you need to specify the location relative to the HTML output directory using the MATHJAX_RELPATH option. The destination directory should contain the MathJax.js script. For instance, if the mathjax directory is located at the same level as the HTML output directory, then MATHJAX_RELPATH should be ../mathjax. The default value points to the MathJax Content Delivery Network so you can quickly see the result without installing MathJax. However, it is strongly recommended to install a local copy of MathJax from <https://www.mathjax.org> before deployment. The default value is:
 # - in case of MathJax version 2: <https://cdn.jsdelivr.net/npm/mathjax@2>
 # - in case of MathJax version 3: <https://cdn.jsdelivr.net/npm/mathjax@3>
 # This tag requires that the tag USE_MATHJAX is set to YES.

MATHJAX_RELPATH =

The MATHJAX_EXTENSIONS tag can be used to specify one or more MathJax extension names that should be enabled during MathJax rendering. For example for MathJax version 2 (see <https://docs.mathjax.org/en/v2.7-latest/tex.html#tex-and-latex-extensions>):
 # MATHJAX_EXTENSIONS = TeX/AMSmath TeX/AMSsymbols
 # For example for MathJax version 3 (see <http://docs.mathjax.org/en/latest/input/tex/extensions/index.html>):
 # MATHJAX_EXTENSIONS = ams
 # This tag requires that the tag USE_MATHJAX is set to YES.

MATHJAX_EXTENSIONS =

The MATHJAX_CODEFILE tag can be used to specify a file with JavaScript pieces of code that will be used on startup of the MathJax code. See the MathJax site (see: <http://docs.mathjax.org/en/v2.7-latest/output.html>) for more details. For an example see the documentation.
 # This tag requires that the tag USE_MATHJAX is set to YES.

MATHJAX_CODEFILE =

When the SEARCHENGINE tag is enabled Doxygen will generate a search box for the HTML output. The underlying search engine uses JavaScript and DHTML and should work on any modern browser. Note that when using HTML help (GENERATE_HTMLHELP), Qt help (GENERATE_QHP), or docsets (GENERATE_DOCSET) there is already a search function so this one should typically be disabled. For large projects the JavaScript based search engine can be slow, then enabling SERVER_BASED_SEARCH may provide a better solution. It is possible to search using the keyboard; to jump to the search box use <access key> + S (what the <access key> is depends on the OS and browser, but it is typically <CTRL>, <ALT>/<option>, or both). Inside the search box use the <cursor down key> to jump into the search results window, the results can be navigated using the <cursor keys>. Press <Enter> to select an item or <escape> to cancel the search. The filter options can be selected when the cursor is inside the search box by pressing <Shift>+<cursor down>. Also here use the <cursor keys> to select a filter and <Enter> or <escape> to activate or cancel the filter option.

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				207
Змн.	Арк.	№ докум.	Підпис	Дата		

```

# The default value is: YES.
# This tag requires that the tag GENERATE_HTML is set to YES.

SEARCHENGINE          = YES

# When the SERVER_BASED_SEARCH tag is enabled the search engine will be
# implemented using a web server instead of a web client using JavaScript. There
# are two flavors of web server based searching depending on the EXTERNAL_SEARCH
# setting. When disabled, Doxygen will generate a PHP script for searching and
# an index file used by the script. When EXTERNAL_SEARCH is enabled the indexing
# and searching needs to be provided by external tools. See the section
# "External Indexing and Searching" for details.
# The default value is: NO.
# This tag requires that the tag SEARCHENGINE is set to YES.

SERVER_BASED_SEARCH   = NO

# When EXTERNAL_SEARCH tag is enabled Doxygen will no longer generate the PHP
# script for searching. Instead the search results are written to an XML file
# which needs to be processed by an external indexer. Doxygen will invoke an
# external search engine pointed to by the SEARCHENGINE_URL option to obtain the
# search results.
#
# Doxygen ships with an example indexer (doxyindexer) and search engine
# (doxysearch.cgi) which are based on the open source search engine library
# Xapian (see:
# https://xapian.org/).
#
# See the section "External Indexing and Searching" for details.
# The default value is: NO.
# This tag requires that the tag SEARCHENGINE is set to YES.

EXTERNAL_SEARCH        = NO

# The SEARCHENGINE_URL should point to a search engine hosted by a web server
# which will return the search results when EXTERNAL_SEARCH is enabled.
#
# Doxygen ships with an example indexer (doxyindexer) and search engine
# (doxysearch.cgi) which are based on the open source search engine library
# Xapian (see:
# https://xapian.org/). See the section "External Indexing and Searching" for
# details.
# This tag requires that the tag SEARCHENGINE is set to YES.

SEARCHENGINE_URL       =

# When SERVER_BASED_SEARCH and EXTERNAL_SEARCH are both enabled the unindexed
# search data is written to a file for indexing by an external tool. With the
# SEARCHDATA_FILE tag the name of this file can be specified.
# The default file is: searchdata.xml.
# This tag requires that the tag SEARCHENGINE is set to YES.

SEARCHDATA_FILE        = searchdata.xml

# When SERVER_BASED_SEARCH and EXTERNAL_SEARCH are both enabled the
# EXTERNAL_SEARCH_ID tag can be used as an identifier for the project. This is
# useful in combination with EXTRA_SEARCH_MAPPINGS to search through multiple
# projects and redirect the results back to the right project.
# This tag requires that the tag SEARCHENGINE is set to YES.

EXTERNAL_SEARCH_ID     =

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІІЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		208

```

# The EXTRA_SEARCH_MAPPINGS tag can be used to enable searching through Doxygen
# projects other than the one defined by this configuration file, but that are
# all added to the same external search index. Each project needs to have a
# unique id set via EXTERNAL_SEARCH_ID. The search mapping then maps the id of
# to a relative location where the documentation can be found. The format is:
# EXTRA_SEARCH_MAPPINGS = tagname1=loc1 tagname2=loc2 ...
# This tag requires that the tag SEARCHENGINE is set to YES.

EXTRA_SEARCH_MAPPINGS  =

#-----
# Configuration options related to the LaTeX output
#-----

# If the GENERATE_LATEX tag is set to YES, Doxygen will generate LaTeX output.
# The default value is: YES.

GENERATE_LATEX          = NO

# The LATEX_OUTPUT tag is used to specify where the LaTeX docs will be put. If a
# relative path is entered the value of OUTPUT_DIRECTORY will be put in front of
# it.
# The default directory is: latex.
# This tag requires that the tag GENERATE_LATEX is set to YES.

LATEX_OUTPUT            = latex

# The LATEX_CMD_NAME tag can be used to specify the LaTeX command name to be
# invoked.
#
# Note that when not enabling USE_PDFLATEX the default is latex when enabling
# USE_PDFLATEX the default is pdflatex and when in the later case latex is
# chosen this is overwritten by pdflatex. For specific output languages the
# default can have been set differently, this depends on the implementation of
# the output language.
# This tag requires that the tag GENERATE_LATEX is set to YES.

LATEX_CMD_NAME          =

# The MAKEINDEX_CMD_NAME tag can be used to specify the command name to generate
# index for LaTeX.
# Note: This tag is used in the Makefile / make.bat.
# See also: LATEX_MAKEINDEX_CMD for the part in the generated output file
# (.tex).
# The default file is: makeindex.
# This tag requires that the tag GENERATE_LATEX is set to YES.

MAKEINDEX_CMD_NAME      = makeindex

# The LATEX_MAKEINDEX_CMD tag can be used to specify the command name to
# generate index for LaTeX. In case there is no backslash (\) as first character
# it will be automatically added in the LaTeX code.
# Note: This tag is used in the generated output file (.tex).
# See also: MAKEINDEX_CMD_NAME for the part in the Makefile / make.bat.
# The default value is: makeindex.
# This tag requires that the tag GENERATE_LATEX is set to YES.

LATEX_MAKEINDEX_CMD     = makeindex

# If the COMPACT_LATEX tag is set to YES, Doxygen generates more compact LaTeX
# documents. This may be useful for small projects and may help to save some
# trees in general.

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		209

```

# The default value is: NO.
# This tag requires that the tag GENERATE_LATEX is set to YES.

COMPACT_LATEX          = NO

# The PAPER_TYPE tag can be used to set the paper type that is used by the
# printer.
# Possible values are: a4 (210 x 297 mm), letter (8.5 x 11 inches), legal (8.5 x
# 14 inches) and executive (7.25 x 10.5 inches).
# The default value is: a4.
# This tag requires that the tag GENERATE_LATEX is set to YES.

PAPER_TYPE              = a4

# The EXTRA_PACKAGES tag can be used to specify one or more LaTeX package names
# that should be included in the LaTeX output. The package can be specified just
# by its name or with the correct syntax as to be used with the LaTeX
# \usepackage command. To get the times font for instance you can specify :
# EXTRA_PACKAGES=times or EXTRA_PACKAGES={times}
# To use the option intlimits with the amsmath package you can specify:
# EXTRA_PACKAGES=[intlimits]{amsmath}
# If left blank no extra packages will be included.
# This tag requires that the tag GENERATE_LATEX is set to YES.

EXTRA_PACKAGES          =

# The LATEX_HEADER tag can be used to specify a user-defined LaTeX header for
# the generated LaTeX document. The header should contain everything until the
# first chapter. If it is left blank Doxygen will generate a standard header. It
# is highly recommended to start with a default header using
# doxygen -w latex new_header.tex new_footer.tex new_stylesheets.sty
# and then modify the file new_header.tex. See also section "Doxygen usage" for
# information on how to generate the default header that Doxygen normally uses.
#
# Note: Only use a user-defined header if you know what you are doing!
# Note: The header is subject to change so you typically have to regenerate the
# default header when upgrading to a newer version of Doxygen. The following
# commands have a special meaning inside the header (and footer): For a
# description of the possible markers and block names see the documentation.
# This tag requires that the tag GENERATE_LATEX is set to YES.

LATEX_HEADER            =

# The LATEX_FOOTER tag can be used to specify a user-defined LaTeX footer for
# the generated LaTeX document. The footer should contain everything after the
# last chapter. If it is left blank Doxygen will generate a standard footer. See
# LATEX_HEADER for more information on how to generate a default footer and what
# special commands can be used inside the footer. See also section "Doxygen
# usage" for information on how to generate the default footer that Doxygen
# normally uses. Note: Only use a user-defined footer if you know what you are
# doing!
# This tag requires that the tag GENERATE_LATEX is set to YES.

LATEX_FOOTER            =

# The LATEX_EXTRA_STYLESHEET tag can be used to specify additional user-defined
# LaTeX style sheets that are included after the standard style sheets created
# by Doxygen. Using this option one can overrule certain style aspects. Doxygen
# will copy the style sheet files to the output directory.
# Note: The order of the extra style sheet files is of importance (e.g. the last
# style sheet in the list overrules the setting of the previous ones in the
# list).

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		210

```

# This tag requires that the tag GENERATE_LATEX is set to YES.

LATEX_EXTRA_STYLESHEET =

# The LATEX_EXTRA_FILES tag can be used to specify one or more extra images or
# other source files which should be copied to the LATEX_OUTPUT output
# directory. Note that the files will be copied as-is; there are no commands or
# markers available.
# This tag requires that the tag GENERATE_LATEX is set to YES.

