

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 «Інженерія програмного забезпечення»)

на тему:

«Вебплатформа для моделювання складних фізичних процесів у
браузері»

Виконав студент групи ІПЗ-22-1

ЗАХАРОВ Іван Андрійович

Керівник роботи:

ВАКАЛЮК Тетяна Анатоліївна

Рецензент:

КОНЦИНДАЙЛО Андрій Михайлович

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
«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: ЗАХАРОВ Іван Андрійович

Керівник роботи: ВАКАЛЮК Тетяна Анатоліївна

Тема роботи: «Вебплатформа для моделювання складних фізичних процесів у браузері»,

затверджена Наказом закладу вищої освіти від «27» лютого 2026 р., №92/с

Вихідні дані для роботи: процес моделювання фізичних явищ у браузерному середовищі з реалізацією методики «Передбачай – Спостерігай – Пояснюй», що поєднує тривимірну візуалізацію симуляцій з інструментами управління навчальним процесом для викладачів та учнів закладів загальної середньої освіти.

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Вакалюк Т. А.	28.02.2026р.	28.02.2026р.
2	Вакалюк Т. А.	28.02.2026р.	28.02.2026р.
3	Вакалюк Т. А.	28.02.2026р.	28.02.2026р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з інтерактивної вебплатформи для моделювання фізичних процесів та пояснювальної записки. Пояснювальна записка до випускної роботи містить 73 сторінки, 37 ілюстрацій та 16 таблиць.

Метою роботи є проектування та розробка вебплатформи для інтерактивного моделювання складних фізичних процесів у браузері з реалізацією методики «Передбачай – Спостерігай – Пояснюй» та інструментами управління навчальним процесом.

Платформа забезпечує тривимірну візуалізацію фізичних симуляцій у шести категоріях: механіка, електромагнетизм, хвилі, термодинаміка, гравітація та квантова фізика. Реалізовано дванадцять інтерактивних модулів, що охоплюють явища від руху маятника до задачі трьох тіл і конструктора атома.

Система підтримує рольову модель доступу з ролями учня, викладача, співавтора та адміністратора. Викладач може створювати класи, формувати завдання чотирьох типів та оцінювати роботи учнів. Учень проходить структурований цикл навчання, отримує автоматичну або ручну оцінку та завантажує PDF-звіт результату. Розроблена платформа може бути впроваджена у навчальний процес закладів загальної середньої освіти та використовуватися репетиторами при підготовці учнів до контрольних робіт і зовнішнього незалежного оцінювання.

КЛЮЧОВІ СЛОВА: ОСВІТНІ ТЕХНОЛОГІЇ, ВЕБПЛАТФОРМА, ФІЗИЧНА СИМУЛЯЦІЯ, 3D-ВІЗУАЛІЗАЦІЯ, РОЕ-МЕТОДИКА, ANGULAR, BABYLON.JS.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		3

ABSTRACT

The bachelor's qualification work consists of an interactive web platform for physics process simulation and an explanatory note. The explanatory note contains 73 pages, 37 figures and 16 tables.

The aim of the work is the design and development web platform for interactive simulation of complex physical processes in the browser, with implementation of the "Predict – Observe – Explain" methodology and tools for managing the educational process.

The platform provides three-dimensional visualization of physical simulations across six categories: mechanics, electromagnetism, waves, thermodynamics, gravity and quantum physics. Twelve interactive modules have been implemented, covering phenomena ranging from pendulum motion to the three-body problem and an atom constructor.

The system supports a role-based access model with student, teacher, co-author and administrator roles. The teacher can create classes, design assignments of four types and grade student submissions. The student goes through a structured learning cycle, receives an automatic or manual grade and downloads a PDF report of the result. The developed platform can be introduced into the educational process of general secondary education institutions and used by private tutors when preparing students for tests and external independent assessment.

KEY WORDS: EDUCATIONAL TECHNOLOGIES, WEB PLATFORM, PHYSICS SIMULATION, 3D VISUALIZATION, POE METHODOLOGY, ANGULAR, BABYLON.JS.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП	7
РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ВЕБПЛАТФОРМИ МОДЕЛЮВАННЯ ФІЗИЧНИХ ПРОЦЕСІВ	11
1.1. Постановка задачі	11
1.2. Аналіз аналогів програмного продукту	12
1.3. Вибір архітектури вебплатформи для моделювання фізичних процесів	17
1.4. Обґрунтування вибору інструментальних засобів	19
Висновки до першого розділу	22
РОЗДІЛ 2 ПРОЕКТУВАННЯ ВЕБПЛАТФОРМИ МОДЕЛЮВАННЯ ФІЗИЧНИХ ПРОЦЕСІВ	24
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	24
2.2. Розробка бази даних системи	35
2.3. Проектування та реалізація алгоритмів роботи системи	44
2.4. Реалізація функціоналу платформи	50
Висновки до другого розділу	57
РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ МОДЕЛЮВАННЯ ФІЗИЧНИХ ПРОЦЕСІВ	59
3.1. Порядок встановлення та налаштування параметрів платформи	59
3.2. Структура інтерфейсу платформи для моделювання складних фізичних процесів у браузері	65
3.3. Тестування платформи	76
Висновки до третього розділу	83
ВИСНОВКИ	85
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	86
ДОДАТКИ	90

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – прикладний програмний інтерфейс

БД – база даних

ПЗ – програмне забезпечення

СУБД – система управління базами даних

SPA – односторінковий застосунок

JWT – токен автентифікації

REST – архітектурний стиль побудови програмних інтерфейсів

ORM – об'єктно-реляційне відображення

CRUD – основні операції з даними: створення, читання, оновлення, видалення

DTO – об'єкт передачі даних

WebGL – вебграфічний інтерфейс для рендерингу 3D-графіки

SMTP – протокол передачі електронної пошти

РОЕ – методика активного навчання «Передбачай – Спостерігай – Пояснюй»

ПСП – методика «Передбачай – Спостерігай – Пояснюй»

GDPR – Загальний регламент захисту персональних даних

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Цифрові інструменти у школі давно перестали бути чимось новим. Після 2020 року дистанційне, а згодом і змішане навчання увійшли в українську освіту дуже швидко – і, по суті, вже залишилися. І вчителі, і учні встигли звикнути працювати з онлайн-матеріалами. Питання тепер не в тому, чи користуватись цифровими ресурсами взагалі, а в тому – якими саме.

Бо є різниця. Відкрити PDF підручника – це фактично та сама книжка, лише на екрані. А ось інструмент, у якому можна запустити процес, змінити параметр і побачити, що з цього вийшло – це вже зовсім інший рівень роботи з матеріалом. Особливо помітно це у фізиці: її складно вивчити просто з тексту. Щоб реально зрозуміти, як поводить себе маятник чи від чого залежить дальність польоту тіла, недостатньо побачити формулу. Явище треба спостерігати, а краще – мати змогу самому покрутити параметри.

З обладнанням у звичайній українській школі – окрема історія. У моїй, наприклад, кабінет фізики був, і навіть з якимись приладами, тільки половиною з них уже давно ніхто не користувався: щось зламалось, щось не повне. У знайомих з інших шкіл картина приблизно така ж. Десять років стоять установки ще радянських часів, де не лишилось майже нічого. Підручник цю діру теж не латає – намальована у книжці стрілка швидкості й реальний рух тіла дають дуже різне розуміння того, що відбувається.

Тому веб-симуляції, які запускаються прямо у браузері й не вимагають нічого окрім інтернету, – це доволі робоча штука. Не модна обгортка для уроку, а нормальний спосіб показати дослід, який інакше у школі показати немає на чому.

Платформа, про яку йдеться у цій роботі, побудована з розрахунком на дві сторони. Учень тут радше дослідник, ніж читач: заходить, відкриває симуляцію, крутить параметри, дивиться, що змінилось, при потребі переробляє – ніхто не ставить двійку за невдалу спробу. Для вчителя система виглядає інакше: створити клас, додати туди учнів, зібрати завдання з потрібних симуляцій, а потім бачити, хто що виконав – без перевірки кожної

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

роботи руками. Технічно – це звичайний веб-додаток, який відкривається з будь-якого пристрою з браузером. Ставити нічого не треба, ні на телефон учневі, ні на ноутбук вчителю.

Сценарій учня в системі виглядає так. Він відкриває каталог, обирає тему – наприклад, рух тіла, кинутого під кутом до горизонту – і запускає симуляцію. Працює з нею за методикою РОЕ: «Передбачай – Спостерігай – Пояснюй» [1]. Спочатку дає прогноз (куди полетить тіло, якщо кут зробити більшим?), потім запускає експеримент і дивиться, що сталося насправді. У кінці отримує підсумок із порівнянням того, що передбачив, з тим, що показала модель.

Вчитель у системі має ширший набір можливостей. Це і створення власного класу з учнями, і формування завдань – чотири різні типи на вибір, від простого тесту до повного циклу РОЕ – і перегляд того, як справляється кожен учень окремо. Підрахунок балів додаток бере на себе, тож час, який зазвичай йшов на перевірку робіт, лишається на нормальне пояснення матеріалу – а це, по суті, і є робота вчителя.

Метою цієї випускної кваліфікаційної роботи бакалавра є розробити веб-платформу, на якій складні фізичні процеси можна моделювати прямо у вікні браузера. Платформа має реалізовувати методику «Передбачай – Спостерігай – Пояснюй» і водночас містити інструменти, які дозволяють вчителю керувати навчальним процесом – формувати класи, видавати завдання та переглядати результати.

Для досягнення мети потрібно виконати наступні завдання:

1. Проаналізувати існуючі аналоги платформ для симуляції фізичних процесів та виявити їх недоліки.
2. Обрати архітектуру системи та обґрунтувати вибір технологічного стеку для роботи з 3D-графікою у браузері.
3. Спроекувати базу даних та інформаційне забезпечення платформи.
4. Розробити модулі симуляцій у шести категоріях: механіка, електромагнетизм, хвилі, термодинаміка, гравітація та квантова фізика.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

5. Реалізувати ролі користувачів і систему завдань з автоматичною та ручною перевіркою.

6. Протестувати готовий продукт та оцінити його продуктивність.

Об'єктом дослідження є процес інтерактивного моделювання фізичних явищ у браузерному середовищі з використанням методики «Передбачай – Спостерігай – Пояснюй».

Предметом дослідження є методи та засоби реалізації тривимірних фізичних симуляцій у вигляді інтерактивної освітньої вебплатформи. Дослідження включає аналіз підходів до організації навчання через веб, вибір інструментів для 3D-рендерингу в браузері, проектування архітектури багатокористувацької системи з ролями та реалізацію автоматичного оцінювання навчальних завдань.

Методи дослідження. В роботі використано методи системного аналізу для дослідження предметної області, виявлення вимог користувачів та порівняння наявних аналогів освітніх платформ; методи математичного моделювання для опису фізичних процесів та реалізації відповідних симуляційних модулів на платформі; методи об'єктно-орієнтованого проектування і сучасного вебпрограмування для побудови архітектури системи, проектування бази даних та написання серверної й клієнтської частин платформи.

Практичне значення одержаних результатів полягає у тому, що розроблена вебплатформа може бути безпосередньо впроваджена у навчальний процес закладів загальної середньої освіти для проведення уроків та лабораторних робіт з фізики, а також використовуватися репетиторами в індивідуальній роботі з учнями при підготовці до контрольних робіт, державної підсумкової атестації та зовнішнього незалежного оцінювання. Використання платформи дозволяє автоматизувати перевірку знань учнів, впровадити методику «Передбачай – Спостерігай – Пояснюй» у щоденну роботу вчителя та значно підвищити рівень засвоєння матеріалу завдяки

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

переходу від пасивного сприйняття теорії до активної взаємодії з тривимірними моделями фізичних явищ.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ВЕБПЛАТФОРМИ МОДЕЛЮВАННЯ ФІЗИЧНИХ ПРОЦЕСІВ

1.1. Постановка задачі

Моделювання фізичних явищ – це відтворення поведінки реальних процесів у віртуальному середовищі засобами математики та комп'ютерної графіки, тобто спосіб побачити те, що складно або неможливо показати в умовах звичайної аудиторії. Раніше такі задачі вирішувалися через настільні застосунки, які треба було встановлювати, а сучасні вебтехнології дозволяють запустити все це в одному вікні браузера. Окрім візуалізації, важливу роль відіграє підхід до навчання. У роботі використовується методика «Передбачай – Спостерігай – Пояснюй» (РОЕ): учень спочатку прогнозує результат, потім запускає симуляцію та спостерігає за реальним перебігом, а наприкінці пояснює, чому його передбачення збіглося або не збіглося з дійсністю. Такий цикл змушує мислити, а не пасивно дивитися на анімацію [1].

Функціонально платформа складається з кількох програмних модулів. Модуль симуляцій є ядром системи: тут користувач працює з фізичними моделями, налаштовує параметри через повзунки, проходить РОЕ-цикл і зберігає власну конфігурацію як іменованій сценарій. Реалізовано порівняння передбачених і фактичних значень з підрахунком похибки та вимогу мінімального часу спостереження, що унеможливорює формальне проходження завдання без вникання у процес.

Модуль навчального контенту відповідає за каталог симуляцій. У системі шість категорій фізики – механіка, електромагнетизм, хвилі, термодинаміка, гравітація та квантова фізика. Кожна симуляція має рівень складності, теги та опис, передбачено фільтрацію за категорією і рівнем та пошук за назвою.

Модуль класів та завдань дозволяє вчителю створити віртуальний клас, отримати восьмисимвольний код запрошення і поділитися ним з учнями. Завдання бувають чотирьох типів: РОЕ-передбачення з автоматичною

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

оцінкою за відносною похибкою; сценарій з ручною перевіркою вчителем; тест з варіантами відповідей; числова задача з допуском. Після здачі учень може завантажити PDF-звіт з результатом.

Модуль аналітики показує статистику виконання робіт. Учень бачить середній бал, відсоток виконаних завдань і теплову карту майстерності за категоріями фізики, а вчитель – статистику по класу, графік середніх балів за завданнями і динаміку оцінок конкретного учня в часі. Це дозволяє швидко знаходити тих, хто відстає.

Модуль користувачів відповідає за реєстрацію, вхід, профіль та ролі, яких у системі чотири – учень, вчитель, співавтор класу та адміністратор. Передбачено можливість подати заявку на зміну ролі, а також, відповідно до вимог GDPR, експортувати власні дані або видалити акаунт. Сповіщення надходять і всередині сайту, і на email. Адмін-панель дозволяє керувати акаунтами і переглядати журнал дій, що важливо для безпеки та аудиту.