LATEX_EXTRA_FILES      =

# If the PDF_HYPERLINKS tag is set to YES, the LaTeX that is generated is
# prepared for conversion to PDF (using ps2pdf or pdflatex). The PDF file will
# contain links (just like the HTML output) instead of page references. This
# makes the output suitable for online browsing using a PDF viewer.
# The default value is: YES.
# This tag requires that the tag GENERATE_LATEX is set to YES.

PDF_HYPERLINKS         = YES

# If the USE_PDFLATEX tag is set to YES, Doxygen will use the engine as
# specified with LATEX_CMD_NAME to generate the PDF file directly from the LaTeX
# files. Set this option to YES, to get a higher quality PDF documentation.
#
# See also section LATEX_CMD_NAME for selecting the engine.
# The default value is: YES.
# This tag requires that the tag GENERATE_LATEX is set to YES.

USE_PDFLATEX           = YES

# The LATEX_BATCHMODE tag signals the behavior of LaTeX in case of an error.
# Possible values are: NO same as ERROR_STOP, YES same as BATCH, BATCH In batch
# mode nothing is printed on the terminal, errors are scrolled as if <return> is
# hit at every error; missing files that TeX tries to input or request from
# keyboard input (\read on a not open input stream) cause the job to abort,
# NON_STOP In nonstop mode the diagnostic message will appear on the terminal,
# but there is no possibility of user interaction just like in batch mode,
# SCROLL In scroll mode, TeX will stop only for missing files to input or if
# keyboard input is necessary and ERROR_STOP In errorstop mode, TeX will stop at
# each error, asking for user intervention.
# The default value is: NO.
# This tag requires that the tag GENERATE_LATEX is set to YES.

LATEX_BATCHMODE        = NO

# If the LATEX_HIDE_INDICES tag is set to YES then Doxygen will not include the
# index chapters (such as File Index, Compound Index, etc.) in the output.
# The default value is: NO.
# This tag requires that the tag GENERATE_LATEX is set to YES.

LATEX_HIDE_INDICES     = NO

# The LATEX_BIB_STYLE tag can be used to specify the style to use for the
# bibliography, e.g. plainnat, or ieeetr. See
# https://en.wikipedia.org/wiki/BibTeX and \cite for more info.
# The default value is: plainnat.
# This tag requires that the tag GENERATE_LATEX is set to YES.

LATEX_BIB_STYLE        = plainnat

# The LATEX_EMOJI_DIRECTORY tag is used to specify the (relative or absolute)

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		211

```

# path from which the emoji images will be read. If a relative path is entered,
# it will be relative to the LATEX_OUTPUT directory. If left blank the
# LATEX_OUTPUT directory will be used.
# This tag requires that the tag GENERATE_LATEX is set to YES.

LATEX_EMOJI_DIRECTORY  =

#-----
# Configuration options related to the RTF output
#-----

# If the GENERATE_RTF tag is set to YES, Doxygen will generate RTF output. The
# RTF output is optimized for Word 97 and may not look too pretty with other RTF
# readers/editors.
# The default value is: NO.

GENERATE_RTF            = NO

# The RTF_OUTPUT tag is used to specify where the RTF docs will be put. If a
# relative path is entered the value of OUTPUT_DIRECTORY will be put in front of
# it.
# The default directory is: rtf.
# This tag requires that the tag GENERATE_RTF is set to YES.

RTF_OUTPUT              = rtf

# If the COMPACT_RTF tag is set to YES, Doxygen generates more compact RTF
# documents. This may be useful for small projects and may help to save some
# trees in general.
# The default value is: NO.
# This tag requires that the tag GENERATE_RTF is set to YES.

COMPACT_RTF            = NO

# If the RTF_HYPERLINKS tag is set to YES, the RTF that is generated will
# contain hyperlink fields. The RTF file will contain links (just like the HTML
# output) instead of page references. This makes the output suitable for online
# browsing using Word or some other Word compatible readers that support those
# fields.
#
# Note: WordPad (write) and others do not support links.
# The default value is: NO.
# This tag requires that the tag GENERATE_RTF is set to YES.

RTF_HYPERLINKS         = NO

# Load stylesheet definitions from file. Syntax is similar to Doxygen's
# configuration file, i.e. a series of assignments. You only have to provide
# replacements, missing definitions are set to their default value.
#
# See also section "Doxygen usage" for information on how to generate the
# default style sheet that Doxygen normally uses.
# This tag requires that the tag GENERATE_RTF is set to YES.

RTF_STYLESHEET_FILE    =

# Set optional variables used in the generation of an RTF document. Syntax is
# similar to Doxygen's configuration file. A template extensions file can be
# generated using doxygen -e rtf extensionFile.
# This tag requires that the tag GENERATE_RTF is set to YES.

RTF_EXTENSIONS_FILE    =

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		212

```
# The RTF_EXTRA_FILES tag can be used to specify one or more extra images or
# other source files which should be copied to the RTF_OUTPUT output directory.
# Note that the files will be copied as-is; there are no commands or markers
# available.
# This tag requires that the tag GENERATE_RTF is set to YES.
```

```
RTF_EXTRA_FILES          =
```

```
#-----
# Configuration options related to the man page output
#-----
```

```
# If the GENERATE_MAN tag is set to YES, Doxygen will generate man pages for
# classes and files.
# The default value is: NO.
```

```
GENERATE_MAN             = NO
```

```
# The MAN_OUTPUT tag is used to specify where the man pages will be put. If a
# relative path is entered the value of OUTPUT_DIRECTORY will be put in front of
# it. A directory man3 will be created inside the directory specified by
# MAN_OUTPUT.
# The default directory is: man.
# This tag requires that the tag GENERATE_MAN is set to YES.
```

```
MAN_OUTPUT               = man
```

```
# The MAN_EXTENSION tag determines the extension that is added to the generated
# man pages. In case the manual section does not start with a number, the number
# 3 is prepended. The dot (.) at the beginning of the MAN_EXTENSION tag is
# optional.
# The default value is: .3.
# This tag requires that the tag GENERATE_MAN is set to YES.
```

```
MAN_EXTENSION            = .3
```

```
# The MAN_SUBDIR tag determines the name of the directory created within
# MAN_OUTPUT in which the man pages are placed. If defaults to man followed by
# MAN_EXTENSION with the initial . removed.
# This tag requires that the tag GENERATE_MAN is set to YES.
```

```
MAN_SUBDIR               =
```

```
# If the MAN_LINKS tag is set to YES and Doxygen generates man output, then it
# will generate one additional man file for each entity documented in the real
# man page(s). These additional files only source the real man page, but without
# them the man command would be unable to find the correct page.
# The default value is: NO.
# This tag requires that the tag GENERATE_MAN is set to YES.
```

```
MAN_LINKS                = NO
```

```
#-----
# Configuration options related to the XML output
#-----
```

```
# If the GENERATE_XML tag is set to YES, Doxygen will generate an XML file that
# captures the structure of the code including all documentation.
# The default value is: NO.
```

```
GENERATE_XML             = NO
```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		213

```
# The XML_OUTPUT tag is used to specify where the XML pages will be put. If a
# relative path is entered the value of OUTPUT_DIRECTORY will be put in front of
# it.
# The default directory is: xml.
# This tag requires that the tag GENERATE_XML is set to YES.
```

```
XML_OUTPUT = xml
```

```
# If the XML_PROGRAMLISTING tag is set to YES, Doxygen will dump the program
# listings (including syntax highlighting and cross-referencing information) to
# the XML output. Note that enabling this will significantly increase the size
# of the XML output.
# The default value is: YES.
# This tag requires that the tag GENERATE_XML is set to YES.
```

```
XML_PROGRAMLISTING = YES
```

```
# If the XML_NS_MEMB_FILE_SCOPE tag is set to YES, Doxygen will include
# namespace members in file scope as well, matching the HTML output.
# The default value is: NO.
# This tag requires that the tag GENERATE_XML is set to YES.
```

```
XML_NS_MEMB_FILE_SCOPE = NO
```

```
#-----
# Configuration options related to the DOCBOOK output
#-----
```

```
# If the GENERATE_DOCBOOK tag is set to YES, Doxygen will generate Docbook files
# that can be used to generate PDF.
# The default value is: NO.
```

```
GENERATE_DOCBOOK = NO
```

```
# The DOCBOOK_OUTPUT tag is used to specify where the Docbook pages will be put.
# If a relative path is entered the value of OUTPUT_DIRECTORY will be put in
# front of it.
# The default directory is: docbook.
# This tag requires that the tag GENERATE_DOCBOOK is set to YES.
```

```
DOCBOOK_OUTPUT = docbook
```

```
#-----
# Configuration options for the AutoGen Definitions output
#-----
```

```
# If the GENERATE_AUTOGEN_DEF tag is set to YES, Doxygen will generate an
# AutoGen Definitions (see https://autogen.sourceforge.net/) file that captures
# the structure of the code including all documentation. Note that this feature
# is still experimental and incomplete at the moment.
# The default value is: NO.
```

```
GENERATE_AUTOGEN_DEF = NO
```

```
#-----
# Configuration options related to Sqlite3 output
#-----
```

```
# If the GENERATE_SQLITE3 tag is set to YES Doxygen will generate a Sqlite3
# database with symbols found by Doxygen stored in tables.
# The default value is: NO.
```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		214

```

GENERATE_SQLITE3          = NO

# The SQLITE3_OUTPUT tag is used to specify where the Sqlite3 database will be
# put. If a relative path is entered the value of OUTPUT_DIRECTORY will be put
# in front of it.
# The default directory is: sqlite3.
# This tag requires that the tag GENERATE_SQLITE3 is set to YES.