Усі обчислення симуляцій виконуються на стороні клієнта, тож сервер не навантажується зайвими розрахунками, а платформа лишається легко масштабованою. Розробка системи охоплює проектування інтерфейсу з підтримкою світлої й темної тем і двох мов, побудову клієнтської частини на Angular 18, серверної частини на ASP.NET Core 9 у форматі REST API, системи авторизації через JWT-токени, бази даних PostgreSQL 16 через Entity Framework Core, а також реалізацію самих 3D-симуляцій на Babylon.js – найскладнішої частини роботи. Завершальним етапом є тестування: функціональне, візуальне та оцінка продуктивності на різних пристроях.

1.2. Аналіз аналогів програмного продукту

Перш ніж починати розробку власної платформи, варто розглянути наявні на ринку освітні інструменти для моделювання фізики. Це дозволяє перейняти вдалі рішення і врахувати слабкі місця. Нижче проаналізовано чотири найвідоміші аналоги, кожен з яких представляє окремий підхід до задачі.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

Першим і найвідомішим у світі є проект PhET Interactive Simulations від Університету Колорадо в Боулдері (рис. 1.1).

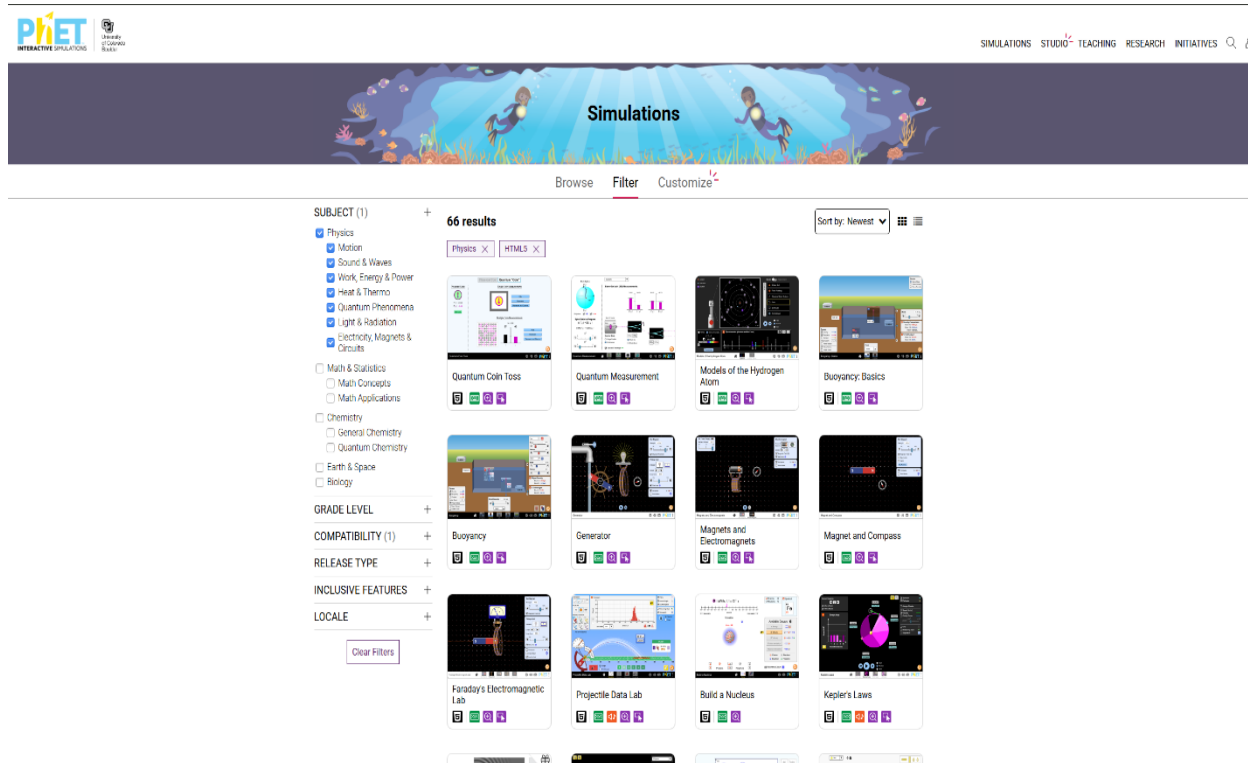


Рис. 1.1. PhET Interactive Simulations

PhET – це велика бібліотека безкоштовних симуляцій з фізики, хімії, біології та математики, що налічує понад 150 модулів і перекладена більш ніж сотнею мов, включно з українською [7]. Симуляції повністю переписані з Java на HTML5 і відкриваються у будь-якому сучасному браузері без плагінів. Проект має відкритий код, методичні рекомендації для вчителів і фільтрацію за віковими групами та предметами; з педагогічного боку симуляції виконані якісно, бо над ними працювали науковці й методисти [8]. Водночас платформа має суттєві обмеження. Майже всі симуляції двовимірні, чого достатньо для базових явищ на зразок маятника чи закону Ома, але замало для руху заряду в магнітному полі по складній траєкторії чи задачі трьох тіл. Інструменти для організації навчання відсутні повністю: немає класів, завдань, оцінювання та аналітики, тож вчитель змушений роздавати посилання і перевіряти результати на словах. РОЕ-методику теж не реалізовано – симуляції працюють як вільна пісочниця.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

Наступний аналог – комерційна платформа Labster (рис. 1.2) [27].

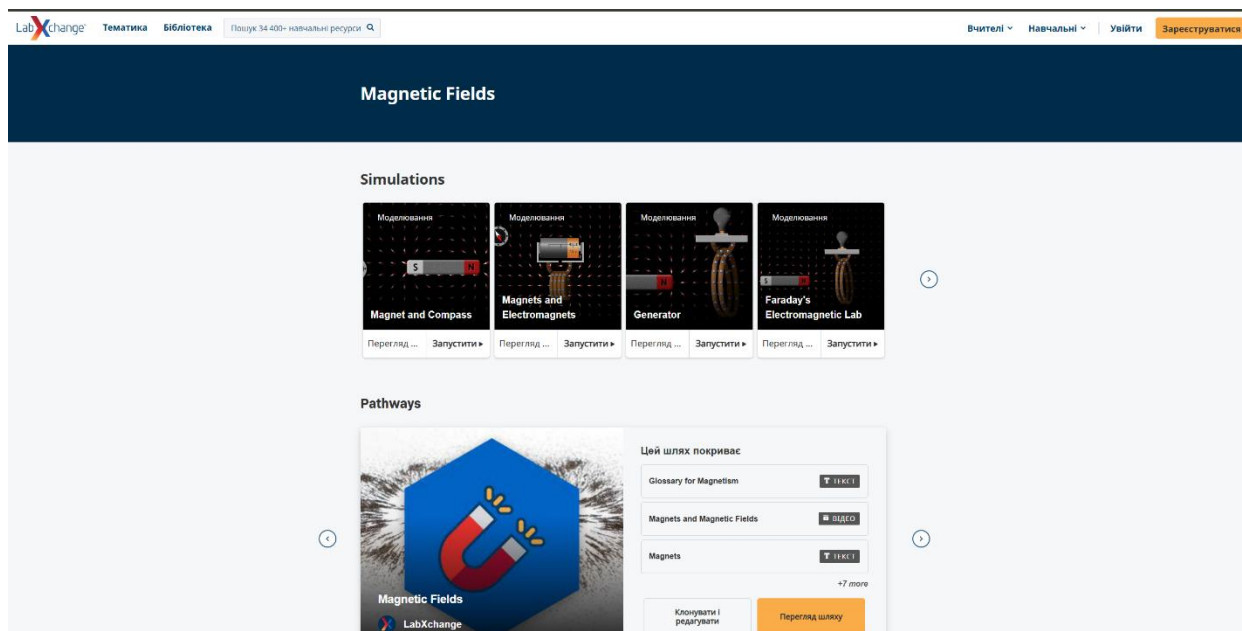


Рис. 1.2. Labster

Labster пропонує повноцінні 3D-лабораторії у вигляді ігрових сцен з персонажами, нарративом і покроковими інструкціями, де учень виконує завдання у віртуальному середовищі. Платформа інтегрується з популярними системами керування навчанням – Moodle, Canvas, Blackboard – і автоматично передає оцінки до журналу, а її аналітика фіксує час на кожному кроці та місця помилок. Головні недоліки Labster лежать в економічній площині: це закритий комерційний продукт з підпискою від кількох тисяч доларів на рік, що для більшості українських закладів освіти неприйнятно. Вчитель не може створити власну симуляцію або налаштувати наявну, бо обмежений готовим каталогом, який до того ж зміщений у бік біології, хімії й медицини, а класична фізика представлена слабо. Української локалізації немає. Окремо варто зазначити, що педагогічна логіка завдань у Labster побудована як лінійний сценарій із заздалегідь визначеною послідовністю кроків і фіксованим результатом, тому методику активного навчання на зразок РОЕ з обов'язковим прогнозуванням учнем результату та подальшим порівнянням передбачення з фактичним перебігом тут не реалізовано. У такій моделі учень фактично проходить інтерактивну інструкцію, а не самостійне дослідження, що зменшує

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

її цінність для формування навичок наукового мислення та критичного аналізу отриманих даних.

Третій аналог – Algodoo (рис. 1.3), програма-пісочниця для творчого моделювання [28].



Рис. 1.3. Algodoo

В Algodoo можна побудувати довільну двовимірну сцену з твердих тіл, з'єднати їх пружинами чи шарнірами, додати двигуни й рідину і спостерігати рух за реальними законами механіки. Інструмент популярний у школах через простоту: достатньо поставити об'єкт і запустити симуляцію, після чого програма самостійно прорахує траєкторії, зіткнення та взаємодію тіл. Підтримуються базові оптичні явища, гравітація і частково термодинаміка, а також побудова графіків швидкості, прискорення й енергії безпосередньо у сцені, що робить інструмент зручним для демонстрації законів збереження. Проте Algodoo – це настільна програма для Windows і macOS, яка не працює у браузері й недоступна на Chromebook та інших пристроях без можливості встановлення стороннього програмного забезпечення. Хмарне збереження відсутнє, тому обмін сценами відбувається лише через ручну передачу файлів, а інструментів для роботи з класом, видачі завдань чи відстеження успішності немає. Двовимірність обмежує застосування у багатьох розділах фізики, зокрема там, де принципово важлива просторова траєкторія руху.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

Останній аналог – Brilliant (рис. 1.4), освітня платформа для самонавчання [29].

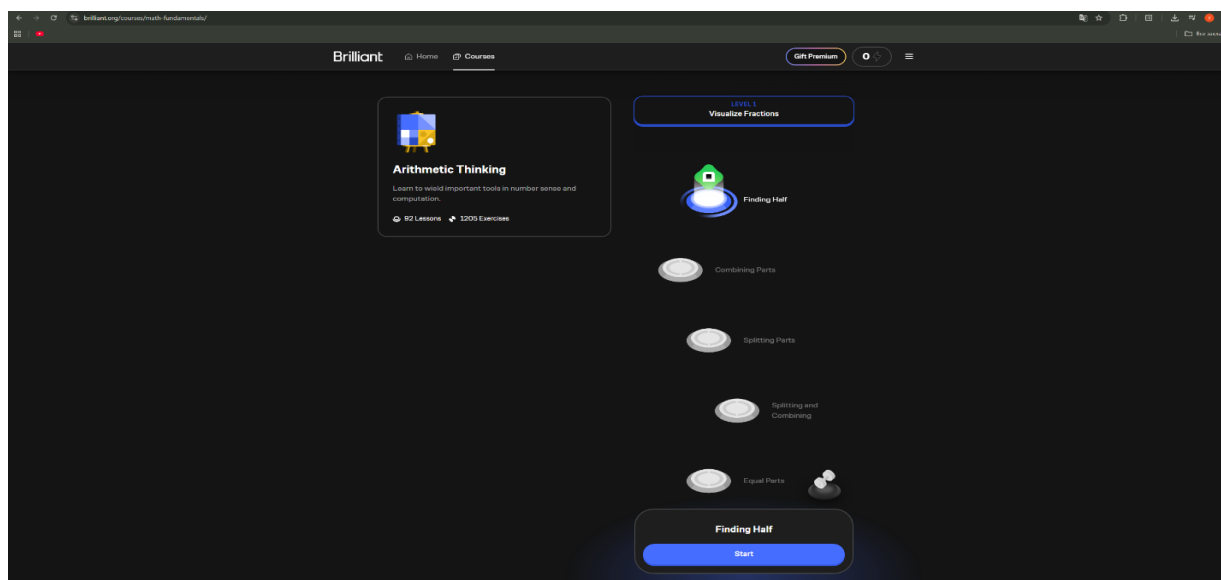


Рис. 1.4. Brilliant

Brilliant пропонує адаптивні курси з математики, фізики, інформатики й логіки з якісним контентом і гейміфікацією – серіями правильних відповідей, щоденними задачами та досягненнями. Платформа працює на десктопі й має мобільні застосунки для iOS та Android із синхронізацією прогресу, а її адаптивний підхід підбирає задачі під рівень користувача. Слабкі сторони полягають у тому, що Brilliant орієнтований на індивідуального користувача, а не на клас: вчитель не може створити групу, задати завдання чи відстежити успішність учнів. Симуляції тут переважно схематичні й двовимірні, радше ілюстративні. До того ж сервіс платний – підписка близько 15 доларів на місяць, що ускладнює масове впровадження у навчальному закладі.

Для узагальнення складемо порівняльну таблицю основних характеристик розглянутих платформ і розроблюваної системи (табл. 1.1).

Таблиця 1.1

Порівняння платформ

Характеристика	PhET	Labster	Algodoo	Brilliant	Розроблювана система
Повноцінна 3D-візуалізація	-	+	-	-	+

Продовження таблиці 1.1

Власні завдання вчителя	-	-	-	-	+
Аналітика успішності	-	+	-	+	+
Безкоштовний доступ	+	-	+	-	+
Вебзастосунок	+	+	-	+	+
Збереження сценаріїв	-	-	+	-	+
Українська локалізація	+	-	+	-	+
Підтримка всіх категорій фізики	+	-	-	+	+
РОЕ-методика навчання	-	-	-	-	+
Система класів та завдань	-	+	-	-	+

Основна проблема всіх розглянутих платформ – фрагментованість можливостей. Одні мають якісний контент, але не дають інструментів для вчителя; інші пропонують такі інструменти, проте є комерційно закритими з вузьким каталогом; треті взагалі не є вебсервісами. Жодна з платформ не використовує РОЕ-методику як основу навчального циклу.