SQLITE3_OUTPUT            = sqlite3

# The SQLITE3_RECREATE_DB tag is set to YES, the existing doxygen_sqlite3.db
# database file will be recreated with each Doxygen run. If set to NO, Doxygen
# will warn if a database file is already found and not modify it.
# The default value is: YES.
# This tag requires that the tag GENERATE_SQLITE3 is set to YES.

SQLITE3_RECREATE_DB      = YES

#-----
# Configuration options related to the Perl module output
#-----

# If the GENERATE_PERLMOD tag is set to YES, Doxygen will generate a Perl module
# file that captures the structure of the code including all documentation.
#
# Note that this feature is still experimental and incomplete at the moment.
# The default value is: NO.

GENERATE_PERLMOD          = NO

# If the PERLMOD_LATEX tag is set to YES, Doxygen will generate the necessary
# Makefile rules, Perl scripts and LaTeX code to be able to generate PDF and DVI
# output from the Perl module output.
# The default value is: NO.
# This tag requires that the tag GENERATE_PERLMOD is set to YES.

PERLMOD_LATEX             = NO

# If the PERLMOD_PRETTY tag is set to YES, the Perl module output will be nicely
# formatted so it can be parsed by a human reader. This is useful if you want to
# understand what is going on. On the other hand, if this tag is set to NO, the
# size of the Perl module output will be much smaller and Perl will parse it
# just the same.
# The default value is: YES.
# This tag requires that the tag GENERATE_PERLMOD is set to YES.

PERLMOD_PRETTY            = YES

# The names of the make variables in the generated doxyrules.make file are
# prefixed with the string contained in PERLMOD_MAKEVAR_PREFIX. This is useful
# so different doxyrules.make files included by the same Makefile don't
# overwrite each other's variables.
# This tag requires that the tag GENERATE_PERLMOD is set to YES.

PERLMOD_MAKEVAR_PREFIX =

#-----
# Configuration options related to the preprocessor
#-----

# If the ENABLE_PREPROCESSING tag is set to YES, Doxygen will evaluate all

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				215
Змн.	Арк.	№ докум.	Підпис	Дата		

```

# C-preprocessor directives found in the sources and include files.
# The default value is: YES.

ENABLE_PREPROCESSING    = YES

# If the MACRO_EXPANSION tag is set to YES, Doxygen will expand all macro names
# in the source code. If set to NO, only conditional compilation will be
# performed. Macro expansion can be done in a controlled way by setting
# EXPAND_ONLY_PREDEF to YES.
# The default value is: NO.
# This tag requires that the tag ENABLE_PREPROCESSING is set to YES.

MACRO_EXPANSION          = NO

# If the EXPAND_ONLY_PREDEF and MACRO_EXPANSION tags are both set to YES then
# the macro expansion is limited to the macros specified with the PREDEFINED and
# EXPAND_AS_DEFINED tags.
# The default value is: NO.
# This tag requires that the tag ENABLE_PREPROCESSING is set to YES.

EXPAND_ONLY_PREDEF       = NO

# If the SEARCH_INCLUDES tag is set to YES, the include files in the
# INCLUDE_PATH will be searched if a #include is found.
# The default value is: YES.
# This tag requires that the tag ENABLE_PREPROCESSING is set to YES.

SEARCH_INCLUDES          = YES

# The INCLUDE_PATH tag can be used to specify one or more directories that
# contain include files that are not input files but should be processed by the
# preprocessor. Note that the INCLUDE_PATH is not recursive, so the setting of
# RECURSIVE has no effect here.
# This tag requires that the tag SEARCH_INCLUDES is set to YES.

INCLUDE_PATH             =

# You can use the INCLUDE_FILE_PATTERNS tag to specify one or more wildcard
# patterns (like *.h and *.hpp) to filter out the header-files in the
# directories. If left blank, the patterns specified with FILE_PATTERNS will be
# used.
# This tag requires that the tag ENABLE_PREPROCESSING is set to YES.

INCLUDE_FILE_PATTERNS    = .h .hpp .hxx .hh

# The PREDEFINED tag can be used to specify one or more macro names that are
# defined before the preprocessor is started (similar to the -D option of e.g.
# gcc). The argument of the tag is a list of macros of the form: name or
# name=definition (no spaces). If the definition and the "=" are omitted, "=1"
# is assumed. To prevent a macro definition from being undefined via #undef or
# recursively expanded use the := operator instead of the = operator.
# This tag requires that the tag ENABLE_PREPROCESSING is set to YES.

PREDEFINED               =

# If the MACRO_EXPANSION and EXPAND_ONLY_PREDEF tags are set to YES then this
# tag can be used to specify a list of macro names that should be expanded. The
# macro definition that is found in the sources will be used. Use the PREDEFINED
# tag if you want to use a different macro definition that overrules the
# definition found in the source code.
# This tag requires that the tag ENABLE_PREPROCESSING is set to YES.

```

		Бабушко А.С.			ІІЗ.КР.Б.-121-25-ІІЗ	Арк.
		Власенко О.В.				216
Змн.	Арк.	№ докум.	Підпис	Дата		

```

EXPAND_AS_DEFINED      =

# If the SKIP_FUNCTION_MACROS tag is set to YES then Doxygen's preprocessor will
# remove all references to function-like macros that are alone on a line, have
# an all uppercase name, and do not end with a semicolon. Such function macros
# are typically used for boiler-plate code, and will confuse the parser if not
# removed.
# The default value is: YES.
# This tag requires that the tag ENABLE_PREPROCESSING is set to YES.

SKIP_FUNCTION_MACROS   = YES

#-----
# Configuration options related to external references
#-----

# The TAGFILES tag can be used to specify one or more tag files. For each tag
# file the location of the external documentation should be added. The format of
# a tag file without this location is as follows:
# TAGFILES = file1 file2 ...
# Adding location for the tag files is done as follows:
# TAGFILES = file1=loc1 "file2 = loc2" ...
# where loc1 and loc2 can be relative or absolute paths or URLs. See the
# section "Linking to external documentation" for more information about the use
# of tag files.
# Note: Each tag file must have a unique name (where the name does NOT include
# the path). If a tag file is not located in the directory in which Doxygen is
# run, you must also specify the path to the tagfile here.

TAGFILES                =

# When a file name is specified after GENERATE_TAGFILE, Doxygen will create a
# tag file that is based on the input files it reads. See section "Linking to
# external documentation" for more information about the usage of tag files.

GENERATE_TAGFILE        =

# If the ALLEXTERNALS tag is set to YES, all external classes and namespaces
# will be listed in the class and namespace index. If set to NO, only the
# inherited external classes will be listed.
# The default value is: NO.

ALLEXTERNALS           = NO

# If the EXTERNAL_GROUPS tag is set to YES, all external groups will be listed
# in the topic index. If set to NO, only the current project's groups will be
# listed.
# The default value is: YES.

EXTERNAL_GROUPS        = YES

# If the EXTERNAL_PAGES tag is set to YES, all external pages will be listed in
# the related pages index. If set to NO, only the current project's pages will
# be listed.
# The default value is: YES.

EXTERNAL_PAGES         = YES

#-----
# Configuration options related to diagram generator tools
#-----

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		217

```

# If set to YES the inheritance and collaboration graphs will hide inheritance
# and usage relations if the target is undocumented or is not a class.
# The default value is: YES.

HIDE_UNDOC_RELATIONS    = YES

# If you set the HAVE_DOT tag to YES then Doxygen will assume the dot tool is
# available from the path. This tool is part of Graphviz (see:
# https://www.graphviz.org/), a graph visualization toolkit from AT&T and Lucent
# Bell Labs. The other options in this section have no effect if this option is
# set to NO
# The default value is: NO.

HAVE_DOT                = NO

# The DOT_NUM_THREADS specifies the number of dot invocations Doxygen is allowed
# to run in parallel. When set to 0 Doxygen will base this on the number of
# processors available in the system. You can set it explicitly to a value
# larger than 0 to get control over the balance between CPU load and processing
# speed.
# Minimum value: 0, maximum value: 32, default value: 0.
# This tag requires that the tag HAVE_DOT is set to YES.

DOT_NUM_THREADS         = 0

# DOT_COMMON_ATTR is common attributes for nodes, edges and labels of
# subgraphs. When you want a differently looking font in the dot files that
# Doxygen generates you can specify fontname, fontcolor and fontsize attributes.
# For details please see <a href=https://graphviz.org/doc/info/attrs.html>Node,
# Edge and Graph Attributes specification</a> You need to make sure dot is able
# to find the font, which can be done by putting it in a standard location or by
# setting the DOTFONTPATH environment variable or by setting DOT_FONTPATH to the
# directory containing the font. Default graphviz fontsize is 14.
# The default value is: fontname=Helvetica,fontsize=10.
# This tag requires that the tag HAVE_DOT is set to YES.

DOT_COMMON_ATTR         = "fontname=Helvetica,fontsize=10"

# DOT_EDGE_ATTR is concatenated with DOT_COMMON_ATTR. For elegant style you can
# add 'arrowhead=open, arrowtail=open, arrowsize=0.5'. <a
# href=https://graphviz.org/doc/info/arrows.html>Complete documentation about
# arrows shapes.</a>
# The default value is: labelfontname=Helvetica,labelfontsize=10.
# This tag requires that the tag HAVE_DOT is set to YES.

DOT_EDGE_ATTR           = "labelfontname=Helvetica,labelfontsize=10"

# DOT_NODE_ATTR is concatenated with DOT_COMMON_ATTR. For view without boxes
# around nodes set 'shape=plain' or 'shape=plaintext' <a
# href=https://www.graphviz.org/doc/info/shapes.html>Shapes specification</a>
# The default value is: shape=box,height=0.2,width=0.4.
# This tag requires that the tag HAVE_DOT is set to YES.

DOT_NODE_ATTR           = "shape=box,height=0.2,width=0.4"

# You can set the path where dot can find font specified with fontname in
# DOT_COMMON_ATTR and others dot attributes.
# This tag requires that the tag HAVE_DOT is set to YES.

DOT_FONTPATH            =

# If the CLASS_GRAPH tag is set to YES or GRAPH or BUILTIN then Doxygen will

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		218

```
# generate a graph for each documented class showing the direct and indirect
# inheritance relations. In case the CLASS_GRAPH tag is set to YES or GRAPH and
# HAVE_DOT is enabled as well, then dot will be used to draw the graph. In case
# the CLASS_GRAPH tag is set to YES and HAVE_DOT is disabled or if the
# CLASS_GRAPH tag is set to BUILTIN, then the built-in generator will be used.
# If the CLASS_GRAPH tag is set to TEXT the direct and indirect inheritance
# relations will be shown as texts / links. Explicit enabling an inheritance
# graph or choosing a different representation for an inheritance graph of a
# specific class, can be accomplished by means of the command \inheritancegraph.
# Disabling an inheritance graph can be accomplished by means of the command
# \hideinheritancegraph.
# Possible values are: NO, YES, TEXT, GRAPH and BUILTIN.
# The default value is: YES.
```

```
CLASS_GRAPH          = YES
```

```
# If the COLLABORATION_GRAPH tag is set to YES then Doxygen will generate a
# graph for each documented class showing the direct and indirect implementation
# dependencies (inheritance, containment, and class references variables) of the
# class with other documented classes. Explicit enabling a collaboration graph,
# when COLLABORATION_GRAPH is set to NO, can be accomplished by means of the
# command \collaborationgraph. Disabling a collaboration graph can be
# accomplished by means of the command \hidecollaborationgraph.
# The default value is: YES.
# This tag requires that the tag HAVE_DOT is set to YES.
```

```
COLLABORATION_GRAPH  = YES
```

```
# If the GROUP_GRAPHS tag is set to YES then Doxygen will generate a graph for
# groups, showing the direct groups dependencies. Explicit enabling a group
# dependency graph, when GROUP_GRAPHS is set to NO, can be accomplished by means
# of the command \groupgraph. Disabling a directory graph can be accomplished by
# means of the command \hidegroupgraph. See also the chapter Grouping in the
# manual.
# The default value is: YES.
# This tag requires that the tag HAVE_DOT is set to YES.
```

```
GROUP_GRAPHS         = YES
```

```
# If the UML_LOOK tag is set to YES, Doxygen will generate inheritance and
# collaboration diagrams in a style similar to the OMG's Unified Modeling
# Language.
# The default value is: NO.
# This tag requires that the tag HAVE_DOT is set to YES.
```

```
UML_LOOK             = NO
```

```
# If the UML_LOOK tag is enabled, the fields and methods are shown inside the
# class node. If there are many fields or methods and many nodes the graph may
# become too big to be useful. The UML_LIMIT_NUM_FIELDS threshold limits the
# number of items for each type to make the size more manageable. Set this to 0
# for no limit. Note that the threshold may be exceeded by 50% before the limit
# is enforced. So when you set the threshold to 10, up to 15 fields may appear,
# but if the number exceeds 15, the total amount of fields shown is limited to
# 10.
# Minimum value: 0, maximum value: 100, default value: 10.
# This tag requires that the tag UML_LOOK is set to YES.
```

```
UML_LIMIT_NUM_FIELDS = 10
```

```
# If the DOT_UML_DETAILS tag is set to NO, Doxygen will show attributes and
# methods without types and arguments in the UML graphs. If the DOT_UML_DETAILS
```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		219

tag is set to YES, Doxygen will add type and arguments for attributes and methods in the UML graphs. If the DOT_UML_DETAILS tag is set to NONE, Doxygen will not generate fields with class member information in the UML graphs. The class diagrams will look similar to the default class diagrams but using UML notation for the relationships.
 # Possible values are: NO, YES and NONE.
 # The default value is: NO.
 # This tag requires that the tag UML_LOOK is set to YES.

DOT_UML_DETAILS = NO

The DOT_WRAP_THRESHOLD tag can be used to set the maximum number of characters to display on a single line. If the actual line length exceeds this threshold significantly it will be wrapped across multiple lines. Some heuristics are applied to avoid ugly line breaks.
 # Minimum value: 0, maximum value: 1000, default value: 17.
 # This tag requires that the tag HAVE_DOT is set to YES.

DOT_WRAP_THRESHOLD = 17

If the TEMPLATE_RELATIONS tag is set to YES then the inheritance and collaboration graphs will show the relations between templates and their instances.
 # The default value is: NO.
 # This tag requires that the tag HAVE_DOT is set to YES.

TEMPLATE_RELATIONS = NO

If the INCLUDE_GRAPH, ENABLE_PREPROCESSING and SEARCH_INCLUDES tags are set to YES then Doxygen will generate a graph for each documented file showing the direct and indirect include dependencies of the file with other documented files. Explicit enabling an include graph, when INCLUDE_GRAPH is set to NO, can be accomplished by means of the command \includegraph. Disabling an include graph can be accomplished by means of the command \hideincludegraph.
 # The default value is: YES.
 # This tag requires that the tag HAVE_DOT is set to YES.

INCLUDE_GRAPH = YES

If the INCLUDED_BY_GRAPH, ENABLE_PREPROCESSING and SEARCH_INCLUDES tags are set to YES then Doxygen will generate a graph for each documented file showing the direct and indirect include dependencies of the file with other documented files. Explicit enabling an included by graph, when INCLUDED_BY_GRAPH is set to NO, can be accomplished by means of the command \includedbygraph. Disabling an included by graph can be accomplished by means of the command \hideincludedbygraph.
 # The default value is: YES.
 # This tag requires that the tag HAVE_DOT is set to YES.

INCLUDED_BY_GRAPH = YES

If the CALL_GRAPH tag is set to YES then Doxygen will generate a call dependency graph for every global function or class method.
 #
 # Note that enabling this option will significantly increase the time of a run. So in most cases it will be better to enable call graphs for selected functions only using the \callgraph command. Disabling a call graph can be accomplished by means of the command \hidecallgraph.
 # The default value is: NO.
 # This tag requires that the tag HAVE_DOT is set to YES.

CALL_GRAPH = NO

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		220

```
# If the CALLER_GRAPH tag is set to YES then Doxygen will generate a caller
# dependency graph for every global function or class method.
#
# Note that enabling this option will significantly increase the time of a run.
# So in most cases it will be better to enable caller graphs for selected
# functions only using the \callergraph command. Disabling a caller graph can be
# accomplished by means of the command \hidecallergraph.
# The default value is: NO.
# This tag requires that the tag HAVE_DOT is set to YES.
```

```
CALLER_GRAPH          = NO
```

```
# If the GRAPHICAL_HIERARCHY tag is set to YES then Doxygen will graphical
# hierarchy of all classes instead of a textual one.
# The default value is: YES.
# This tag requires that the tag HAVE_DOT is set to YES.
```

```
GRAPHICAL_HIERARCHY   = YES
```

```
# If the DIRECTORY_GRAPH tag is set to YES then Doxygen will show the
# dependencies a directory has on other directories in a graphical way. The
# dependency relations are determined by the #include relations between the
# files in the directories. Explicit enabling a directory graph, when
# DIRECTORY_GRAPH is set to NO, can be accomplished by means of the command
# \directorygraph. Disabling a directory graph can be accomplished by means of
# the command \hidedirectorygraph.
# The default value is: YES.
# This tag requires that the tag HAVE_DOT is set to YES.
```

```
DIRECTORY_GRAPH       = YES
```

```
# The DIR_GRAPH_MAX_DEPTH tag can be used to limit the maximum number of levels
# of child directories generated in directory dependency graphs by dot.
# Minimum value: 1, maximum value: 25, default value: 1.
# This tag requires that the tag DIRECTORY_GRAPH is set to YES.
```

```
DIR_GRAPH_MAX_DEPTH   = 1
```

```
# The DOT_IMAGE_FORMAT tag can be used to set the image format of the images
# generated by dot. For an explanation of the image formats see the section
# output formats in the documentation of the dot tool (Graphviz (see:
# https://www.graphviz.org/)).
```

```
#
# Note the formats svg:cairo and svg:cairo:cairo cannot be used in combination
# with INTERACTIVE_SVG (the INTERACTIVE_SVG will be set to NO).
# Possible values are: png, jpg, gif, svg, png:gd, png:gd:gd, png:cairo,
# png:cairo:gd, png:cairo:cairo, png:cairo:gdiplus, png:gdiplus,
# png:gdiplus:gdiplus, svg:cairo, svg:cairo:cairo, svg:svg, svg:svg:core,
# gif:cairo, gif:cairo:gd, gif:cairo:gdiplus, gif:gdiplus, gif:gdiplus:gdiplus,
# gif:gd, gif:gd:gd, jpg:cairo, jpg:cairo:gd, jpg:cairo:gdiplus, jpg:gd,
# jpg:gd:gd, jpg:gdiplus and jpg:gdiplus:gdiplus.
# The default value is: png.
# This tag requires that the tag HAVE_DOT is set to YES.
```

```
DOT_IMAGE_FORMAT      = png
```

```
# If DOT_IMAGE_FORMAT is set to svg or svg:svg or svg:svg:core, then this option
# can be set to YES to enable generation of interactive SVG images that allow
# zooming and panning.
```

```
#
# Note that this requires a modern browser other than Internet Explorer. Tested
```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				221
Змн.	Арк.	№ докум.	Підпис	Дата		

```

# and working are Firefox, Chrome, Safari, and Opera.
#
# Note This option will be automatically disabled when DOT_IMAGE_FORMAT is set
# to svg:cairo or svg:cairo:cairo.
# The default value is: NO.
# This tag requires that the tag HAVE_DOT is set to YES.

INTERACTIVE_SVG          = NO

# The DOT_PATH tag can be used to specify the path where the dot tool can be
# found. If left blank, it is assumed the dot tool can be found in the path.
# This tag requires that the tag HAVE_DOT is set to YES.

DOT_PATH                  =

# The DOTFILE_DIRS tag can be used to specify one or more directories that
# contain dot files that are included in the documentation (see the \dotfile
# command).
# This tag requires that the tag HAVE_DOT is set to YES.

DOTFILE_DIRS              =

# You can include diagrams made with dia in Doxygen documentation. Doxygen will
# then run dia to produce the diagram and insert it in the documentation. The
# DIA_PATH tag allows you to specify the directory where the dia binary resides.
# If left empty dia is assumed to be found in the default search path.

DIA_PATH                  =

# The DIAFILE_DIRS tag can be used to specify one or more directories that
# contain dia files that are included in the documentation (see the \diafile
# command).

DIAFILE_DIRS              =

# When using PlantUML, the PLANTUML_JAR_PATH tag should be used to specify the
# path where java can find the plantuml.jar file or to the filename of jar file
# to be used. If left blank, it is assumed PlantUML is not used or called during
# a preprocessing step. Doxygen will generate a warning when it encounters a
# \startuml command in this case and will not generate output for the diagram.

PLANTUML_JAR_PATH        =

# When using PlantUML, the PLANTUML_CFG_FILE tag can be used to specify a
# configuration file for PlantUML.

PLANTUML_CFG_FILE        =

# When using PlantUML, the specified paths are searched for files specified by
# the !include statement in a PlantUML block.

PLANTUML_INCLUDE_PATH    =

# The PLANTUMLFILE_DIRS tag can be used to specify one or more directories that
# contain PlantUml files that are included in the documentation (see the
# \plantumlfile command).

PLANTUMLFILE_DIRS        =

# The DOT_GRAPH_MAX_NODES tag can be used to set the maximum number of nodes
# that will be shown in the graph. If the number of nodes in a graph becomes
# larger than this value, Doxygen will truncate the graph, which is visualized

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		222

```

# by representing a node as a red box. Note that if the number of direct
# children of the root node in a graph is already larger than
# DOT_GRAPH_MAX_NODES then the graph will not be shown at all. Also note that
# the size of a graph can be further restricted by MAX_DOT_GRAPH_DEPTH.
# Minimum value: 0, maximum value: 10000, default value: 50.
# This tag requires that the tag HAVE_DOT is set to YES.