Розроблювана платформа покликана усунути цей розрив і об'єднати в одній системі повноцінну 3D-візуалізацію, структурований РОЕ-цикл, інструменти для вчителя та безкоштовний відкритий доступ. Усі симуляції рендеряться на стороні клієнта засобами Babylon.js, що знімає навантаження з сервера; бекенд на ASP.NET Core відповідає лише за бізнес-логіку – користувачів, класи, завдання та оцінювання; PostgreSQL зберігає дані з підтримкою JSONB для гнучких структур. Український інтерфейс встановлено за замовчуванням з можливістю переключення на англійську.

1.3. Вибір архітектури вебплатформи для моделювання фізичних процесів

Вебплатформа моделювання фізичних процесів проектується як розподілена система за класичною клієнт-серверною моделлю. Логіку розділено на два рівні – фронтенд і бекенд, між якими відбувається обмін

даними через REST API у форматі JSON [16]. Такий підхід дозволяє розгортати клієнтську й серверну частини незалежно, а в перспективі – додати мобільний чи десктопний застосунок без переписування серверної логіки.

Клієнтська частина – це односторінковий вебзастосунок (SPA) на Angular 18, який виконує дві принципово різні задачі [17]. Перша – звичайна робота з даними: відображення списків, форми введення, навігація, графіки аналітики. Друга – рендеринг тривимірних фізичних симуляцій засобами рушія Babylon.js. Усі обчислення фізичних моделей відбуваються безпосередньо в браузері, тож сервер не бере участі в моделюванні, а лише зберігає параметри сценаріїв і результати виконання завдань. Завдяки такому розподілу навантаження платформа добре масштабується – кількість одночасних користувачів обмежена переважно ресурсами клієнтських пристроїв.

Серверна частина побудована на ASP.NET Core 9 і відповідає за бізнес-логіку, не пов'язану із симуляціями: автентифікацію та авторизацію, керування класами, завданнями та задачами, оцінювання робіт, формування PDF-звітів, надсилання email-сповіщень і ведення журналу адміністративних дій. Взаємодія з базою даних PostgreSQL відбувається через ORM Entity Framework Core, що забезпечує строгу типізацію та контроль міграцій схеми. Структурно серверна логіка поділена на функціональні блоки – автентифікації та профілю, навчального контенту, навчального процесу, аналітики та адміністрування; їхню детальну будову розглянуто у другому розділі.

Обмін даними між клієнтом і сервером відбувається через захищені HTTPS-запити з access-токеном у заголовку Authorization. Для тривалих сесій передбачено механізм refresh-токенів, а механізм TokenVersion гарантує миттєву інвалідацію активних сесій при зміні ролі або деактивації акаунту.

Кодова база організована як монорепозиторій на основі npm workspaces і об'єднує три артефакти – клієнтський застосунок Angular, серверний застосунок ASP.NET Core та спільний пакет physis-sdk з TypeScript-типами DTO. Така структура спрощує синхронізацію інтерфейсів між фронтом і

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

бекендом, бо зміна структури даних на сервері одразу відображається у спільному пакеті.

Передбачено три рівні доступу. Публічний доступний без автентифікації й охоплює лендинг, форми входу та інформаційні сторінки. Приватний доступний авторизованим користувачам і включає каталог симуляцій, профіль, власні сценарії та роботу з класами. Адміністративний доступний лише користувачам з відповідними правами. Розмежування реалізовано двома механізмами: на клієнті через RouterGuard, який перевіряє токен і роль ще до завантаження сторінки, а на сервері – через атрибут авторизації з конкретними вимогами до ролей.

1.4. Обґрунтування вибору інструментальних засобів

Технологічний стек розроблюваної вебплатформи формується з огляду на специфіку поставленої задачі та поєднує бекенд на ASP.NET Core 9, фронтенд на Angular 18, бібліотеку тривимірного рендерингу Babylon.js і реляційну базу даних PostgreSQL 16. Нижче послідовно обґрунтовано вибір кожного з цих компонентів та пояснено, які саме вимоги системи зумовили його прийняття.

Як серверна платформа обрано ASP.NET Core 9 з огляду на наявний рівень володіння мовою C#, який перевищує знання альтернативних серверних середовищ на основі Node.js чи Go, що дозволило одразу зосередитись на проектуванні бізнес-логіки замість вивчення нового технологічного стеку. Ключовим аргументом на користь саме цієї версії фреймворку стала вбудована підсистема ASP.NET Core Identity, яка дає змогу реалізувати автентифікацію через JWT, керування ролями та механізм скидання пароля кількома рядками конфігурації замість тижнів самостійної розробки відповідних компонентів [20]. Окремою перевагою є статична типізація C#, що виявляє цілий клас помилок невідповідності типів між шарами, особливо у DTO-об'єктах при передачі даних між контролером і сервісом, ще на етапі компіляції – до того, як вони перетворяться на збої під

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

час виконання. Кросплатформність фреймворку підтверджена практичним розгортанням у Linux-контейнері на Railway без жодних змін у коді відносно локального запуску під Windows [2].

Для клієнтської частини обрано фреймворк Angular 18 замість альтернативного React, причому остаточним аргументом стала технічна особливість роботи Angular зі змінами стану, що проявилась безпосередньо у процесі розробки. У попередніх версіях фреймворк використовував бібліотеку zone.js, яка патчила браузерні події та ініціювала перевірку змін у всьому дереві компонентів, і при паралельній роботі рендер-циклу Babylon.js це призводило до помітного падіння частоти кадрів у момент перетягування слайдерів керування симуляцією. У версії 18 з'явився механізм signals, який оновлює лише ті елементи DOM, що залежать від конкретного зміненого значення, і повністю усуває описану проблему без втручання у цикл рендерингу графічного рушія. Додатковою перевагою цієї версії є підтримка standalone-компонентів, які не потребують оголошення NgModule, що зменшує обсяг шаблонного коду й спрощує структуру проекту [3].

Серед бібліотек тривимірного рендерингу для браузера розглядалося дві основні альтернативи – Three.js і Babylon.js, – кожна з яких має зрілу спільноту й активну підтримку. Three.js є мінімальною графічною бібліотекою без вбудованої системи сцен, обробки подій і керованої камери, тож при її використанні відповідні підсистеми довелося б проектувати та реалізовувати власноруч. Натомість Babylon.js надає готовий граф сцени SceneGraph, керовану камеру ArcRotateCamera для зручної навігації навколо тривимірних об'єктів і клас DynamicTexture, що дозволяє накладати двовимірну графіку Canvas 2D API на поверхні моделей, тобто повністю покриває потреби розроблюваної платформи [4]. Вбудований у Babylon.js фізичний рушій на базі Cannon.js або Havok у підсумку не задіяний, оскільки кожна симуляція реалізує власний числовий інтегратор відповідних диференціальних рівнянь – стандартний rigid-body підхід не пристосований до хвильових процесів, дифузії газу чи квантових явищ на зразок фотоефекту.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

Доступ до реляційної бази даних організовано через об'єктно-реляційний маппер Entity Framework Core за підходом Code First, що дозволяє описувати схему безпосередньо мовою C# та автоматично генерувати скрипти міграцій, які зберігаються в одному репозиторії з кодом застосунку. Завдяки цьому будь-яка зміна структури таблиць проходить через створення нової міграції, яку можна відтворити на іншому середовищі або відкотити у разі помилки, а вся історія еволюції схеми залишається відстежуваною засобами системи контролю версій – за час розробки накопичилося одинадцять таких міграцій, що відображає поступовий характер проектування системи [6].

Окремо корисним для проекту виявився механізм глобальних запитальних фільтрів, який дозволяє один раз описати умову м'якого видалення `IsDeleted = false` у класі `DbContext` і автоматично застосовувати її до всіх подальших запитів без потреби дублювати цю перевірку у бізнес-логіці.

Як систему керування базою даних обрано PostgreSQL 16, причому головним аргументом стала повноцінна підтримка двійкового типу `JSONB`. Низка полів у розроблюваній платформі принципово не має фіксованої структури: параметри симуляцій, питання тестів та очікувані метрики POE-завдань відрізняються залежно від конкретного модуля, тому зберігання їх у звичайних реляційних таблицях вимагало б розширення схеми при додаванні кожного нового фізичного явища. Тип `JSONB` дозволяє обійти це обмеження, зберігаючи довільну структуру даних в одному полі та забезпечуючи при потребі ефективний пошук через додавання GIN-індексу за конкретним JSON-шляхом, що робить таку схему водночас гнучкою й продуктивною [5].

Окремої уваги заслуговує обрана модель безпеки сесій, побудована на парі JWT-токенів з власним механізмом миттєвої інвалідації. Стандартна специфікація JWT не передбачає можливості примусово припинити дію виданого токена до моменту його природного закінчення, що створює суттєвий ризик з точки зору адміністративного керування доступом [15]. Якщо адміністратор деактивує обліковий запис користувача, раніше виданий токен

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

залишається валідним і продовжує надавати доступ до системи до закінчення свого терміну дії. Для усунення цього обмеження впроваджено цілочисельне поле `tokenVersion`, яке одночасно зашите у `payload` токена та збережене у записі користувача в базі даних, а спеціальний `middleware` при кожному автентифікованому запиті звіряє ці два значення між собою. Будь-яка зміна ролі або деактивація акаунту супроводжується інкрементом лічильника `tokenVersion` у базі, що автоматично робить недійсними всі раніше видані токени конкретного користувача та звільняє систему від потреби вести окрему таблицю чорного списку відкликаних сесій.

Вимоги до клієнтського пристрою свідомо тримаються на мінімальному рівні, оскільки платформа повністю виконується у вікні браузера й не потребує встановлення стороннього програмного забезпечення. Для коректної роботи модулів тривимірної симуляції необхідна підтримка стандарту WebGL 2.0, наявна у Chrome версії 70 та вище, Firefox 68 та вище, Edge 79 та вище й Safari 15 та вище, а також не менше чотирьох гігабайтів оперативної пам'яті та будь-який сучасний графічний адаптер без особливих вимог до моделі [18]. Для обчислювально складніших симуляцій, зокрема задачі трьох тіл та молекулярної дифузії, рекомендується центральний процесор з тактовою частотою від 2,5 ГГц для забезпечення стабільної частоти кадрів. Серверна частина потребує двох процесорних ядер, чотирьох гігабайтів оперативної пам'яті та двадцяти гігабайтів дискового простору, а на рівні програмних залежностей – встановленого середовища виконання .NET 9 і системи керування базою даних PostgreSQL 16.

Висновки до першого розділу

У першому розділі сформовано постановку задачі та визначено функціональні вимоги до вебплатформи моделювання фізичних процесів. Описано шість програмних модулів – симуляцій, навчального контенту, класів і завдань, аналітики, користувачів та адміністрування, кожен з чітко

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

окресленою зоною відповідальності, а їхня взаємодія через REST API забезпечує цілісність системи.

Обґрунтовано вибір педагогічного підходу. Методика «Передбачай – Спостерігай – Пояснюй» перетворює симуляцію з пасивної демонстрації на повноцінний навчальний цикл, у якому учень активно мислить і зіставляє власні передбачення з фактичними результатами [10, 11, 14]. Цей підхід добре поєднується з тривимірною візуалізацією, бо наочність явищ підвищує точність прогнозів і робить навчання осмисленим [12, 13].

Аналіз ринку показав, що чотири провідні аналоги – PhET, Labster, Algodoo та Brilliant – мають сильні сторони, але жоден не закриває всі задачі одночасно: PhET сильний контентом, проте обмежений двовимірністю і не має інструментів для вчителя; Labster пропонує якісні 3D-лабораторії, але є дорогим закритим продуктом; Algodoo – настільна 2D-пісочниця без хмарної складової; Brilliant зосереджений на самонавчанні, а не на роботі з класом. Розроблювана платформа усуває цей розрив, об'єднуючи 3D-симуляції, РОЕ-цикл, інструменти для вчителя та аналітику в одній безкоштовній вебсистемі з українським інтерфейсом.

На основі вимог і аналізу аналогів обрано технологічний стек: ASP.NET Core 9 на сервері, Angular 18 зі standalone-компонентами та signals на клієнті, рушій Babylon.js для тривимірної графіки з обчисленнями на стороні клієнта, база даних PostgreSQL 16 з підтримкою JSONB для гнучкого контенту та захист доступу через ASP.NET Core Identity з парою JWT-токенів і механізмом TokenVersion.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

РОЗДІЛ 2 ПРОЕКТУВАННЯ ВЕБПЛАТФОРМИ МОДЕЛЮВАННЯ ФІЗИЧНИХ ПРОЦЕСІВ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Перш ніж переходити безпосередньо до реалізації, треба зафіксувати, що саме платформа має робити і хто з нею взаємодіятиме. На цьому етапі набагато дешевше виявити помилку в логіці чи прогалину у вимогах, ніж потім переписувати готові модулі. Тому в цьому параграфі зведено основні групи користувачів, перелічено функціональні й нефункціональні вимоги до системи, описано ключові варіанти використання та наведено об'єктну модель, на якій базується вся подальша робота.

Платформа Physis працює у браузері й не вимагає від користувача жодного встановлення. Доступ до функціональності залежить від того, авторизований користувач чи ні, а у разі авторизації – яку роль має його акаунт. Ролей у системі п'ять, причому одна з них – гість – формально не є роллю в Identity, а просто відповідає неавторизованому стану.

Гість може потрапити на лендинг, переглянути інформаційні сторінки «Про проект» та «Політика конфіденційності», перейти до каталогу симуляцій, відкрити будь-яку з них і запустити РОЕ-цикл без збереження результатів. Подати заявку на роль, створити сценарій чи приєднатися до класу гість не може – для цього треба зареєструватися. Реєстрація відбувається на сторінці /register із одразу вибраною стартовою роллю викладача або студента. Адмінську роль через звичайну реєстрацію отримати не можна – вона видається або через сід-механізм при першому запуску, або вручну іншим адміністратором.

Учень – базова авторизована роль для тих, хто користується платформою для навчання. Учень переглядає каталог, запускає симуляції, проходить РОЕ-цикл, зберігає власні конфігурації як іменовані сценарії,

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

приєднується до класу за восьмисимвольним кодом і виконує завдання. Для типів РОЕ, тест і задача з допуском оцінка виставляється автоматично, для типу «сценарій» – після ручної перевірки вчителем. Учень також завантажує PDF-звіт за результатами здачі, переглядає особисту аналітику у вигляді теплової карти за категоріями фізики та може подати заявку на зміну ролі.

Викладач успадковує весь функціонал учня і додатково керує навчанням: створює віртуальні класи, отримує код запрошення, бачить список учасників, видаляє учнів, скидає код на новий, формує завдання чотирьох типів, переглядає та оцінює здачі, перевіряє статистику класу.

Співавтор – це випадок, коли інший викладач долучається до вже існуючого класу. Якщо до класу приєднується користувач з глобальною роллю Teacher, система автоматично призначає йому локальну роль CoAuthor. Співавтор має майже ті самі права, що й власник: створення завдань, перевірку робіт, видалення учнів, але не може видалити сам клас чи скинути код запрошення. Роль дозволяє кільком викладачам спільно вести один потік.