DOT_GRAPH_MAX_NODES      = 50

# The MAX_DOT_GRAPH_DEPTH tag can be used to set the maximum depth of the graphs
# generated by dot. A depth value of 3 means that only nodes reachable from the
# root by following a path via at most 3 edges will be shown. Nodes that lay
# further from the root node will be omitted. Note that setting this option to 1
# or 2 may greatly reduce the computation time needed for large code bases. Also
# note that the size of a graph can be further restricted by
# DOT_GRAPH_MAX_NODES. Using a depth of 0 means no depth restriction.
# Minimum value: 0, maximum value: 1000, default value: 0.
# This tag requires that the tag HAVE_DOT is set to YES.

MAX_DOT_GRAPH_DEPTH      = 0

# Set the DOT_MULTI_TARGETS tag to YES to allow dot to generate multiple output
# files in one run (i.e. multiple -o and -T options on the command line). This
# makes dot run faster, but since only newer versions of dot (>1.8.10) support
# this, this feature is disabled by default.
# The default value is: NO.
# This tag requires that the tag HAVE_DOT is set to YES.

DOT_MULTI_TARGETS        = NO

# If the GENERATE_LEGEND tag is set to YES Doxygen will generate a legend page
# explaining the meaning of the various boxes and arrows in the dot generated
# graphs.
# Note: This tag requires that UML_LOOK isn't set, i.e. the Doxygen internal
# graphical representation for inheritance and collaboration diagrams is used.
# The default value is: YES.
# This tag requires that the tag HAVE_DOT is set to YES.

GENERATE_LEGEND           = YES

# If the DOT_CLEANUP tag is set to YES, Doxygen will remove the intermediate
# files that are used to generate the various graphs.
#
# Note: This setting is not only used for dot files but also for msc temporary
# files.
# The default value is: YES.

DOT_CLEANUP               = YES

# You can define message sequence charts within Doxygen comments using the \msc
# command. If the MSCGEN_TOOL tag is left empty (the default), then Doxygen will
# use a built-in version of mscgen tool to produce the charts. Alternatively,
# the MSCGEN_TOOL tag can also specify the name an external tool. For instance,
# specifying prog as the value, Doxygen will call the tool as prog -T
# <outfile_format> -o <outputfile> <inputfile>. The external tool should support
# output file formats "png", "eps", "svg", and "ismap".

MSCGEN_TOOL               =

# The MSCFILE_DIRS tag can be used to specify one or more directories that
# contain msc files that are included in the documentation (see the \mscfile
# command).

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		223

```

MSCFILE_DIRS          =

VERSION    ?= 1.0.0
USER       ?= root
CHANNEL    ?= stable
REMOTE     ?= myserver
LIBRARY_BUILD_DIR = build
NET_DIR    = config/sumo/test
NET_OUTPUT = $(NET_DIR)/test_net.net.xml

.PHONY: help build package create publish test clean generate-net

# TODO: extend help test for new commands
help:
    @echo "Makefile for building, packaging, and publishing the library"
    @echo "Usage: make [target] [parameters]"
    @echo ""
    @echo "Targets:"
    @echo "  build      - Builds the project using CMake"
    @echo "  package    - Packages (installs) the library using CMake install
target"
    @echo "  create     - Creates the Conan package"
    @echo "  publish    - Builds, packages, and publishes the package"
    @echo "  clean      - Removes the build directory"
    @echo ""
    @echo "Parameters (can be overridden):"
    @echo "  VERSION    - Library version (default: $(VERSION))"
    @echo "  USER       - Conan user (default: $(USER))"
    @echo "  CHANNEL    - Conan channel (default: $(CHANNEL))"

build:
    @echo "Building the project, version $(VERSION)..."
    cmake -S . -B build -DCMAKE_BUILD_TYPE=Release -DPROJECT_VERSION=$(VERSION)
    cmake --build build

package: build
    @echo "Packaging (installing) the library..."
    cmake --build build --target install

create: package
    @echo "Creating Conan package for the library version $(VERSION)..."
    $(CONAN) create . --user=$(USER) --channel=$(CHANNEL)

publish: create
    @echo "Uploading Conan package to remote '$(REMOTE)'..."
    conan remote add myserver http://localhost:9300
    conan remote login myserver admin -p admin
    conan upload traffic-pulse-library/$(VERSION)@$(USER)/$(CHANNEL) -r=$(REMOTE)
    @echo "Library version $(VERSION) has been uploaded to '$(REMOTE)' as
$(USER)/$(CHANNEL) !"

test:
    make clean
    make build
    cd $(LIBRARY_BUILD_DIR) && ctest -L library --output-on-failure
    cd ..

clean:
    @echo "Removing the build directory..."
    rm -rf $(LIBRARY_BUILD_DIR)

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		224

```

generate-net:
    @echo "Generating SUMO network file using netconvert..."
    netconvert \
        -n $(NET_DIR)/test_nodes.nod.xml \
        -e $(NET_DIR)/test_edges.edg.xml \
        -x $(NET_DIR)/test_connections.con.xml \
        -t $(NET_DIR)/test_tllogic.tll.xml \
        --output-file=$(NET_OUTPUT)
    @echo "Generated: $(NET_OUTPUT) "

#include "utils/Plot.h"
#include <TrafficPulseLibrary>
#include <chrono>
#include <ctime>
#include <filesystem>
#include <fstream>
#include <iomanip>
#include <spdlog/spdlog.h>
#include <sstream>
#include <vector>

// TODO: docs
static std::string currentTimestamp() {
    const auto now = std::chrono::system_clock::now();
    const std::time_t time_t = std::chrono::system_clock::to_time_t(now);
    const std::tm time = *std::localtime(&time_t);
    std::ostringstream oss;
    oss << std::put_time(&time, "%Y-%m-%d %H:%M:%S");
    return oss.str();
}

int main() {
    const std::string csv_path = "../analysis/comparison.csv";
    spdlog::set_level(spdlog::level::off);

    const auto metrics_map = TrafficPulse::MetricsUtils::readCsv(csv_path);
    TrafficPulse::MetricsUtils::printTable(metrics_map);

    std::vector<std::string> labels;
    std::vector<double> base_avg_travel_time, custom_avg_travel_time;
    std::vector<double> base_avg_waiting_time, custom_avg_waiting_time;
    std::vector<double> base_avg_speed, cust_avg_speed;
    std::vector<double> base_avg_queue_length, custom_avg_queue_length;
    std::vector<double> base_throughput, custom_throughput;
    std::vector<double> base_time_loss, custom_time_loss;
    std::vector<double> base_halting_peak, custom_halting_peak;
    std::vector<double> base_density_peak, custom_density_peak;
    std::vector<double> base_queue_length_peak, custom_queue_length_peak;

    for(auto const& [scenario, metrics_pair] : metrics_map) {
        if(metrics_pair.base.throughput > 0 && metrics_pair.custom.throughput > 0)
        {
            labels.push_back(scenario);
            base_avg_travel_time.push_back(metrics_pair.base.avg_travel_time);
            custom_avg_travel_time.push_back(metrics_pair.custom.avg_travel_time);

            base_avg_waiting_time.push_back(metrics_pair.base.avg_waiting_time);
            custom_avg_waiting_time.push_back(metrics_pair.custom.avg_waiting_time);

            base_avg_speed.push_back(metrics_pair.base.avg_speed);
            cust_avg_speed.push_back(metrics_pair.custom.avg_speed);
        }
    }
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				225
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        base_avg_queue_length.push_back(metrics_pair.base.avg_queue_length);
        custom_avg_queue_length.push_back(metrics_pair.custom.avg_queue_length);

        base_throughput.push_back(metrics_pair.base.throughput);
        custom_throughput.push_back(metrics_pair.custom.throughput);

        base_time_loss.push_back(metrics_pair.base.time_loss);
        custom_time_loss.push_back(metrics_pair.custom.time_loss);

        base_halting_peak.push_back(metrics_pair.base.halted_peak);
        custom_halting_peak.push_back(metrics_pair.custom.halted_peak);

        base_density_peak.push_back(metrics_pair.base.density_peak);
        custom_density_peak.push_back(metrics_pair.custom.density_peak);

        base_queue_length_peak.push_back(metrics_pair.base.queue_peak);
        custom_queue_length_peak.push_back(metrics_pair.custom.queue_peak);
    }
}

Plot plot_travel_time("../analysis/travel_time.png");
plot_travel_time.compare("Avg. travel time", "s", labels,
base_avg_travel_time, custom_avg_travel_time);

Plot plot_waiting_time("../analysis/waiting_time.png");
plot_waiting_time.compare("Avg. waiting time", "s", labels,
base_avg_waiting_time, custom_avg_waiting_time);

Plot plot_average_speed("../analysis/avg_speed.png");
plot_average_speed.compare("Avg. speed", "m/s", labels, base_avg_speed,
cust_avg_speed);

Plot plot_queue_length("../analysis/queue_length.png");
plot_queue_length.compare("Avg. queue length", "s", labels,
base_avg_queue_length, custom_avg_queue_length);

Plot plot_throughput("../analysis/throughput.png");
plot_throughput.compare("Throughput", "trips", labels, base_throughput,
custom_throughput);

Plot plot_time_loss("../analysis/time_loss.png");
plot_time_loss.compare("Total time-loss", "veh*s", labels, base_time_loss,
custom_time_loss);

Plot plot_halting_peak("../analysis/halting_peak.png");
plot_halting_peak.compare("Peak halted vehicles", "veh", labels,
base_halting_peak, custom_halting_peak);

Plot plot_density_peak("../analysis/density_peak.png");
plot_density_peak.compare("Peak density", "veh/km", labels, base_density_peak,
custom_density_peak);

Plot plot_queue_length_peak("../analysis/queue_length_peak.png");
plot_queue_length_peak.compare("Peak queue length", "m", labels,
base_queue_length_peak, custom_queue_length_peak);

std::filesystem::create_directories("../analysis");
std::ofstream md("../analysis/report.md");
md << "## TrafficPulse Metrics Report\n\n";
md << "***Generated:** " << currentTimestamp() << "\n\n";

md << "## Travel-Time Chart\n\n";

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				226
Змн.	Арк.	№ докум.	Підпис	Дата		

```

md << "![Average travel time](travel_time.png)\n\n";

md << "## Waiting-Time Chart\n\n";
md << "![Average waiting time](waiting_time.png)\n\n";

md << "## Speed Chart\n\n";
md << "![Average speed](avg_speed.png)\n\n";

md << "## Throughput Chart\n\n";
md << "![Throughput](throughput.png)\n\n";

md << "## Queue-Length Chart\n\n";
md << "![Average queue length](queue_length.png)\n\n";

md << "## Total time-loss\n\n";
md << "![Total time-loss](time_loss.png)\n\n";

md << "## Peak halted vehicles\n\n";
md << "![Peak halted vehicles](halting_peak.png)\n\n";

md << "## Detailed Breakdown\n\n";
md << "| Scenario | Base (s) | Custom (s) | Alpha% | \n";
md << "|:-----|:-----|:-----|:-----| \n";

for(auto const& [scenario, metrics_pair] : metrics_map) {
    if(metrics_pair.base.throughput == 0 || metrics_pair.custom.throughput ==
0)
        continue;

    double dp = (metrics_pair.base.avg_travel_time -
metrics_pair.custom.avg_travel_time) /
metrics_pair.base.avg_travel_time * 100.0;
    md << "| " << scenario << " | " << std::fixed << std::setprecision(2) <<
metrics_pair.base.avg_travel_time
<< " | " << std::fixed << std::setprecision(2) <<
metrics_pair.custom.avg_travel_time << " | " << std::fixed
<< std::setprecision(1) << dp << "% | \n";
}
md.close();

return 0;
}

#include "scripts/Scenario.h"
#include <TrafficPulseLibrary>
#include <fstream>
#include <regex>

TrafficPulse::Metrics runSumoAndCollect(const std::string& scenario_name,
const std::string& sumo_config,
const std::string& net_config,
const bool is_custom_scenario,
const bool gui) {

TrafficPulse::EventBus bus;

auto timestampTag = [] {
    const std::time_t time_t = std::time(nullptr);
    char buf[32];
    std::strftime(buf, sizeof(buf), "%Y%m%d_%H%M%S", std::localtime(&time_t));
    return std::string(buf);
};

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІЗ	Арк.
		Власенко О.В.				227
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    const std::string tag = scenario_name + "_" + (is_custom_scenario ? "custom_"
: "base_") + timestampTag();

    const std::vector<TrafficPulse::SumoOutput> outs = {"--tripinfo-output",
"tripinfo_" + tag + ".xml"},
                                                    {"--summary-output", "summary_" + tag +
".xml"},
                                                    {"--statistic-output", "stats_" + tag +
".xml"},
                                                    {"--edgedata-output", "edge_" + tag +
".xml"},
                                                    {"--queue-output", "queue_" + tag +
".xml"};

    TrafficPulse::TrafficSystem system(1.0, sumo_config, net_config, bus,
is_custom_scenario, gui, outs);

    system.run();

    const std::string file_trip_info = "tripinfo_" + tag + ".xml";
    const std::string file_summary = "summary_" + tag + ".xml";
    const std::string file_stats = "stats_" + tag + ".xml";
    const std::string file_edge = "edge_" + tag + ".xml";
    const std::string file_queue = "queue_" + tag + ".xml";

    TrafficPulse::Metrics metrics =
TrafficPulse::XML::parseTripInfo(file_trip_info);
    TrafficPulse::XML::enrichFromStats(file_stats, metrics);
    TrafficPulse::XML::enrichFromSummary(file_summary, metrics);
    TrafficPulse::XML::enrichFromQueue(file_queue, metrics);
    TrafficPulse::XML::enrichFromEdge(file_edge, metrics);

    return metrics;
}

void writeMetricsCsv(const std::string& out_csv,
                    const std::string& scenario_name,
                    bool is_custom_scenario,
                    const TrafficPulse::Metrics& metrics) {
    if(std::filesystem::path csv_file_path(out_csv);
csv_file_path.has_parent_path()) {
        std::filesystem::create_directories(csv_file_path.parent_path());
    }

    bool exists = std::ifstream(out_csv).good();
    std::ofstream fo(out_csv, std::ios::app);
    if(!fo)
        throw std::runtime_error("Cannot open CSV: " + out_csv);

    if(!exists) {
        fo << "timestamp,scenario,mode,"
"avg_travel_time,avg_waiting_time,avg_speed,throughput,num_stops,avg_queue_length,
"
        "time_loss,halted_peak,density_peak,queue_peak\n";
    }

    fo << currentTimestamp() << ',' << scenario_name << ',' << (is_custom_scenario
? "custom" : "base") << ','
        << metrics.avg_travel_time << ',' << metrics.avg_waiting_time << ',' <<
metrics.avg_speed << ','
        << metrics.throughput << ',' << metrics.num_stops << ',' <<

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				228
Змн.	Арк.	№ докум.	Підпис	Дата		

```

metrics.avg_queue_length << ',' << metrics.time_loss
    << ',' << metrics.halted_peak << ',' << metrics.density_peak << ',' <<
metrics.queue_peak << '\n';
}

void writeMetricsCsv(std::ofstream& fo,
    const std::string& scenario_name,
    const bool is_custom_scenario,
    const TrafficPulse::Metrics& metrics) {
    fo << currentTimestamp() << ',' << scenario_name << ',' << (is_custom_scenario
? "custom" : "base") << ','
    << metrics.avg_travel_time << ',' << metrics.avg_waiting_time << ',' <<
metrics.avg_speed << ','
    << metrics.throughput << ',' << metrics.num_stops << ',' <<
metrics.avg_queue_length << ',' << metrics.time_loss
    << ',' << metrics.halted_peak << ',' << metrics.density_peak << ',' <<
metrics.queue_peak << '\n';
}

#include "scripts/Scenario.h"
#include <fstream>
#include <iostream>
#include <set>
#include <sstream>
#include <vector>

static std::vector<std::string> split(const std::string& s, char delim) {
    std::vector<std::string> out;
    std::stringstream ss(s);
    std::string item;
    while(std::getline(ss, item, delim))
        out.push_back(item);
    return out;
}

int main(const int argc, char** argv) {
    TrafficPulse::Log::init();

    bool gui = false;
    bool run_custom = false;
    std::string scenario_name;
    std::string sumo_config;
    std::string net_config;
    std::string out_csv = "../analysis/comparison.csv";

    for(int i = 1; i < argc; ++i) {
        std::string a = argv[i];
        if(a == "--gui") {
            gui = true;
        }
        else if(a == "--custom") {
            run_custom = true;
        }
        else if((a == "--output" || a == "-o") && i + 1 < argc) {
            out_csv = argv[++i];
        }
        else if(a == "--scenario" && i + 1 < argc) {
            scenario_name = argv[++i];
        }
        else if(a == "--sumoCfg" && i + 1 < argc) {
            sumo_config = argv[++i];
        }
    }
}

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				229
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        else if(a == "--netCfg" && i + 1 < argc) {
            net_config = argv[++i];
        }
    }

    if(scenario_name.empty() || sumo_config.empty() || net_config.empty()) {
        std::cerr << "Usage: " << argv[0] << " --scenario <name>"
            << " --sumoCfg <path>"
            << " --netCfg <path>"
            << " [--custom] [--gui] [-o <out.csv>]\n";
        return 1;
    }

    std::set<std::pair<std::string, bool>> done;
    if(std::ifstream ifs(out_csv); ifs) {
        std::string line;
        std::getline(ifs, line);
        while(std::getline(ifs, line)) {
            auto parts = split(line, ',');
            if(parts.size() >= 2) {
                bool isCustom = (parts[2] == "custom");
                done.emplace(parts[1], isCustom);
            }
        }
    }

    {
        if(std::filesystem::path p(out_csv); p.has_parent_path())
            std::filesystem::create_directories(p.parent_path());
    }

    bool need_header = !std::ifstream(out_csv).good();
    std::ofstream ofs(out_csv, std::ios::app);
    if(!ofs) {
        std::cerr << "Cannot open CSV for writing: " << out_csv << "\n";
        return 1;
    }
    if(need_header) {
        ofs << "timestamp,scenario,mode,"
            << "avg_travel_time,avg_waiting_time,avg_speed,throughput,num_stops,avg_queue_length,"
            << "time_loss,halted_peak,density_peak,queue_peak\n";
    }

    spdlog::info("Running {} for {}{}...", run_custom ? "custom" : "base",
        scenario_name, gui ? " (GUI)" : "");

    if(!run_custom) {
        if(!done.contains({scenario_name, false})) {
            auto m = runSumoAndCollect(scenario_name, sumo_config, net_config,
                false, gui);
            writeMetricsCsv(ofs, scenario_name, false, m);
        }
        else {
            spdlog::info("Skipping base for {} (already have results)",
                scenario_name);
        }
    }
    else {
        auto m = runSumoAndCollect(scenario_name, sumo_config, net_config, true,
            gui);
        writeMetricsCsv(ofs, scenario_name, true, m);
    }

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				230
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    spdlog::info("Done. Results in {}", out_csv);
    return 0;
}

#pragma once

#ifndef SCENARIO_H
#define SCENARIO_H

#include <TrafficPulseLibrary>
#include <chrono>
#include <ctime>
#include <string>

/**
 * @brief Runs a SUMO + TrafficPulse simulation, either base or with your
 * algorithm.
 *
 * @param scenario_name identifier, for example, "low_traffic"
 * @param sumo_config path to .sumocfg
 * @param net_config path to .net.xml
 * @param is_custom_scenario true = run with pulse algo, false = base run
 * @param gui true = show SUMO GUI, false = run in background
 *
 * @return collected metrics
 */
TrafficPulse::Metrics runSumoAndCollect(const std::string& scenario_name,
                                         const std::string& sumo_config,
                                         const std::string& net_config,
                                         bool is_custom_scenario = false,
                                         bool gui = false);