Адміністратор відповідає за платформу в цілому: керує акаунтами (активація, деактивація, м'яке видалення), розглядає заявки на зміну ролі, переглядає журнал дій і бачить загальну статистику. Це службова, а не навчальна роль. Передбачено захисну перевірку: змінити роль користувачу, який уже є адміністратором, не можна, що запобігає випадковій втраті прав.

Перш ніж переходити до конкретних сценаріїв, варто згрупувати вимоги.

Функціональні вимоги до розроблюваної системи визначають спектр її можливостей та описують ключові сценарії взаємодії користувачів із платформою. Насамперед, система повинна забезпечувати безпечне управління доступом, що включає реєстрацію нових користувачів із вибором початкової ролі та генерацією пари JWT-токенів (access та refresh) для автентифікації. Передбачено авторизацію за допомогою електронної пошти та пароля з перевіркою статусу акаунта, що автоматично блокує доступ (із поверненням помилки 403) у разі його деактивації. Також реалізовано

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

механізм автоматичного оновлення сесії через refresh-токен без потреби повторного входу в систему. Для персоналізації досвіду користувачам надається можливість редагувати дані профілю, зокрема вказувати ім'я, прізвище, заклад вищої освіти, факультет, курс, біографію та завантажувати аватар. З метою дотримання стандартів захисту персональних даних (GDPR) розроблено функції експорту особистої інформації у формат JSON та безповоротного видалення облікового запису [34].

Основним функціональним ядром платформи є робота з інтерактивним освітнім контентом. Користувачі мають змогу переглядати каталог симуляцій із застосуванням фільтрів за категоріями та рівнем складності. Навчальний процес передбачає запуск тривимірних фізичних симуляцій безпосередньо у веб-браузері з можливістю керування їх параметрами за допомогою повзунків. Важливою складовою є проходження тристадійного РОЕ-циклу (Predict-Observe-Explain), який включає автоматичну оцінку точності передбачень студента на основі відносної похибки. Крім того, налаштовані параметри симуляції можна зберігати як іменовані сценарії для їх подальшого використання у навчальному процесі.

Для організації взаємодії між учасниками система підтримує створення віртуальних класів, доступ до яких здійснюється за унікальним восьмисимвольним кодом запрошення із подальшим автоматичним визначенням ролі учасника; при цьому викладач наділений правом скидати та генерувати новий код. Викладачі можуть формувати навчальні завдання чотирьох типів: РОЕ-передбачення, виконання сценаріїв, тестування з варіантами відповідей та вирішення числових задач із заданим допуском. Студентам надається можливість здавати виконані завдання, прикріплюючи скріншоти симуляцій, що супроводжується автоматичним оцінюванням результатів. Водночас для завдань типу «сценарій» передбачено ручне оцінювання викладачем із можливістю додавання текстових коментарів. Для зручного документування результатів реалізовано генерацію звітів про здачу у форматі PDF.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

Платформа здійснює збір детальної статистики виконання завдань, візуалізуючи її за допомогою теплових карт рівня майстерності та графіків розподілу середніх балів.

Нефункціональні вимоги описують, як саме система має це робити.

Час реакції на більшість дій користувача – до 300 мс на ендпоінтах, що не вимагають важких обчислень. Це, зокрема, вхід, отримання списку класів, перегляд завдання. Запуск 3D-симуляції в браузері – не більше 1 секунди на середньостатистичному пристрої з підтримкою WebGL 2.0. Симуляція має йти зі стабільним кадровим темпом 60 кадрів за секунду.

Платформа працює цілодобово; планові прості припустимі лише для оновлення схеми БД або деплою нової версії. Архітектура передбачає, що бекенд можна горизонтально масштабувати – він не зберігає стану між запитами, а вся персистенція винесена у PostgreSQL.

Безпека реалізована через кілька шарів. По-перше, всі захищені ендпоінти позначені атрибутом Authorize, а адмінські вимагають додаткового параметра з роллю Admin. По-друге, працює міدلвар TokenVersionMiddleware, який порівнює поле tokenVersion у JWT-токені з актуальним значенням у базі – якщо вони не співпадають, токен миттєво вважається недійсним. Це дає можливість виганяти користувача з усіх сесій одразу при зміні його ролі або деактивації акаунту. Третій шар – на рівні OnModelCreating у DbContext налаштовано глобальний фільтр IsDeleted = false; видалені користувачі не повертаються у жодному запиті, навіть якщо хтось забув додати умову вручну. І нарешті, у Program.cs додано HTTP-заголовки безпеки: X-Content-Type-Options зі значенням nosniff, X-Frame-Options зі значенням DENY, Referrer-Policy зі значенням no-referrer та Content-Security-Policy, що блокує сторонні ресурси за замовчуванням.

Інтерфейс має бути зрозумілий без додаткової підготовки. Перший вхід нового користувача до моменту запуску першої симуляції не повинен займати більше двох-трьох хвилин. Передбачено світлу й темну теми, причому темна

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

стоїть за замовчуванням; також доступні дві мови – українська та англійська, перемикання між якими відбувається миттєво через перемикач у топбарі.

Тепер – самі варіанти використання. Узагальнена діаграма наведена на рисунку 2.1.

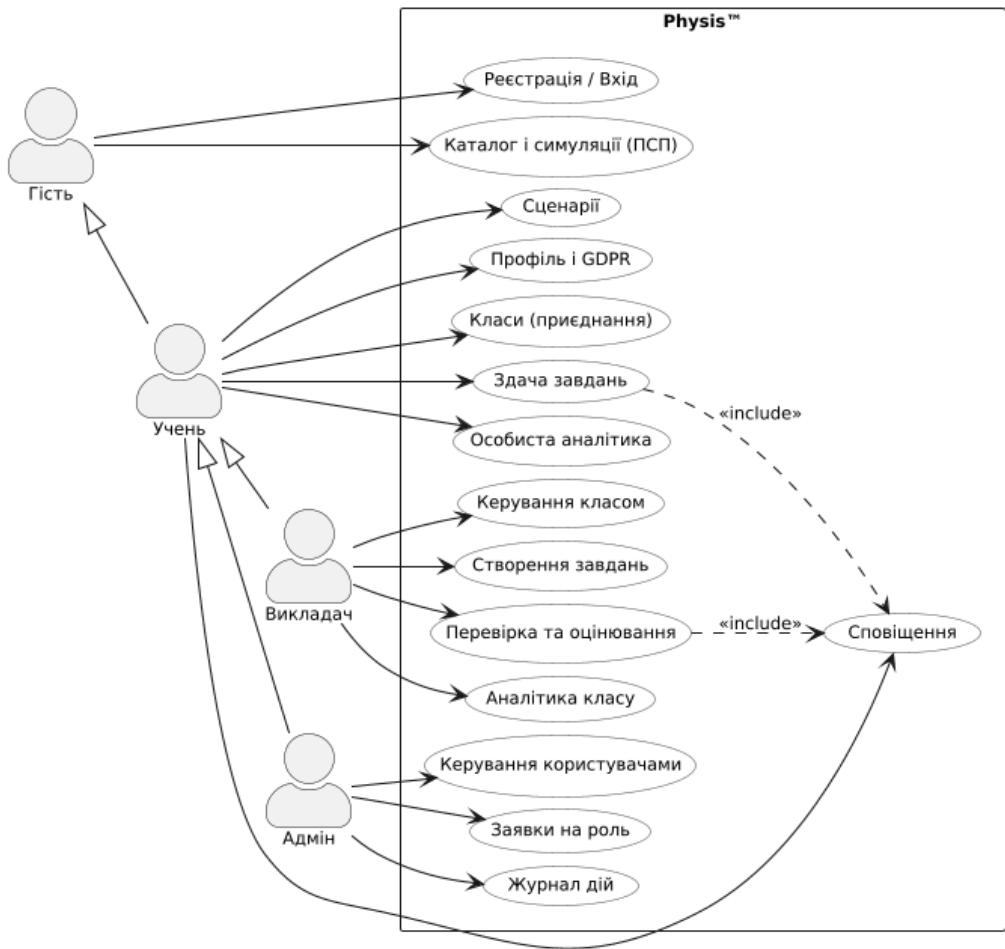


Рис. 2.1. Діаграма варіантів використання

Тепер пройдемося по кожному з ключових прецедентів окремо.

Прецедент «Реєстрація». Основним актором є гість. Користувач вказує email, ім'я облікового запису та пароль, обирає стартову роль – студент або викладач. Сервер створює акаунт через метод UserManager.CreateAsync, присвоює роль через AddToRoleAsync і повертає клієнту пару JWT-токенів. Запис до бази потрапляє з полями IsActive = true та IsDeleted = false за замовчуванням, тож новий акаунт одразу готовий до використання без будь-якого підтвердження з боку адміністратора.

Прецедент «Авторизація». Основні актори – учень, викладач, співавтор та адміністратор. Користувач вводить email і пароль, після чого AuthController.Login перевіряє пароль через CheckPasswordAsync та статус акаунту. У разі успіху видається пара токенів, refresh-токен зберігається у полі RefreshToken з терміном дії 7 днів. Деактивовані акаунти отримують HTTP 403 ще до видачі токенів. Для тривалих сесій передбачено окремий ендпоінт Refresh, який видає нову пару токенів без повторного введення пароля.

Прецедент «Перегляд каталогу симуляцій». Цей сценарій доступний усім, у тому числі неавторизованим гостям. Користувач відкриває сторінку /modules і бачить дванадцять модулів, поділених на шість категорій – механіку, електромагнетизм, хвилі, термодинаміку, гравітацію та квантову фізику. Доступні фільтри за категорією і рівнем складності, а також пошук за назвою чи тегом. Каталог отримується з ендпоінта /api/modules, який повертає вміст статичного класу ModuleCatalog.Entries; це означає, що список доступних симуляцій не зберігається в базі, а зашитий у код і змінюється разом з релізами. Таке рішення обране свідомо: модулі – це не контент, який можна динамічно додавати з адмін-панелі, а складні фронтенд-компоненти з власною логікою рендерингу, тому додавання нового модуля у будь-якому разі вимагає випуску нової версії клієнтського коду.

Прецедент «Проходження РОЕ-циклу». Актори – учень або викладач. Користувач відкриває симуляцію, налаштовує параметри й вводить очікувані значення метрик, запускає симуляцію та спостерігає не менше п'яти секунд, після чого бачить таблицю порівняння передбачених і фактичних значень з підрахунком похибки. У межах завдання типу РОЕ результат автоматично перетворюється на оцінку. Перехід між стадіями не можна прискорити: таймер мінімального часу спостереження блокує кнопку переходу, доки симуляція не відпрацює потрібний інтервал.

Прецедент «Збереження сценарію» доступний учням і викладачам. Користувач задає сценарію назву й натискає «Зберегти», після чого параметри симуляції, останній стан і передбачення серіалізуються у JSON і зберігаються

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

у таблиці SavedScenarios через ScenarioService. Сценарій доступний на сторінці «Мої сценарії» з можливістю повернутися до тієї ж конфігурації одним кліком або поділитися посиланням на читання. Кожен сценарій належить лише своєму автору і недоступний для редагування іншими.

Прецедент «Створення класу та видача коду» доступний лише викладачам. Викладач вводить назву й опис класу; метод ClassroomService.CreateAsync генерує унікальний восьмисимвольний код запрошення з обмеженого алфавіту і одразу створює запис Classroom, додаючи засновника як ClassroomMembership з роллю Teacher. До алфавіту входять літери A–Z без I та O плюс цифри від 2 до 9 – такий вибір зроблено навмисно, щоб не плутати схожі символи на різних шрифтах. Код можна скопіювати у буфер обміну або скинути на новий через дію rotate-code, що корисно у випадку, якщо старий код випадково потрапив не тим людям.

Прецедент «Створення завдання» доступний викладачу та співавтору. У межах класу автор обирає симуляційний модуль, задає назву, опис, опціональний дедлайн і тип завдання – РОЕ, сценарій, тест або задача з допуском. Залежно від типу задаються специфічні поля: для РОЕ і сценарію – очікувані значення метрик, для тесту – питання з варіантами та номером правильної відповіді, для задачі – поля з міткою, одиницею вимірювання, правильним значенням і допуском. Дані серіалізуються у відповідні JSON-поля запису Assignment. Право на створення має лише учасник класу з роллю Teacher або CoAuthor.

Прецедент «Здача та оцінювання роботи» розпадається на дві послідовні дії. Учень виконує завдання та надсилає здачу; для типів РОЕ, тест і задача GradingService рахує оцінку автоматично. Для типу «сценарій» оцінювання очікує ручної перевірки викладачем, який після виставлення балу публікує результат. Одразу після здачі система надсилає сповіщення викладачам та співавторам класу, а після перевірки – учневі. Повторно здати те саме завдання не можна.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

Окрім цих восьми, на діаграмі представлені й інші варіанти – приєднання до класу за кодом, генерація PDF-звіту, перегляд аналітики, адмінські дії на зразок підтвердження заявки на роль чи деактивації користувача, редагування профілю. Усі вони мають подібну структуру і не вимагають окремих таблиць, тому стисло згадаються у відповідних параграфах далі.

Аналіз функціональних вимог дозволив виокремити дев'ять основних доменних сутностей, реалізованих як C#-класи у проекті Physis.Api. Їхні поля та зв'язки показано на діаграмі класів (рис. 2.2), а нижче кожену сутність коротко описано.