/**
 * @brief Appends a metrics row to CSV, creates a header if needed.
 *
 * @param out_csv CSV path
 * @param scenario_name identifier, for example, "low_traffic"
 * @param is_custom_scenario true = custom scenario with pulse algo, false =
 * default with base SUMO algo
 * @param metrics the metrics to write
 */
void writeMetricsCsv(const std::string& out_csv,
                    const std::string& scenario_name,
                    bool is_custom_scenario,
                    const TrafficPulse::Metrics& metrics);

/** @copydoc writeMetricsCsv(const std::string&,const std::string&,bool, const
 * TrafficPulse::Metrics&) */
void writeMetricsCsv(std::ofstream& fo,
                    const std::string& scenario_name,
                    bool is_custom_scenario,
                    const TrafficPulse::Metrics& metrics);

/**
 * @brief Current wall-time stamp in ISO-8601 format.
 *
 * @return string with the current local time in ISO-8601 format.
 */
inline std::string currentTimestamp() {
    const auto now = std::chrono::system_clock::now();

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		231

```

    const std::time_t time_t = std::chrono::system_clock::to_time_t(now);
    const std::tm time = *std::localtime(&time_t);
    std::ostringstream oss;
    oss << std::put_time(&time, "%Y-%m-%dT%H:%M:%S");
    return oss.str();
}

#endif // SCENARIO_H

#pragma once

#ifndef PLOT_H
#define PLOT_H

#include <matplot/matplot.h>
#include <numeric>
#include <spdlog/spdlog.h>
#include <string>
#include <vector>

/**
 * @class Plot
 * @brief Thin wrapper around matplot++ providing one-shot comparison charts.
 *
 * The class draws a line-chart that compares baseline vs. custom metrics and
 * stores the PNG on disk.
 *
 * @ref compare(), destroy.
 */
class Plot {
public:
    /**
     * @brief Ctor.
     * @param out_png Absolute/relative path where the PNG will be saved.
     */
    explicit Plot(std::string out_png)
        : m_output_png_path{std::move(out_png)} { }

    /**
     * @brief Draw two overlaid line-series base vs. custom and persist PNG.
     *
     * @param title Chart title.
     * @param y_label Label for Y-axis units.
     * @param labels X-axis textual labels (one per data-point).
     * @param base Baseline data (same size as @p labels).
     * @param custom Custom/Pulse data (same size as @p labels).
     */
    void compare(const std::string& title,
                const std::string& y_label,
                const std::vector<std::string>& labels,
                const std::vector<double>& base,
                const std::vector<double>& custom) const {
        namespace mp = matplot;

        if(labels.empty()) {
            spdlog::warn("compare: no data - skipping '{}'", title);
            return;
        }

        if(base.empty() || custom.empty()) {
            spdlog::warn("No data for '{}'", title);
            return;
        }
    }
};

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ІЗ	Арк.
		Власенко О.В.				232
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    const auto figure_handle = mp::figure(true);
    figure_handle->size(1600, 900);
    const auto axes_handle = mp::gca();
    axes_handle->hold(mp::on);

    std::vector<double> x_indices(labels.size());
    std::iota(x_indices.begin(), x_indices.end(), 0.0);

    const auto line_base = mp::plot(x_indices, base);
    line_base->color("b").marker("o").display_name("base");

    const auto line_custom = mp::plot(x_indices, custom);
    line_custom->color("r").marker("s").display_name("custom");

    axes_handle->title(title);
    axes_handle->ylabel(y_label);
    mp::xticks(x_indices);
    mp::xticklabels(labels);
    mp::grid(mp::on);

    mp::legend()->location(mp::legend::general_alignment::topleft);
    mp::save(m_output_png_path);
}

private:
    std::string m_output_png_path; ///< Target PNG file written by @ref compare().
};

#endif //PLOT_H

# Common settings
BasedOnStyle: WebKit
TabWidth: 4
IndentWidth: 4
UseTab: Always
ColumnLimit: 120

---
Language: Cpp

DisableFormat: false
Standard: c++14

AccessModifierOffset: -4
AlignAfterOpenBracket: true
AlignConsecutiveAssignments: false
AlignConsecutiveDeclarations: false
AlignOperands: true
AlignTrailingComments: false
AllowAllParametersOfDeclarationOnNextLine: true
AllowShortBlocksOnASingleLine: Never
AllowShortCaseLabelsOnASingleLine: false
AllowShortFunctionsOnASingleLine: Empty
AllowShortIfStatementsOnASingleLine: false
AllowShortLoopsOnASingleLine: false
AlwaysBreakAfterDefinitionReturnType: None
AlwaysBreakAfterReturnType: None
AlwaysBreakBeforeMultilineStrings: false
BreakTemplateDeclarations: No
BinPackArguments: false

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				233
Змн.	Арк.	№ докум.	Підпис	Дата		

```

BinPackParameters: false

BreakBeforeBraces: Custom
BraceWrapping:
  AfterEnum : true
  AfterFunction : false
  AfterNamespace : true
  AfterStruct : true
  AfterUnion : true
  BeforeCatch : true
  BeforeElse : true
  IndentBraces : false
  AfterExternBlock : true
  SplitEmptyFunction : false
  SplitEmptyRecord : false
  SplitEmptyNamespace : true

BreakAfterJavaFieldAnnotations: true
BreakBeforeInheritanceComma: false
BreakBeforeBinaryOperators: None
BreakBeforeTernaryOperators: true
BreakConstructorInitializersBeforeComma: true
BreakStringLiterals: true

CompactNamespaces: false
ConstructorInitializerAllOnOneLineOrOnePerLine: false
ConstructorInitializerIndentWidth: 4
ContinuationIndentWidth: 4
Cpp11BracedListStyle: true
SpaceBeforeCpp11BracedList: false
DerivePointerAlignment: false
ExperimentalAutoDetectBinPacking: false
ForEachMacros: [ foreach, Q_FOREACH, BOOST_FOREACH ]
IndentCaseLabels: false
FixNamespaceComments: true
IndentWrappedFunctionNames: false
KeepEmptyLinesAtTheStartOfBlocks: true
MacroBlockBegin: ''
MacroBlockEnd: ''
JavaScriptQuotes: Double
MaxEmptyLinesToKeep: 1
NamespaceIndentation: None
ObjCBlockIndentWidth: 4
ObjCSpaceAfterProperty: true
ObjCSpaceBeforeProtocolList: true
PenaltyBreakBeforeFirstCallParameter: 19
PenaltyBreakComment: 300
PenaltyBreakFirstLessLess: 120
PenaltyBreakString: 1000

PenaltyExcessCharacter: 1000000
PenaltyReturnTypeOnItsOwnLine: 60
PointerAlignment: Left
SpaceAfterCStyleCast: false
SpaceBeforeAssignmentOperators: true
SpaceBeforeParens: Never
SpaceInEmptyParentheses: false
SpacesBeforeTrailingComments: 1
SpacesInAngles: false
SpacesInContainerLiterals: true
SpacesInCStyleCastParentheses: false
SpacesInParentheses: false

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				234
Змн.	Арк.	№ докум.	Підпис	Дата		

```

SpacesInSquareBrackets: false
SpaceAfterTemplateKeyword: true
SpaceBeforeInheritanceColon: true

SortUsingDeclarations: true
SortIncludes: CaseSensitive

# Comments are for developers, they should arrange them
ReflowComments: false

IncludeBlocks: Preserve
IndentPPDirectives: None

default:
  image: andriibabushko/sumo-libsumocpp:latest
  tags:
    - docker
  before_script:
    - git lfs pull

stages:
  - clang-format
  - test
  - build
  - deploy

clang_format_check:
  stage: clang-format
  image: ubuntu:22.04
  before_script:
    - apt-get update
    - apt-get install -y wget gnupg lsb-release software-properties-common git-lfs
lsb
    - wget https://apt.llvm.org/llvm.sh
    - chmod +x llvm.sh
    - ./llvm.sh 19
    - apt-get install -y clang-format-19
    - update-alternatives --install /usr/bin/clang-format clang-format
/usr/bin/clang-format-19 100
    - git lfs pull
  script:
    - make format-check

test_library:
  stage: test
  script:
    - make test-library
  only:
    - main
    - library

test_all:
  stage: test
  script:
    - make test-all
  only:
    - main

build-doxxygen:
  stage: build
  before_script:
    - apt-get update

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				235
Змн.	Арк.	№ докум.	Підпис	Дата		

```

- apt-get install -y doxygen graphviz
script:
- cd library
- doxygen Doxyfile
artifacts:
paths:
- library/html/
expire_in: 1 week
only:
- main
- library

pages:
stage: deploy
script:
- mv library/html public
artifacts:
paths:
- public
expire_in: 1 week
only:
- main
- library

cmake_minimum_required(VERSION 3.18)
project(TrafficPulseSystem)

set(CMAKE_CXX_STANDARD 20)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

list(APPEND CMAKE_PREFIX_PATH "/opt/homebrew/opt/yaml-cpp")

add_subdirectory(library)

include_directories(${CMAKE_CURRENT_SOURCE_DIR}/include)

file(MAKE_DIRECTORY ${CMAKE_BINARY_DIR}/config/sumo)
execute_process(COMMAND ${CMAKE_COMMAND} -E copy_directory
${CMAKE_CURRENT_SOURCE_DIR}/config/sumo
${CMAKE_CURRENT_BINARY_DIR}/config/sumo
)

find_package(Matplot++ REQUIRED)
find_package(Pugixml REQUIRED)
find_package(spdlog CONFIG REQUIRED)
find_library(LIBSUMOCPP sumocpp HINTS /usr/lib /usr/local/lib)
if(NOT LIBSUMOCPP)
message(FATAL_ERROR "libsumocpp not found! Please install SUMO with libsumo-
cpp support.")
endif()

add_executable(ScenarioRunner
src/scripts/ScenarioRunner.cpp
src/scripts/Scenario.cpp
)
target_link_libraries(ScenarioRunner PRIVATE ${LIBSUMOCPP} TrafficPulseLibrary
pugixml spdlog::spdlog)

add_executable(MetricsReporter
src/scripts/MetricsReporter.cpp
)
target_link_libraries(MetricsReporter PRIVATE

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				236
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        TrafficPulseLibrary Matplot++::matplot pugixml spdlog::spdlog
    )

include(FetchContent)
FetchContent_Declare(
    googletest
    GIT_REPOSITORY https://github.com/google/googletest.git
    GIT_TAG         v1.15.2
)
FetchContent_MakeAvailable(googletest)
option(BUILD_TESTING "Build unit tests" ON)
if(BUILD_TESTING)
    enable_testing()
    if(EXISTS "${CMAKE_CURRENT_SOURCE_DIR}/tests")
        add_subdirectory(tests)
    endif()
endif()

FROM ubuntu:22.04

ARG DEBIAN_FRONTEND=noninteractive

# Base system + SUMO dependencies
RUN apt-get update && \
    apt-get install -y \
    git cmake g++ build-essential python3-dev swig \
    libxerces-c-dev libfox-1.6-dev libgdal-dev libproj-dev \
    libgl2ps-dev libeigen3-dev libpugixml-dev \
    clang-format curl wget lsb-release gnupg git-lfs && \
    apt-get clean

# Clone and build SUMO
WORKDIR /opt
RUN git clone --recursive https://github.com/eclipse-sumo/sumo.git && \
    cd sumo && \
    cmake -B build -DCMAKE_INSTALL_PREFIX=/usr && \
    cmake --build build -j$(nproc) && \
    cmake --install build

ENV SUMO_HOME=/usr/share/sumo
ENV LD_LIBRARY_PATH=/usr/lib:/usr/local/lib:$LD_LIBRARY_PATH
WORKDIR /builds/project

BUILD_DIR      ?= build
CMAKE_BUILD_DIR ?= cmake-build-debug
VERSION        ?= 1.0.0
LOG            ?= debug
UNPACK         = ./scripts/unpack_scenarios.sh
OUTPUT_SCENARIOS_COMPARISON = ../analysis/comparison.csv
SAN_FLAGS      = -g -O1 -fsanitize=address,undefined -fno-omit-frame-pointer
CMAKE_EXTRA_ARGS ?=
SCENARIOS = \
    "Low Traffic
XS,config/sumo/scenarios/xs/low_traffic/osm.sumocfg,config/sumo/scenarios/xs/low_t
raffic/osm.net.xml" \
    "Low Traffic
S,config/sumo/scenarios/s/low_traffic/osm.sumocfg,config/sumo/scenarios/s/low_traf
fic/osm.net.xml" \
    "Medium Traffic
XS,config/sumo/scenarios/xs/medium_traffic/osm.sumocfg,config/sumo/scenarios/xs/me
dium_traffic/osm.net.xml" \
    "Medium Traffic

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		237

```

S,config/sumo/scenarios/s/medium_traffic/osm.sumocfg,config/sumo/scenarios/s/mediu
m_traffic/osm.net.xml" \
    "High Traffic
XS,config/sumo/scenarios/xs/high_traffic/osm.sumocfg,config/sumo/scenarios/xs/high
_traffic/osm.net.xml" \
    "High Traffic
S,config/sumo/scenarios/s/high_traffic/osm.sumocfg,config/sumo/scenarios/s/high_tr
affic/osm.net.xml"
#SCENARIOS = \
#    "High Traffic
S(56km/56km),config/sumo/scenarios/s/high_traffic/osm.sumocfg,config/sumo/scenario
s/s/high_traffic/osm.net.xml"
#SCENARIOS = \
#    "Medium Traffic
S(56km/56km),config/sumo/scenarios/s/medium_traffic/osm.sumocfg,config/sumo/scenar
ios/s/medium_traffic/osm.net.xml"

.PHONY: clean build asan-build test-library format format-check run-scenarios-
debug run-scenarios-gui report-metrics unpack-scenarios

# TODO: write help target

define RUN_CLANG_FORMAT
    @find . -type f \( -name "*.cpp" -o -name "*.h" \) \
        -not -path "./$(BUILD_DIR)/*" \
        -not -path "./cmake-build-debug/*" \
        -not -path "./library/cmake-build-debug/*" \
        -not -path "./library/test_package/build/*" \
        -exec clang-format -style=file $(1) {} +
endef

define RUN_BUILD
    @echo "❌ Configuring $(BUILD_TYPE), TESTING=$(BUILD_TESTING))"
    @mkdir -p $(BUILD_DIR)
    @cmake -S . -B $(BUILD_DIR) \
        -DINSTALL_GTEST=$(BUILD_TESTING) \
        -DINSTALL_GMOCK=$(BUILD_TESTING) \
        -DBUILD_TESTING=$(BUILD_TESTING) \
        -DCMAKE_BUILD_TYPE=$(BUILD_TYPE) \
        -DPROJECT_VERSION=$(VERSION) \
        $(CMAKE_EXTRA_ARGS)
    @echo "🏗 Building $(BUILD_TARGET)..."
    @cmake --build $(BUILD_DIR) $(BUILD_TARGET) --parallel
endef

clean:
    rm -rf $(BUILD_DIR)
    rm -rf $(CMAKE_BUILD_DIR)
    rm -rf library/$(BUILD_DIR)
    rm -rf library/$(CMAKE_BUILD_DIR)
    rm -f analysis/*.png analysis/*.md analysis/*.pdf

test-library:
    mkdir -p $(BUILD_DIR)
    cd $(BUILD_DIR) && cmake -DBUILD_TESTING=ON .. && make
    cd $(BUILD_DIR)/library && ctest -L library --output-on-failure

format:
    @command -v clang-format >/dev/null 2>&1 || { \
        echo "clang-format is not installed. Please install clang-format and try
again."; \
        exit 1; \
    }

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		238

```

    }
    @echo "Formatting all *.cpp and *.h files (excluding build directories)..."
    $(call RUN_CLANG_FORMAT, -i)
    @echo "✓ Formatting complete!"

format-check:
    @command -v clang-format >/dev/null 2>&1 || { \
        echo "clang-format is not installed. Please install clang-format and try again."; \
        exit 1; \
    }
    @echo "Checking formatting for all *.cpp and *.h files (excluding build directories)..."
    $(call RUN_CLANG_FORMAT, --dry-run --Werror)
    @echo "✓ Checking complete!"

build: BUILD_TYPE      := Release
build: BUILD_TESTING := ON
build:
    $(call RUN_BUILD)

SAN_FLAGS = -g -O1 -fsanitize=address,undefined -fno-omit-frame-pointer

asan-build: BUILD_TYPE      := RelWithDebInfo
asan-build: BUILD_TESTING   := OFF
asan-build: CMAKE_EXTRA_ARGS := \
    -DCMAKE_CXX_FLAGS_RELWITHDEBINFO="$(SAN_FLAGS)" \
    -DCMAKE_EXE_LINKER_FLAGS_RELWITHDEBINFO="$(SAN_FLAGS)"
asan-build:
    $(call RUN_BUILD)

run-scenarios-debug: asan-build
    @echo "🔍 Running all scenarios... (log=$(LOG))"
    cd $(BUILD_DIR) && \
    for sc in $(SCENARIOS); do \
        IFS=','; set -- $$sc; \
        NAME=$$1; SCFG=$$2; NCFG=$$3; \
        env TP_LOG=$(LOG) \
        gdb --args ./ScenarioRunner \
            --scenario "$$NAME" \
            --sumoCfg "$$SCFG" \
            --netCfg "$$NCFG" \
            --output "$$OUTPUT_SCENARIOS_COMPARISON"; \repo
    env TP_LOG=$(LOG) \
    gdb --args ./ScenarioRunner \
        --scenario "$$NAME" \
        --sumoCfg "$$SCFG" \
        --netCfg "$$NCFG" \
        --output "../comparison.csv" \
        --custom; \
    done
    @echo "✓ All scenarios completed."

run-scenarios: build
    @echo "🔍 Running all scenarios... (log=$(LOG))"
    cd $(BUILD_DIR) && \
    for sc in $(SCENARIOS); do \
        IFS=','; set -- $$sc; \
        NAME=$$1; SCFG=$$2; NCFG=$$3; \
        env TP_LOG=$(LOG) ./ScenarioRunner \
            --scenario "$$NAME" \

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				239
Змн.	Арк.	№ докум.	Підпис	Дата		

```

--sumoCfg "$$SCFG" \
--netCfg "$$NCFG" \
--output "../analysis/comparison.csv"; \
env TP_LOG=$(LOG) ./ScenarioRunner \
--scenario "$$NAME" \
--sumoCfg "$$SCFG" \
--netCfg "$$NCFG" \
--output "../analysis/comparison.csv" \
--custom; \
done
@echo "✔ All scenarios completed."

run-scenarios-gui: build
@echo "🖱 Running all scenarios with GUI..."
cd $(BUILD_DIR) && \
for sc in $(SCENARIOS); do \
IFS=','; set -- $$sc; \
NAME=$$1; SCFG=$$2; NCFG=$$3; \
TP_LOG=$(LOG) ./ScenarioRunner \
--gui \
--scenario "$$NAME" \
--sumoCfg "$$SCFG" \
--netCfg "$$NCFG" \
--output "../analysis/comparison.csv"; \
TP_LOG=$(LOG) ./ScenarioRunner \
--gui \
--scenario "$$NAME" \
--sumoCfg "$$SCFG" \
--netCfg "$$NCFG" \
--output "../analysis/comparison.csv" \
--custom; \
done
@echo "✔ All GUI scenarios completed."

report-metrics: build
@echo "📊 Generating charts and Markdown summary"
cd $(BUILD_DIR) && ./MetricsReporter
@if command -v pandoc >/dev/null 2>&1; then \
if command -v xelatex >/dev/null 2>&1; then \
echo "📄 Converting analysis/report.md → analysis/TrafficPulseReport.pdf"; \
pandoc analysis/report.md \
--resource-path=analysis \
--pdf-engine=pdflatex \
-H figure-force.tex \
-V graphics=true \
-o analysis/TrafficPulseReport.pdf && \
echo "✔ PDF ready: analysis/TrafficPulseReport.pdf"; \
else \
echo "⚠ xelatex not found; skipping PDF generation. Install a TeX engine."; \
fi; \
else \
echo "⚠ pandoc not found; skipping PDF generation. Install pandoc."; \
fi

unpack-scenarios:
@echo "-> Unpacking all scenarios..."
bash $(UNPACK)

```

		Бабушко А.С.			ІПЗ.КР.Б.-121-25-ПЗ	Арк.
		Власенко О.В.				240
Змн.	Арк.	№ докум.	Підпис	Дата		