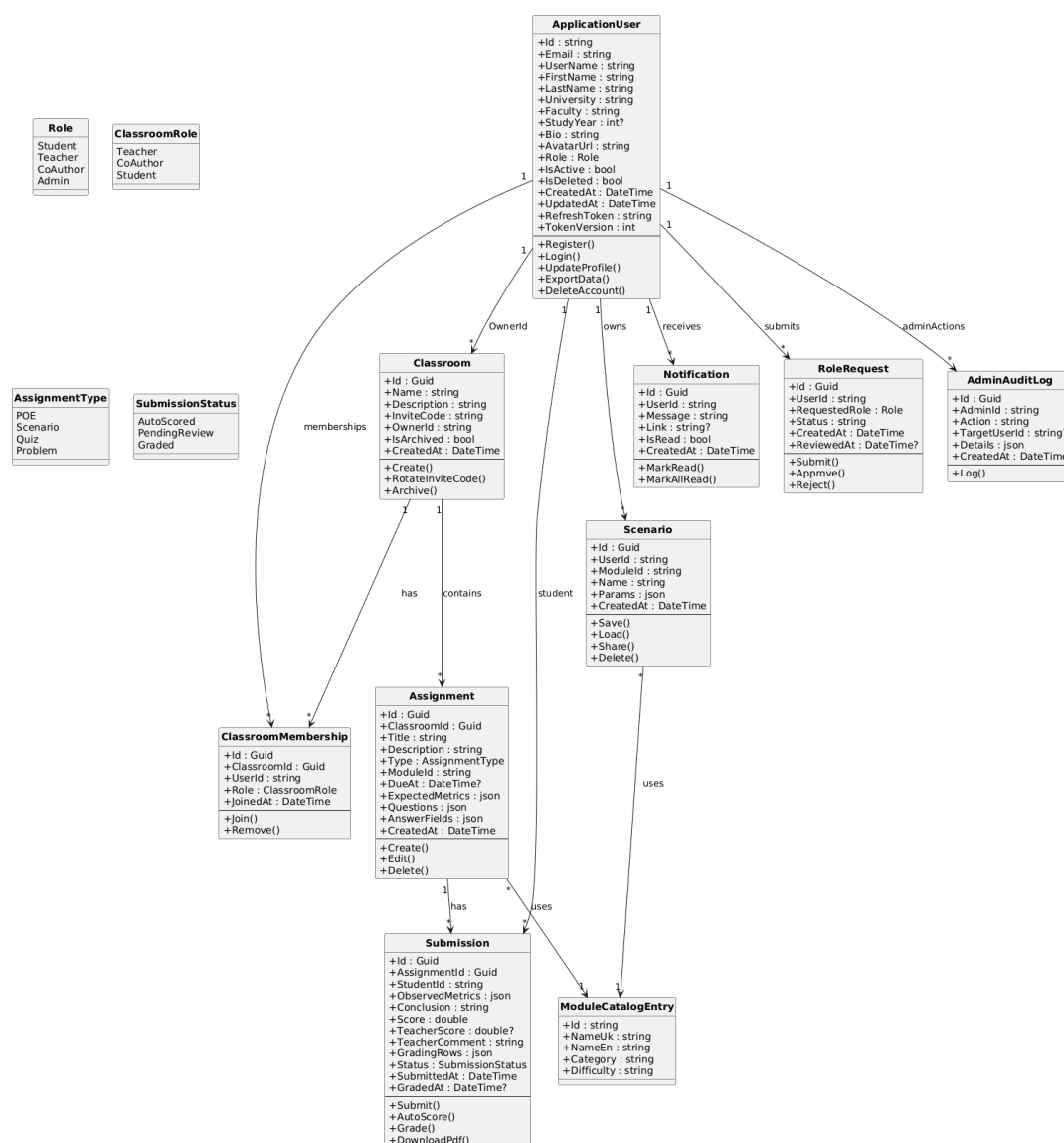


Рис. 2.2 Діаграма класів

ApplicationUser – користувач системи, спадкоємець стандартного класу IdentityUser зі стека ASP.NET Core Identity. До базових полів додано доменні (ім'я, прізвище, ВНЗ, факультет, курс, біографія, аватар) і службові – IsActive, IsDeleted, TokenVersion та refresh-токен з датою його закінчення. Поле TokenVersion – лічильник, який збільшується при зміні ролі чи деактивації акаунту і використовується для миттєвої інвалідації сесій.

Classroom – віртуальний клас, який створює викладач. Власник зберігається окремо від членства, що дозволяє передавати клас іншому викладачу без його втрати. Код запрошення InviteCode має унікальний індекс.

ClassroomMembership – асоціативна сутність, що пов'язує користувача з класом і фіксує його локальну роль через перерахування ClassroomRole. Унікальний індекс на парі (ClassroomId, UserId) не дозволяє додати того самого користувача до класу двічі. Локальна роль не зобов'язана збігатися з глобальною: викладач, який приєднується до чужого класу, автоматично стає співавтором, а не учнем.

Assignment – навчальне завдання у складі класу. Окрім базових полів містить тип AssignmentType і три текстові JSON-поля – ExpectedMetrics, Questions та AnswerFieldsJson, які заповнюються залежно від типу завдання. Завдяки такому підходу один запис описує всі чотири типи завдань без розведення їх по окремих таблицях.

Submission – задача завдання учнем. Містить фактичні метрики, рядки порівняння з похибками, автоматичну оцінку Score, опціональну ручну оцінку TeacherScore, статус і дату надсилання. Унікальний індекс на парі (AssignmentId, StudentId) забороняє повторну задачу того самого завдання.

SubmissionArtifact – допоміжна сутність для зберігання бінарних даних задачі (скріншот, PDF-звіт) у форматі bytea. Зв'язок із Submission один-до-одного з каскадним видаленням.

SavedScenario – збережена іменована конфігурація симуляції з параметрами, станом і передбаченнями у JSON. Сутність незалежна від класів і завдань – це персональна закладка користувача.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

Notification – внутрішнє сповіщення, що з'являється у дзвоник топбару. Складений індекс на полях (UserId, IsRead) пришвидшує вибірку непрочитаних сповіщень користувача.

AdminAuditLog – журнал адміністративних дій з типом дії, ідентифікатором цілі та JSON-контекстом. Зовнішній ключ на адміністратора має режим Restrict, тож журнал не зникає навіть при видаленні самого адмін-акаунту.

RoleChangeRequest – заявка на зміну ролі зі статусом Pending, Approved або Rejected. Активна заявка на користувача може бути лише одна; повторна замінює попередню.

Поряд із сутностями важливу роль відіграють сервіси. Це класи, які містять бізнес-логіку, що виходить за межі одного запиту. Архітектурно вони відповідають патерну Service Layer і реєструються у DI-контейнері через AddScoped у Program.cs. Основні сервіси та їхні ключові методи зведені у таблиці 2.1.

Таблиця 2.1

Доменні сервіси та їхні функції

Сервіс	Призначення	Ключові методи
TokenService	Генерація та валідація JWT	GenerateAccessTokenAsync, GenerateRefreshToken
AssignmentService	Завдання, здачі та виставлення оцінки	CreateAsync, SubmitAsync, GradeAsync, GetDetailAsync
GradingService	Автоматичне оцінювання	Grade (POE/сценарій), GradeProblem (задача з допуском)
AnalyticsService	Підрахунок статистики	GetPersonalAsync, GetOverviewAsync, GetStudentTimelineAsync
NotificationService	In-app сповіщення	CreateAsync, CreateManyAsync, MarkAsReadAsync
ScenarioService	Збережені сценарії	CreateAsync, GetMineAsync, DeleteAsync

ProfileService	Профіль і завантаження аватара	UpdateAsync, UploadAvatarAsync
IService / SmtplibService	Email через SMTP	SendAsync
LabReportService	Генерація PDF-звітів	GenerateAsync (використовує QuestPDF)
ClassroomService	Робота з класами та учасниками	CreateAsync, JoinAsync, RotateCodeAsync, RemoveMemberAsync

GradingService реалізовано як статичний клас без залежностей, бо оцінювання є чистою функцією: на вхід надходять очікувані й фактичні значення, на вихід – оцінка та рядки порівняння. Виносити це в окремий DI-сервіс було б надмірно, а статичні методи зручні ще й для тестування, бо викликаються напряму без моків.

Контролери тримають мінімум логіки: типовий метод дістає ідентифікатор користувача з токена, делегує виконання сервісу і повертає результат. Усе складніше – перевірки прав, серіалізація, відправка сповіщень – винесене у сервісний шар, завдяки чому контролери лишаються прозорими, а логіка не дублюється між схожими ендпоінтами. Форму даних в обміні між клієнтом і сервером описують DTO – невеликі імутабельні рекорди, які приховують від клієнта внутрішні поля на зразок PasswordHash чи TokenVersion. На фронтенді їм відповідають синхронні за структурою TypeScript-інтерфейси, що дає повну підказку типів і одразу сигналізує про зміну структури відповіді на бекенді.

Отже, об'єктна модель платформи побудована на чіткому поділі відповідальності: сутності описують стан, сервіси – поведінку, контролери – транспорт, DTO – формат обміну. Така структура спрощує і тестування, і розвиток системи, бо додавання нової функції зазвичай зводиться до однотипних змін у кожному з шарів.

2.2. Розробка бази даних системи

Структура бази даних – фундамент, на якому тримається решта системи, бо змінювати схему у працюючому застосунку складніше, ніж переписати окремий контролер. У цьому параграфі описано вибір PostgreSQL як сховища, організацію доступу через Entity Framework Core, перелік таблиць, зв'язки між ними та накладені обмеження й індекси.

Сховищем даних виступає PostgreSQL версії 16 [5, 24]. Цей вибір зроблено навмисно: серед безкоштовних реляційних СУБД Postgres має найкращу комбінацію надійності, продуктивності та функціональних можливостей. Він підтримує ACID-транзакції з повноцінною ізоляцією рівнів, має багатий набір вбудованих типів даних, дозволяє створювати власні індекси та оператори, працює стабільно під значним навантаженням. Окремо варто згадати підтримку JSON-типів – це виявилось критично важливим для проекту, у якому деякі поля принципово не вкладаються у фіксовану схему.

Доступ до бази здійснюється через Entity Framework Core 9 з Npgsql-провайдером [6, 21, 25]. Це об'єктно-реляційний маппер, який дозволяє описувати схему C#-класами і працювати з даними через LINQ замість сирого SQL. Застосовано підхід Code First: схема описується кодом, а SQL-скрипти генеруються автоматично у вигляді міграцій, які зберігаються у проекті разом з рештою коду. Завдяки цьому історія змін схеми лежить у Git, розгортання на новому середовищі зводиться до послідовного застосування міграцій, а відкат керується через CLI без ручного редагування бази. На момент написання роботи накопичено 11 міграцій, що відображає ітеративну розробку – схема еволюціонувала разом з функціональністю, а не з'явилася одразу повністю.

Автоматичне застосування міграцій налаштоване у файлі Program.cs через виклик `db.Database.Migrate()` при кожному запуску сервера. Завдяки цьому розгортання нової версії не вимагає окремого кроку оновлення бази – щойно сервер стартує, він приводить схему до актуального стану.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

Загалом база даних системи складається з десяти доменних таблиць плюс набору системних таблиць ASP.NET Core Identity. Identity створює сім допоміжних таблиць –AspNetUsers, AspNetRoles, AspNetUserRoles, AspNetUserClaims, AspNetUserLogins, AspNetUserTokens та AspNetRoleClaims – які відповідають за зберігання паролів, хешів, ролей, claims і токенів. Усі вони генеруються автоматично і працюють «з коробки», тому окремо описувати їх немає сенсу: достатньо знати, що користувачський акаунт у системі – це насправді запис у таблиці AspNetUsers, розширений власним класом ApplicationUser із додатковими профільними полями. Далі у параграфі описано саме доменну частину схеми. Зведений перелік таблиць з призначенням наведено нижче.

Першою йде таблиця Users – точніше, її логічне продовження AspNetUsers + ApplicationUser, де зберігається інформація про користувачів платформи. Друга – SavedScenarios – містить збережені користувачами конфігурації симуляцій. Третя – Classrooms – описує віртуальні класи, створені викладачами. Четверта – ClassroomMemberships – асоціативна таблиця, що пов'язує користувачів з класами. П'ята – Assignments – навчальні завдання у складі класів. Шоста – Submissions – здачі завдань учнями. Сьома – SubmissionArtifacts – бінарні артефакти здач, як-от PDF-звіти чи скріншоти. Восьма – RoleChangeRequests – заявки на зміну ролі. Дев'ята – AdminAuditLogs – журнал адміністративних дій. Десята – Notifications – внутрішні сповіщення системи. Опис полів кожної з цих таблиць наведено у таблицях 2.2–2.11.

Таблиця 2.2

Опис таблиці Users

Поле	Тип	Призначення
Id	string	Ідентифікатор користувача, первинний ключ
Email	string	Електронна адреса користувача
UserName	string	Ім'я облікового запису
FirstName	string	Ім'я

LastName	string	Прізвище
University	string	Заклад освіти
Faculty	string	Факультет
StudyYear	int	Курс навчання
Bio	string	Коротка біографія
AvatarUrl	string	Посилання на аватар
IsActive	bool	Активність акаунту
IsDeleted	bool	Прапорець м'якого видалення
TokenVersion	int	Лічильник для інвалідації JWT-токенів
RefreshToken	string	Поточний refresh-токен
RefreshTokenExpiry	DateTime	Термін дії refresh-токена
CreatedAt	DateTime	Дата створення акаунту
UpdatedAt	DateTime	Дата останнього оновлення

Окремо для таблиці Users налаштовано глобальний фільтр у DbContext: будь-який запит автоматично доповнюється умовою IsDeleted = false. Це означає, що видалений користувач фізично зберігається у базі для цілей аудиту та можливого відновлення, але жоден звичайний запит його не побачить. Така схема м'якого видалення безпечніша за фізичне DELETE, бо запобігає каскадній втраті пов'язаних даних – здач, сценаріїв, журналів.

Таблиця 2.3

Опис таблиці SavedScenarios

Поле	Тип	Призначення
Id	Guid	Ідентифікатор сценарію
UserId	string	Зовнішній ключ на власника сценарію
ModuleId	string	Ідентифікатор модуля симуляції з каталогу
Name	string	Назва сценарію, задана користувачем
ParamsJson	string	JSON з параметрами симуляції
StateSnapshotJson	string	JSON з миттєвим знімком стану сцени
PredictionsJson	string	JSON з передбаченнями користувача
CreatedAt	DateTime	Дата створення

Таблиця 2.4

Опис таблиці Classrooms

Поле	Тип	Призначення
Id	Guid	Ідентифікатор класу
Name	string	Назва класу
Description	string	Опис класу
OwnerId	string	Зовнішній ключ на викладача-засновника
InviteCode	string	Унікальний восьмицифровий код запрошення
IsArchived	bool	Прапорець архівованого класу
CreatedAt	DateTime	Дата створення

На полі InviteCode стоїть унікальний індекс, який гарантує, що два класи не можуть мати однакові коди запрошення. Зв'язок з власником через OwnerId налаштований з режимом OnDelete(Restrict): класи не видаляються автоматично при видаленні викладача, бо у разі soft delete акаунту клас має залишитися доступним для інших викладачів-співавторів.

Таблиця 2.5

Опис таблиці ClassroomMemberships

Поле	Тип	Призначення
Id	Guid	Ідентифікатор запису
ClassroomId	Guid	Зовнішній ключ на клас
UserId	string	Зовнішній ключ на користувача
Role	enum	Локальна роль у класі: Teacher, Student або CoAuthor
JoinedAt	DateTime	Дата приєднання до класу

На парі полів ClassroomId та UserId стоїть складений унікальний індекс. Це гарантує, що один користувач не може бути доданий до того самого класу двічі – повторна спроба приєднатися за тим же кодом просто оновить існуючий запис, а не створить дублікат. Перевірка унікальності виконується на рівні бази даних, тому навіть у разі одночасних паралельних запитів від одного користувача система зберігає узгоджений стан складу учасників класу.

Таблиця 2.6

Опис таблиці Assignments

Поле	Тип	Призначення
Id	Guid	Ідентифікатор завдання
ClassroomId	Guid	Зовнішній ключ на клас
CreatedById	string	Зовнішній ключ на автора завдання
ModuleId	string	Ідентифікатор симуляції з каталогу
Title	string	Назва завдання
Description	string	Опис умови
AssignmentType	enum	Тип завдання: Poe, Scenario, Quiz або Problem
ExpectedMetrics	string	JSON з очікуваними значеннями метрик
Questions	string	JSON з питаннями тесту
AnswerFieldsJson	string	JSON з полями числової задачі
DueAt	DateTime	Опціональний дедлайн
CreatedAt	DateTime	Дата створення

Таблиця 2.7

Опис таблиці Submissions

Поле	Тип	Призначення
Id	Guid	Ідентифікатор здачі
AssignmentId	Guid	Зовнішній ключ на завдання
StudentId	string	Зовнішній ключ на учня
ObservedMetrics	string	JSON з фактичними значеннями метрик
GradingRows	string	JSON з рядками порівняння і похибками
QuizAnswers	string	JSON з відповідями на тест
ProblemAnswers	string	JSON з відповідями на числову задачу
ConclusionText	string	Текстове пояснення або висновок учня
Score	double	Автоматична оцінка від 0 до 1
TeacherScore	double	Опціональна ручна оцінка викладача
TeacherComment	string	Коментар викладача при ручній перевірці

Status	enum	Статус здачі: Submitted, PendingReview, ReviewedPublished
SubmittedAt	DateTime	Дата надсилання

На парі полів AssignmentId та StudentId стоїть складений унікальний індекс, що унеможлиблює повторну здачу завдання тим же учнем. Поле TeacherComment додано у дев'ятій міграції разом із ручною перевіркою завдань типу Scenario.

Таблиця 2.8

Опис таблиці SubmissionArtifacts

Поле	Тип	Призначення
Id	Guid	Ідентифікатор артефакту
SubmissionId	Guid	Зовнішній ключ на здачу, унікальний
Kind	string	Тип артефакту, наприклад pdf
ContentType	string	MIME-тип вмісту
Data	bytea	Бінарний вміст файлу

Таблиця 2.9

Опис таблиці RoleChangeRequests

Поле	Тип	Призначення
Id	Guid	Ідентифікатор заявки
UserId	string	Зовнішній ключ на користувача-заявника
RequestedRole	string	Запитувана роль
Status	string	Статус заявки: Pending, Approved, Rejected
CreatedAt	DateTime	Дата подання
ResolvedAt	DateTime	Дата розгляду
ResolvedByAdminId	string	Адміністратор, який розглянув заявку

Таблиця 2.10

Опис таблиці AdminAuditLogs

Поле	Тип	Призначення
Id	Guid	Ідентифікатор запису
AdminId	string	Адміністратор, який виконав дію

Action	string	Тип дії: Activate, Deactivate, Delete тощо
TargetUserId	string	Користувач, над яким виконано дію
Details	string	JSON з додатковим контекстом
CreatedAt	DateTime	Дата дії

Зовнішній ключ на адміністратора має режим Restrict: видалити обліковий запис адміністратора, поки за ним числяться записи у журналі, неможливо. Це навмисний захист цілісності журналу – записи аудиту не повинні зникати разом з акаунтом того, хто їх створив.

Таблиця 2.11

Опис таблиці Notifications

Поле	Тип	Призначення
Id	Guid	Ідентифікатор сповіщення
UserId	string	Зовнішній ключ на адресата
Message	string	Текст сповіщення
Link	string	Опціональне посилання для переходу
IsRead	bool	Прапорець прочитаного сповіщення
CreatedAt	DateTime	Дата створення

На парі полів UserId та IsRead стоїть складений індекс. Він пришвидшує найчастіший запит до цієї таблиці – отримання списку непрочитаних сповіщень користувача, який запускається при кожному відкритті дзвоника у топбарі.

Окремо варто зупинитись на JSON-полях. У схемі їх кілька, причому з різним призначенням. У SavedScenarios зберігаються параметри симуляції, знімок стану та передбачення; у Assignments – очікувані метрики, питання тесту та поля числової задачі; у Submissions – фактичні метрики, рядки порівняння, відповіді тесту та задачі; у AdminAuditLogs – довільний контекст дії. Усі ці поля принципово не мають фіксованої структури: різні модулі симуляцій оперують різними наборами параметрів, тести можуть мати різну кількість варіантів, задачі – різну кількість полів. Якщо описувати все це через додаткові таблиці зі зв'язками, схема стане громіздкою і важкопідтримуваною.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

Натомість обрано простіший підхід: вміст таких полів серіалізується у JSON-рядок і зберігається як text. Postgres має повноцінний бінарний тип jsonb, але у поточній реалізації використовується text, бо ці поля не запитуються через WHERE-умови і не потребують індексації за вмістом. Якщо у майбутньому виникне потреба фільтрувати чи шукати всередині JSON – міграція з text на jsonb потребує лише однієї невеликої зміни у налаштуваннях моделі. Серіалізація та десеріалізація відбуваються через System.Text.Json і реалізовані всередині відповідних сервісів.

Перерахування – ClassroomRole, AssignmentType та SubmissionStatus – зберігаються як числові коди типу int. Це компактніше за рядки і захищає від помилок у написанні, бо C# не дозволить присвоїти полю значення, якого немає у перерахуванні.

Зв'язки між таблицями мають кілька різних режимів каскадування, кожен з яких обраний свідомо. При видаленні користувача каскадно видаляються його сценарії, сповіщення, заявки на роль, членства у класах. Класи, створені цим користувачем, не видаляються (режим Restrict) – вони залишаються у системі і можуть бути передані іншому викладачу. Те ж саме з журналом адміністративних дій: записи зберігаються навіть якщо акаунт самого адміністратора буде деактивовано. При видаленні класу каскадно видаляються членства, завдання та всі здачі, прив'язані до цих завдань. При видаленні здачі автоматично видаляється і прив'язаний до неї артефакт.

Узагальнена ER-діаграма бази даних з усіма таблицями, типами зв'язків та ключовими полями наведена на рис. 2.3. На діаграмі чітко прослідковується центральне положення таблиці користувачів, від якої відходить більшість зовнішніх ключів до інших сутностей, а також групування таблиць за функціональними блоками – навчальний процес, аналітика та адміністрування. Стрілки зв'язків відображають кардинальність і режим каскадування, що дозволяє з одного погляду оцінити цілісність схеми та зрозуміти, які саме сутності будуть зачеплені при видаленні або зміні записів.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

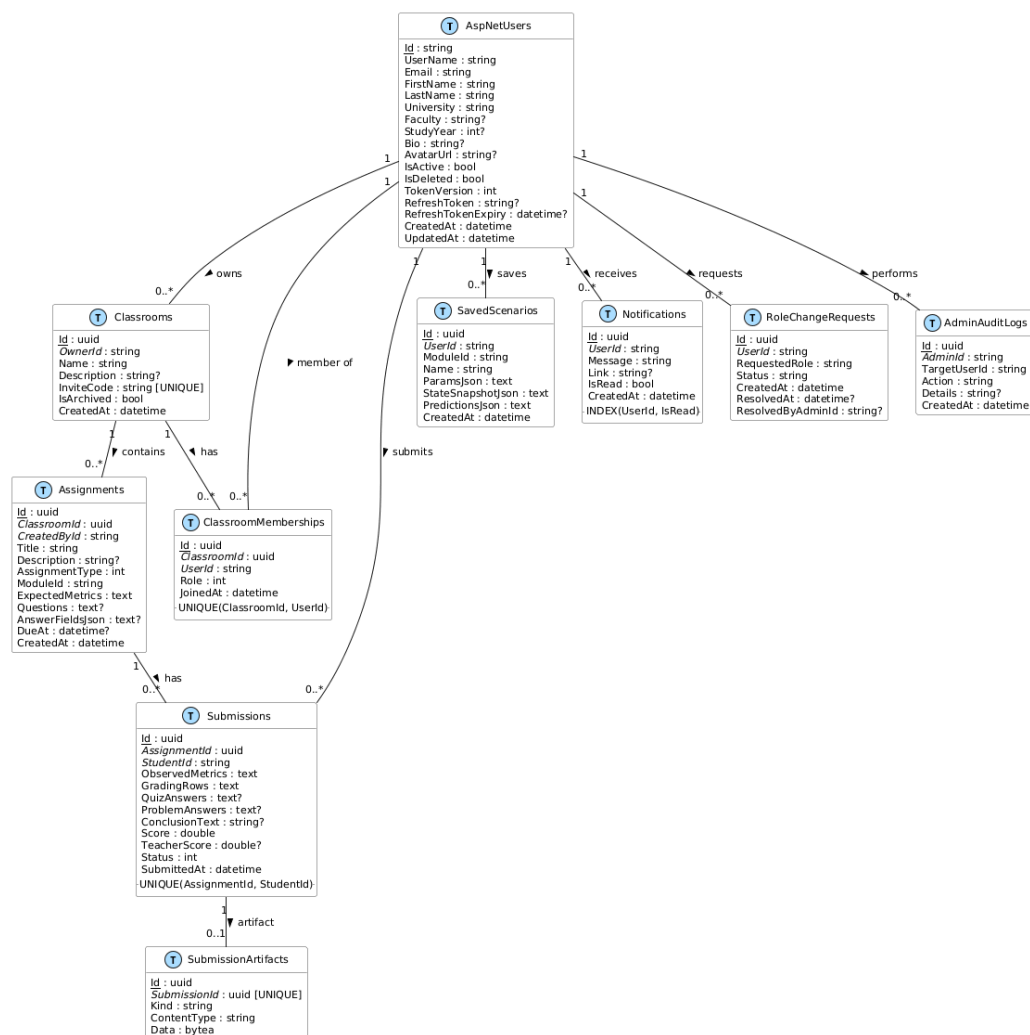


Рис. 2.3 ER-діаграма бази даних системи

Окремо від бази, але в межах загальної інформаційної моделі, існує статичний каталог симуляційних модулів – C#-клас ModuleCatalog з переліком дванадцяти модулів, кожен з ідентифікатором, назвами українською й англійською, категорією та рівнем складності. Каталог не зберігається у базі навмисно: модулі є складними фронтенд-компонентами, а не контентом для адмін-панелі, тож додавання нового модуля у будь-якому разі вимагає нової версії клієнтського коду. У моделях завдань і сценаріїв на модуль посилається рядкове поле ModuleId без явного зовнішнього ключа.

Загалом схема вийшла компактна і пристосована саме до потреб платформи. У ній немає універсальних рішень «на всі випадки життя» – кожна таблиця, кожен зв'язок і кожен індекс мають конкретне обґрунтування у реальній логіці роботи системи. Реляційна частина відповідає за основну структуру, JSON-поля – за гнучкий контент, м'яке видалення – за збереження

історичних даних. Одинадцять міграцій у репозиторії дозволяють у будь-який момент відкатитись на потрібну точку у розвитку схеми і простежити, як саме змінювалась структура від першого прототипу до фінальної версії.

2.3. Проектування та реалізація алгоритмів роботи системи

За простим інтерфейсом платформи стоїть набір алгоритмів, спроектованих з урахуванням різних ролей користувачів, типів навчальних завдань та особливостей роботи з тривимірними симуляціями. У цьому параграфі описано загальну логіку роботи системи, окремо розглянуто цикл РОЕ, а також наведено формули та пояснення до ключових обчислювальних алгоритмів – оцінювання здач, авторизації та безпеки сесій.

Загальний алгоритм роботи з платформою у вигляді UML-діаграми активності винесено у додаток А; нижче коротко описано його основні гілки за ролями користувачів.

Точкою входу для будь-якого користувача є головна сторінка платформи, з якої можна перейти або до сторінки реєстрації, або до сторінки входу, або одразу до публічного каталогу симуляцій. Гість, який не має облікового запису, може ознайомитись з каталогом, відкрити будь-яку симуляцію і пройти РОЕ-цикл, але без можливості зберегти результати, приєднатися до класу або виконати завдання. Подальший шлях користувача гілкується залежно від результату авторизації.

Подальший шлях користувача гілкується залежно від ролі. Адміністратор після входу потрапляє до адмін-панелі, де переглядає список усіх користувачів і керує їхніми акаунтами (активація, деактивація, м'яке видалення), розглядає заявки на зміну ролі, переглядає журнал адміністративних дій з фільтрацією за датою та виконавцем і має доступ до загальної статистики платформи. Викладач переходить на сторінку «Мої класи» з переліком власних класів і тих, де він є співавтором; він створює нові класи з кодом запрошення, формує завдання чотирьох типів з опціональним

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

дедлайном, переглядає здачі, виставляє ручні оцінки для завдань типу «сценарій», публікує результати та за потреби скидає код запрошення на новий. Учень на тій самій сторінці бачить класи, до яких приєднався, вводить код для приєднання до нового, виконує видані завдання через відповідні симуляції, надсилає здачі, отримує сповіщення з результатом і PDF-звіт та переглядає особисту аналітику у вигляді теплової карти майстерності. Незалежно від ролі, авторизований користувач має доступ до профілю, де редагує особисті дані, завантажує аватар, експортує власні дані у JSON або видаляє акаунт відповідно до GDPR.

Незалежно від ролі, авторизований користувач має доступ до сторінки профілю, де може редагувати особисті дані, завантажити аватар, експортувати власні дані у форматі JSON або повністю видалити свій акаунт відповідно до GDPR.

Центральним педагогічним елементом платформи є цикл «Передбачай – Спостерігай – Пояснюй», що проходить кожен учень при виконанні симуляційного завдання. Деталізовану діаграму активності цього циклу наведено у додатку А, а далі описано логіку трьох послідовних стадій з опорою на її реалізацію у коді.

РОЕ-цикл реалізований у компоненті `poe-workflow.component.ts` і має три послідовні стадії, що описуються одним TypeScript-типом – `PoePhase` який набуває значень `predict`, `observe` або `explain`. Перехід між стадіями керується умовами, які не дозволяють просто «прокликати» цикл без виконання.

На стадії «Передбачити» учень бачить опис явища і набір очікуваних метрик з порожніми полями, куди вводить свої прогнози через слайдери або числові поля. Кнопка переходу до наступної стадії тут не має жодних умов – задача стадії не в тому, щоб одразу вгадати правильну відповідь, а в тому, щоб сформулювати власне очікування для подальшого порівняння з реальністю.

На стадії «Спостерігати» запускається тривимірна симуляція з параметрами завдання, а паралельно стартує таймер реального часу спостереження. Стадія керується константою `MIN_OBSERVE_SECONDS = 5`:

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

до завершення п'ятисекундного інтервалу кнопка переходу заблокована, а під симуляцією заповнюється прогрес-бар. Без цього обмеження учень міг би миттєво натиснути «далі», не подивившись на процес, і РОЕ-цикл втратив би сенс. П'ять секунд – компромісне значення, достатнє, щоб побачити характер руху, і не настільки довге, щоб дратувати при серії завдань.

На стадії «Пояснити» компонент через виклик `module.getMetrics()` отримує фактичні значення тих самих метрик і будує таблицю порівняння передбачених і фактичних значень з підрахунком абсолютної та відносної похибки. Таблиця формується інлайн у самому компоненті; за оцінювання при задачі завдання відповідає вже серверна логіка. Учень бачить, наскільки його прогноз збігся з реальністю, і вводить текстове пояснення причин розбіжності. На цьому РОЕ-цикл завершується, а у разі виконання завдання формується об'єкт задачі, який відправляється на сервер.

Алгоритми оцінювання визначають, як платформа переводить результати завдань у числові оцінки. Уся логіка зосереджена у класі `GradingService` і викликається з `AssignmentService.SubmitAsync` при кожній задачі. Перед оцінюванням сервер проводить чотири перевірки: наявність завдання, членство учня у класі, відповідність ролі (здавати може лише учень) і відсутність дублікату задачі. Лише після цього алгоритм гілкується за типом завдання.

Для завдання типу РОЕ та для типу «сценарій» оцінювання виконує метод `Grade`. На вхід він приймає два словники – очікувані значення `expected` (з поля `ExpectedMatrix` завдання) та фактичні `observed` (з результату симуляції). Для кожного ключа з очікуваних метрик алгоритм виконує наступне. Якщо у фактичних значеннях відповідного ключа немає – метрика пропускається без штрафу. Інакше обчислюється абсолютна похибка як модуль різниці фактичного і очікуваного значень:

$$\text{abs_error} = |\text{actual} - \text{expected}|$$

Далі обчислюється відносна похибка з урахуванням випадку, коли очікуване значення близьке до нуля. Якщо `|expected|` менше за порогову

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

константу 10^{-12} , похибка вважається нульовою за умови, що фактичне значення теж близьке до нуля, або одиничною, якщо фактичне значення ненульове. У звичайному випадку формула стандартна:

$$\text{rel_error} = \text{abs_error} / |\text{expected}|$$

Бал за окрему метрику обчислюється як одиниця мінус відносна похибка, з нижньою межею нуль:

$$\text{metric_score} = \max(0, 1 - \text{rel_error})$$

При точному попаданні метрика дає 1.0, при відхиленні на 50% – 0.5, при відхиленні на 100% і більше – нуль. Кліпування знизу нулем гарантує, що жодна метрика не зробить підсумковий бал від'ємним. Підсумкова оцінка за завдання – середнє арифметичне балів за всіма зіставленими метриками:

$$\text{score} = \sum \text{metric_score}(i) / N$$

де N – кількість метрик, у яких знайшлося і очікуване, і фактичне значення.

Для завдання типу «тест» алгоритм простіший. Список питань зберігається у полі Questions у форматі JSON, відповіді учня – у полі QuizAnswers як масив індексів обраних варіантів. Для кожного питання з індексом i алгоритм перевіряє, чи дав учень відповідь – тобто чи виходить індекс i за межі масиву відповідей – і чи дорівнює його обраний варіант полю CorrectIndex цього питання. Якщо учень пропустив питання або обрав неправильний варіант, бал за нього дорівнює нулю; часткові бали за «частково правильні» відповіді не передбачено. Підсумкова оцінка обчислюється як частка правильних відповідей від загальної кількості питань:

$$\text{score} = \text{correct} / \text{total}$$

Для завдання типу «числова задача» алгоритм реалізований у методі GradeProblem. Поля задачі описані у JSON-полі AnswerFieldsJson і містять для кожного поля очікуване значення CorrectValue та допуск Tolerance. Відповіді учня надходять у полі ProblemAnswers. Для кожного поля алгоритм знаходить відповідь учня за назвою, обчислює абсолютну різницю і зараховує поле, якщо ця різниця не перевищує допуск:

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

$\text{field_correct} = (|\text{student_value} - \text{correct_value}| \leq \text{tolerance})$

Tolerance є абсолютною величиною, а не відсотковим відхиленням: при допуску 0.1 м/с і правильному значенні 1.00 м/с відповідь 1.05 м/с зараховується, а 1.15 м/с – ні. Підсумкова оцінка – частка зарахованих полів від загальної кількості, тож незаповнені поля знижують результат.

Після обчислення оцінки створюється об'єкт Submission зі статусом PendingReview – і цей статус виставляється для всіх чотирьох типів завдань без розгалуження. Навіть якщо оцінка обчислилась автоматично, остаточне публікування результату викладач робить вручну: переглядає роботу та пояснення учня, за потреби виставляє власну оцінку через TeacherScore і переводить статус у ReviewedPublished. Дедлайн DueAt при здачі не перевіряється – пізні здачі приймаються, але дата надсилання зберігається у SubmittedAt, тож викладач може врахувати запізнення вручну.

Перевірка роботи викладачем реалізована у методі GradeAsync. Алгоритм починається з перевірки прав: викладач повинен мати у відповідному класі локальну роль Teacher або CoAuthor. У разі відповідності алгоритм встановлює нову оцінку, обмежену діапазоном [0, 1] через виклик Math.Clamp, опціональний коментар (який перезаписує поле ConclusionText) і переводить статус здачі у ReviewedPublished. Одночасно надсилається сповіщення учневі – і всередині платформи через дзвоник у топбарі, і на електронну пошту через SMTP-сервер.

Окрему частину алгоритмічного забезпечення складають механізми авторизації та безпеки сесій. Авторизація реалізована у методі Login контролера AuthController. Спершу за email шукається користувач через UserManager.FindByEmailAsync, далі через CheckPasswordAsync перевіряється пароль; при будь-якій помилці повертається HTTP 401. Якщо акаунт неактивний (IsActive = false), повертається HTTP 403. У разі успіху TokenService генерує access-токен у вигляді JWT з підписаним набором claims – ідентифікатором користувача, email, іменем, унікальним ідентифікатором токена, поточним tokenVersion та ролями [15]. Окремо генерується refresh-

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

токен як випадкова послідовність з 64 криптографічно безпечних байт у Base64 і записується разом з датою закінчення (поточний час плюс сім днів). Терміни дії – година для access-токена і сім днів для refresh-токена – налаштовуються у appsettings.json без перекомпіляції.

Оновлення сесії виконує ендпоінт Refresh. У базі даних шукається користувач з відповідним значенням поля RefreshToken. Якщо такого користувача немає або поле RefreshTokenExpiry менше за поточний час, повертається HTTP 401, і клієнт зобов'язаний виконати повторний логін. У разі успіху генерується нова пара токенів, причому старий refresh-токен перезаписується новим – це називається ротацією refresh-токена і запобігає сценарію, коли той самий токен може бути використаний повторно у разі його перехоплення.

Третій компонент безпеки – міدلвар TokenVersionMiddleware, що працює на кожному автентифікованому запиті. Він зареєстрований у конвеєрі ASP.NET Core між UseAuthentication() і UseAuthorization(), тож на момент його роботи користувач уже автентифікований за токеном. Неавтентифіковані запити міدلвар пропускає одразу. Для решти з claim sub витягується ідентифікатор користувача і робиться запит до бази: якщо користувача не знайдено або його прапорець IsDeleted дорівнює true – повертається HTTP 401, якщо IsActive дорівнює false – HTTP 403, а якщо tokenVersion у токені не збігається з актуальним значенням у базі – HTTP 401 з повідомленням про недійсність токена. Остання ситуація виникає, коли адміністратор змінив роль або деактивував користувача: TokenVersion у базі збільшується, і всі видані раніше токени стають недійсними. Це і є механізм миттєвої інвалідації сесій без ведення чорного списку токенів.

Звернення до бази на кожному запиті є додатковим навантаженням, але за поточного обсягу трафіку це прийнятна ціна за миттєвість реакції на адміністративні дії. У навантаженій продакшен-системі такий міدلвар зазвичай доповнюють кешуванням, проте для поточної реалізації обрано простіший варіант, де простота важливіша за максимальну продуктивність.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

Останній алгоритм – генерація восьмисимвольного коду запрошення до класу. Він реалізований у ClassroomService через криптографічно безпечний генератор `RandomNumberGenerator.GetBytes(8)`: кожен з восьми байтів береться за модулем 32 і використовується як індекс у алфавіті з 32 символів – великих латинських літер без I та O плюс цифр від 2 до 9. Виключення схожих символів зменшує кількість помилок при ручному введенні. Унікальність гарантується подвійно – унікальним індексом на полі `InviteCode` та циклом повторної генерації при колізії. Загальна кількість комбінацій 32^8 становить близько 1.1 трильйона, тож імовірність колізії за реалістичної кількості класів вкрай низька.

2.4. Реалізація функціоналу платформи

Цей параграф присвячено інженерній стороні проекту – як архітектура, описана раніше, втілена у коді. Розглянуто структуру проекту, технологічні рішення на бекенді й фронтенді, наведено фрагменти найцікавіших ділянок реалізації та подано діаграму компонентів системи. Повні лістинги ключових модулів винесено у Додаток Б.

Кодову базу організовано як монорепозиторій на основі `npm workspaces`: бекенд, фронтенд і спільні бібліотеки лежать в одному Git-репозиторії, але збираються незалежно. Такий підхід синхронізує інтерфейси між клієнтом і сервером, бо при додаванні нового поля у DTO одночасно правляться серверний C#-клас і клієнтський TypeScript-інтерфейс у спільному пакеті.

Серверна частина побудована на ASP.NET Core 9 за класичною трирівневою схемою: контролери приймають HTTP-запити, делегують роботу сервісам, а ті звертаються до бази через Entity Framework Core [19]. Усі залежності реєструються у DI-контейнері методом `AddScoped`, що створює окремий екземпляр сервісу на кожен запит – це оптимальний час життя і для бізнес-сервісів, і для `DbContext`. Центральною точкою конфігурації є файл `Program.cs`, де реєструються контекст бази, ASP.NET Core Identity, схема JWT-

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

автентифікації, доменні сервіси, SMTP-служба та CORS-політика. У кінці файлу формується конвеєр обробки запитів – статичні файли, маршрутизація, автентифікація, TokenVersionMiddleware, авторизація і власне контролери. Порядок міدلварів критичний: TokenVersionMiddleware має йти після UseAuthentication, інакше у нього не буде ідентифікованого користувача для перевірки.

Конфігурація Entity Framework Core винесена у клас ApplicationDbContext. Його метод OnModelCreating містить опис усіх обмежень, індексів і глобальних фільтрів – те, що неможливо описати атрибутами на полях моделей. Найбільш цікаві фрагменти конфігурації наведено нижче.

```
modelBuilder.Entity<ApplicationUser>()
    .HasQueryFilter(u => !u.IsDeleted);
modelBuilder.Entity<Classroom>()
    .HasIndex(c => c.InviteCode)
    .IsUnique();
modelBuilder.Entity<ClassroomMembership>()
    .HasIndex(m => new { m.ClassroomId, m.UserId })
    .IsUnique();
modelBuilder.Entity<Submission>()
    .HasIndex(s => new { s.AssignmentId, s.StudentId })
    .IsUnique();
```

Перший фрагмент встановлює глобальний фільтр на сутність користувача: будь-який LINQ-запит автоматично доповнюється умовою WHERE IsDeleted = false, тому видалені користувачі не потраплять у вибірку навіть без явного фільтра. Це безпечніше за ручне додавання умов у кожен запит. Інші три рядки описують унікальні індекси, що захищають структуру від некоректних станів – код запрошення класу не може повторитися, користувач не може двічі приєднатися до того ж класу, а задача завдання тим самим учнем не може існувати у двох екземплярах.

Безпека сесій реалізована через зв'язку класу TokenService і міدلвара TokenVersionMiddleware. TokenService відповідає за створення JWT-токенів і формує набір claims, які потім вшиваються у токен. Ключова частина методу – формування претензій, що включають службове поле tokenVersion:

```
var claims = new List<Claim>
{
```

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

```

new(JwtRegisteredClaimNames.Sub, user.Id),
new(JwtRegisteredClaimNames.Email, user.Email!),
new(ClaimTypes.NameIdentifier, user.Id),
new("tokenVersion", user.TokenVersion.ToString())
};

```

```

foreach (var role in await _userManager.GetRolesAsync(user))
    claims.Add(new Claim(ClaimTypes.Role, role));

```

Перевірка tokenVersion відбувається у міدلварі при кожному автентифікованому запиті. Якщо значення в токени не збігається з актуальним у базі даних, міدلвар миттєво повертає 401-у відповідь, не пропускаючи запит далі по конвеєру.

```

var versionClaim = user.FindFirst("tokenVersion")?.Value;
if (versionClaim != null &&
    int.TryParse(versionClaim, out var v) &&
    v != dbUser.TokenVersion)
{
    context.Response.StatusCode = 401;
    await context.Response.WriteAsJsonAsync(
        new { error = "Token is no longer valid." });
    return;
}
await _next(context);

```

Це і дає можливість миттєво виганяти користувача з усіх активних сесій без потреби вести окремий чорний список токенів – просте інкрементування TokenVersion у базі робить недійсними всі видані до того токени конкретного користувача. Повний код міدلвару наведено у Додатку Б.

Логіка автоматичного оцінювання сконцентрована у статичному класі GradingService. Він не має жодних залежностей і працює як набір чистих функцій – це робить його ідеальним для модульного тестування без використання моків. Ключовий метод Grade обчислює відносні похибки за метриками і повертає підсумкову оцінку.

```

foreach (var (key, expVal) in expected)
{
    if (!observed.TryGetValue(key, out var actVal)) continue;
    var absError = Math.Abs(actVal - expVal);
    var relError = Math.Abs(expVal) < 1e-12
        ? (Math.Abs(actVal) < 1e-12 ? 0.0 : 1.0)
        : absError / Math.Abs(expVal);
    rows.Add(new GradingRow(key, expVal, actVal, absError, relError));
}
var score = rows.Average(r => Math.Max(0.0, 1.0 - r.RelError));

```

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

Константа 1e-12 тут не випадкова – це класичний поріг для порівняння чисел з плаваючою комою на «практичний нуль». Реалізація через Average і Math.Max гарантує, що підсумкова оцінка завжди лежить у діапазоні від 0 до 1 без явного клампування.

Доменна логіка приєднання до класу теж заслуговує окремої уваги, бо ілюструє неочевидний бізнес-сценарій. Метод JoinAsync у ClassroomService автоматично визначає роль нового учасника на основі глобальної ролі в Identity.

```
var classroom = await _db.Classrooms
    .Include(c => c.Memberships)
    .FirstOrDefaultAsync(c => c.InviteCode == code && !c.IsArchived);
if (classroom == null) return null;
if (classroom.Memberships.Any(m => m.UserId == userId)) return null;

var user = await _userManager.FindByIdAsync(userId);
var assignedRole = await _userManager.IsInRoleAsync(user!, "Teacher")
    ? ClassroomRole.CoAuthor
    : ClassroomRole.Student;
```

Тобто якщо до класу за кодом приєднується інший викладач, він автоматично отримує локальну роль CoAuthor, а звичайний студент – Student. Це звільняє власника класу від необхідності вручну надавати співавторам спеціальні права – достатньо просто поділитися кодом запрошення.

Клієнтська частина реалізована на Angular 18 за сучасним підходом standalone-компонентів [22]. На відміну від класичного Angular з модулями NgModule, у standalone-моделі кожен компонент сам декларує свої залежності у полі imports, що значно спрощує структуру проекту. Маршрутизація налаштована з лінивим завантаженням – компоненти сторінок підвантажуються лише тоді, коли користувач переходить на відповідний маршрут, що зменшує початковий розмір бандлу.

```
export const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'login', component: LoginComponent },
  { path: 'sim/:moduleId',
    loadComponent: () => import('./pages/sim/sim.component')
      .then(m => m.SimComponent) },
  { path: 'my-classes',
    loadComponent: () => import('./pages/my-classes/my-classes.component')}
```

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

```

        .then(m => m.MyClassesComponent),
    canActivate: [authGuard] },
    { path: 'admin',
      loadComponent: () => import('./pages/admin/admin-layout/admin-
        layout.component')
        .then(m => m.AdminLayoutComponent),
      canActivate: [authGuard, adminGuard] }
  ];

```

Маршрути захищені через guards – невеликі функції, які перевіряють умови доступу ще до завантаження сторінки. authGuard перевіряє наявність валідного токена, adminGuard – додатково перевіряє роль користувача. Якщо умови не виконані, користувач перенаправляється на сторінку входу. Це працює як «передня лінія оборони» – навіть якщо хтось спробує напямую перейти на адмінську сторінку, фронтенд просто не покаже її, а навіть якщо покаже, сервер заблокує запит через атрибут авторизації на ендпоінті.

Стан додатку керується через механізм signals – це новий реактивний примітив Angular, що працює аналогічно реактивним змінним у Vue або Svelte. На відміну від попереднього підходу через Zone.js, signals не вимагають патчингу глобальних API і дозволяють Angular точково перемальовувати лише ті частини DOM, які залежать від конкретного зміненого signal. Для проекту це особливо важливо, бо паралельно з фізичною симуляцією у Babylon.js на сторінці працює звичайний Angular-інтерфейс, і будь-яке зайве перемальовування могло б знизити частоту кадрів симуляції.

Найскладніша і найоб'ємніша частина клієнтської реалізації – це фізичні модулі симуляцій на Babylon.js [23]. Кожен модуль складається з трьох файлів: типів параметрів і стану, описання сцени і інтегратора фізики та допоміжного рендерера, який керує візуальним представленням. Розглянемо коротко, як побудовано модуль маятника – це найпростіший з фізичних модулів і добре ілюструє загальний підхід.

```

const omegaSquared = gravity / length;
const angularAcceleration = -omegaSquared * Math.sin(theta)
    - damping * thetaDot;
thetaDot += angularAcceleration * dt;
theta += thetaDot * dt;

```

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

Тут реалізовано напівнеявну схему Ейлера для класичного рівняння математичного маятника: спочатку обчислюється кутове прискорення з урахуванням гравітації та демпфування, потім оновлюється кутова швидкість і нарешті сам кут. Схема стабільніша за пряму Ейлера і простіша за Рунге-Кутту, що добре підходить для інтерактивної симуляції з кроком близько 16 мс (60 кадрів за секунду). Аналогічна логіка реалізована у решті 11 модулів: цикл рендерингу Babylon.js викликає метод оновлення фізики, який інтегрує рівняння на один крок dt і оновлює позиції 3D-об'єктів у сцені.

Поza основними модулями реалізовано три допоміжні підсистеми. PDF-звіти за результатами задачі формує LabReportService через бібліотеку QuestPDF: документ описується декларативно у C#-кодi, створюється у пам'яті як байтовий масив і зберігається у таблиці SubmissionArtifacts, звідки видається учневі за окремим ендпоінтом GET-запитом. У звіті відображаються параметри симуляції, передбачення учня, фактичні значення метрик, таблиця похибок з кольоровим виділенням розбіжностей та підсумкова оцінка з коментарем учня. Email-сповіщення надсилаються через стандартний клас SmtпClient за інтерфейсом IEmailService з реалізацією у SmtпEmailService. Налаштування SMTP-сервера – адреса, порт, реквізити автентифікації та зворотна адреса – читаються з appsettings.json; самі листи у базі не зберігаються, бо платформа не претендує на роль повноцінної поштової системи і не відстежує подальшу долю надісланих повідомлень. Аналітика на клієнті будується на бібліотеці Chart.js. Дані для графіків надходять з ендпоінтів AnalyticsController, що делегують роботу серверному AnalyticsService. Останній виконує агрегаційні LINQ-запити до бази: для теплової карти підраховується середній бал учня за кожною з шести категорій фізики, для графіка успішності класу – середній бал за кожним завданням, а для динаміки конкретного учня – хронологічно впорядкований ряд його оцінок. Підсумувати все вище викладене допомагає діаграма компонентів

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

системи (рис. 2.6), на якій показано основні логічні блоки платформи та зв'язки між ними.

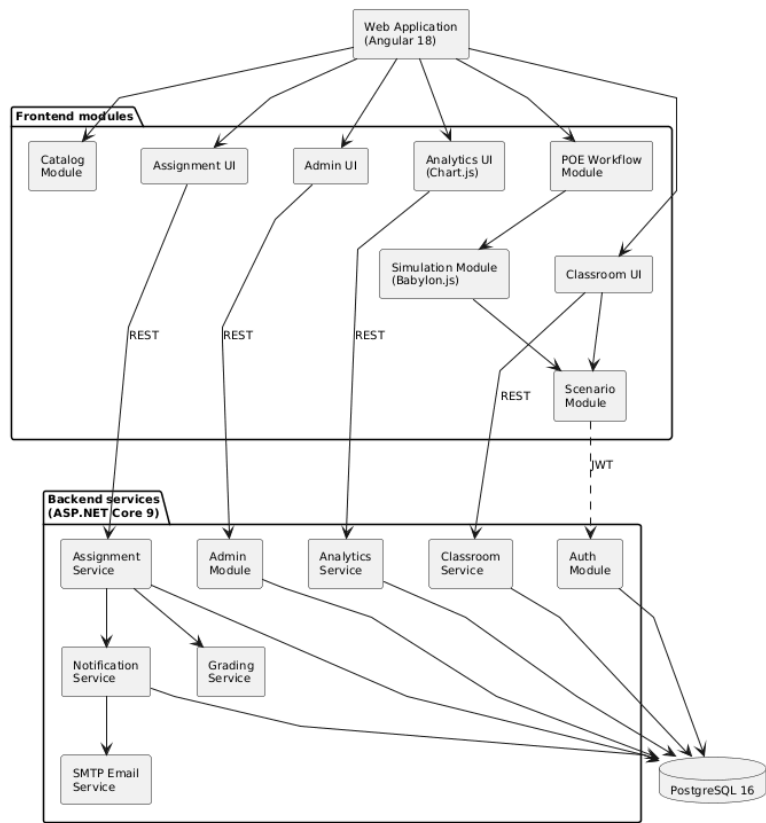


Рис. 2.6. Діаграма компонентів системи

На діаграмі представлено десять логічних компонентів системи. Інфраструктурний рівень формують Auth Module – модуль автентифікації з видачею JWT-токенів, ротацією refresh-токенів та механізмом TokenVersion, і Notification Module – модуль внутрішніх та email-сповіщень у зв'язці з SmtпEmailService. На рівні доменної логіки розташовані Classroom Module з керуванням віртуальними класами та кодами запрошення, Assignment Module з підтримкою чотирьох типів завдань і автоматичним оцінюванням через GradingService, Analytics Module з тепловими картами та графіками успішності, а також Admin Module з керуванням користувачами, заявками на зміну ролі та журналом аудиту.

Окрему групу складають клієнтсько-симуляційні модулі. Catalog Module відображає каталог симуляцій з фільтрацією за категорією і рівнем складності. Simulation Module виконує тривимірні симуляції у браузері засобами Babylon.js і містить дванадцять незалежних фізичних моделей. POE Workflow

Module реалізує трьохстадійний педагогічний цикл з таймером мінімального часу спостереження. Scenario Module зберігає й завантажує користувацькі конфігурації симуляцій.

Модулі взаємодіють між собою у логіці «знизу вгору»: інфраструктурні (Auth, Notification) обслуговують доменні (Classroom, Assignment, Analytics), а ті, у свою чергу, доступні через окремі сторінки фронтенд-додатку. Така структура дозволяє розвивати систему модульно – наприклад, додавання нового типу завдання вимагає змін лише у модулі Assignment та супутніх UI-компонентах, не торкаючись інших підсистем.

Загалом реалізація платформи побудована на простих і перевірених рішеннях замість екзотичних патернів: стандартний DI, стандартний Identity, стандартні signals. Кожне ключове рішення – м'яке видалення, миттєва інвалідація сесій, гнучкі JSON-поля, лінива маршрутизація з guards – має конкретне обґрунтування у вимогах. Така структура дозволяє новому розробнику зорієнтуватися у проекті за кілька годин, а нова функціональність зазвичай зводиться до однотипних змін у моделі, сервісі та контролері.

Висновки до другого розділу

У другому розділі виконано повний цикл проектування вебплатформи – від формалізації вимог до опису реалізації ключових компонентів.

На етапі визначення варіантів використання виокремлено п'ять ролей користувачів та деталізовано вісім ключових прецедентів, що покривають основні сценарії роботи з платформою. Об'єктну модель побудовано на десяти доменних сутностях і десяти сервісах, які реалізують усю бізнес-логіку і реєструються у DI-контейнері. Розподіл відповідальності між сутностями, сервісами, контролерами та DTO дозволяє розвивати систему модульно.

При проектуванні бази обрано PostgreSQL 16 і Entity Framework Core 9 за підходом Code First. Розроблено схему з десяти доменних таблиць з налаштованими режимами каскадування під конкретні сценарії; динамічний контент завдань і сценаріїв зберігається у JSON-полях, що дозволило обійтися

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

без розгалуження схеми на десятки додаткових таблиць. Передбачено м'яке видалення через глобальний фільтр, унікальні складені індекси та індекс пришвидшення для запиту непрочитаних сповіщень. Накопичені 11 міграцій документують ітеративну еволюцію схеми.

Алгоритмічна частина описана у вигляді діаграми активності системи та окремої діаграми РОЕ-циклу. Сформульовано формули автоматичного оцінювання для чотирьох типів завдань: середнє арифметичне відносних похибок для РОЕ та сценарію, частка правильних відповідей для тесту, частка зарахованих полів з абсолютним допуском для числової задачі. Описано авторизацію з парою JWT-токенів і ротацією refresh-токена, механізм миттєвої інвалідації сесій через лічильник TokenVersion та алгоритм генерації унікального коду запрошення.

На етапі реалізації описано організацію коду як монорепозиторію зі спільним пакетом TypeScript-типів, трирівневу структуру бекенду на ASP.NET Core 9, фронтенд на Angular 18 зі standalone-компонентами та signals, захист маршрутів через guards на клієнті й атрибути авторизації на сервері, а також тривимірні симуляції на Babylon.js з власними інтеграторами фізичних рівнянь. Розроблена проектна документація – діаграми використання, об'єктна модель, ER-схема бази, діаграми активності та компонентів – дає повноцінне уявлення про внутрішню будову системи і відкриває перехід до її практичного розгортання й тестування, описаних у наступному розділі.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ МОДЕЛЮВАННЯ ФІЗИЧНИХ ПРОЦЕСІВ

3.1. Порядок встановлення та налаштування параметрів платформи

Для забезпечення публічного доступу до платформи й перевірки її роботи у реальних умовах було проведено розгортання у хмарі. Production-середовище використовує безкоштовні тарифи трьох незалежних провайдерів і не вимагає окремого серверного обладнання, що робить його придатним для дипломного демонстраційного проекту з мінімальними витратами на інфраструктуру.

Платформа розділена на три шари, кожен з яких розгортається на спеціалізованому хмарному сервісі. Розподіл відображає типову архітектуру сучасних веб-застосунків: статичний фронтенд на CDN, бекенд у контейнері та керована база даних. Кожен з провайдерів відповідає за свою задачу й може масштабуватися чи замінюватися окремо без впливу на інші шари.

Клієнтську частину розгорнуто на Vercel – платформі для статичного хостингу з глобальною мережею доставки контенту [30]. Vercel розповсюджує файли збірки Angular-застосунку через edge-ноди по всьому світу, що мінімізує час завантаження сторінки для користувача незалежно від його географічного розташування. Перевага Vercel у автоматичній підтримці HTTPS-сертифікатів від Let's Encrypt і простому механізмі перенаправлення запитів через декларативний файл `vercel.json`.

Серверну частину розгорнуто на Railway – платформі-як-сервіс, що нативно підтримує .NET 9 завдяки автоматичному buildpack-механізму Railpack [31]. На відміну від інших хмарних провайдерів, які вимагають Dockerfile для запуску .NET-застосунків, Railway самостійно виявляє csproj-файл у репозиторії й виконує стандартну послідовність команд `dotnet restore`, `dotnet publish` і запуск отриманого артефакту. Цей підхід усуває потребу в

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

ручному написанні Dockerfile і дозволяє публікувати оновлення одним push-ом в Git.

Базу даних розгорнуто на Neon – серверлес-провайдері PostgreSQL з нульовою вартістю простою [32]. Neon автоматично призупиняє обчислювальний вузол бази при відсутності запитів, що економить ресурси у безкоштовному тарифі, і відновлює його за кілька секунд при першому новому запиті. Постгрес-сумісність повна, тому Entity Framework Core й Npgsql-провайдер працюють без жодних змін у коді.

Загальна архітектура production-розгортання та маршрут обробки запиту показані на рисунку 3.1.

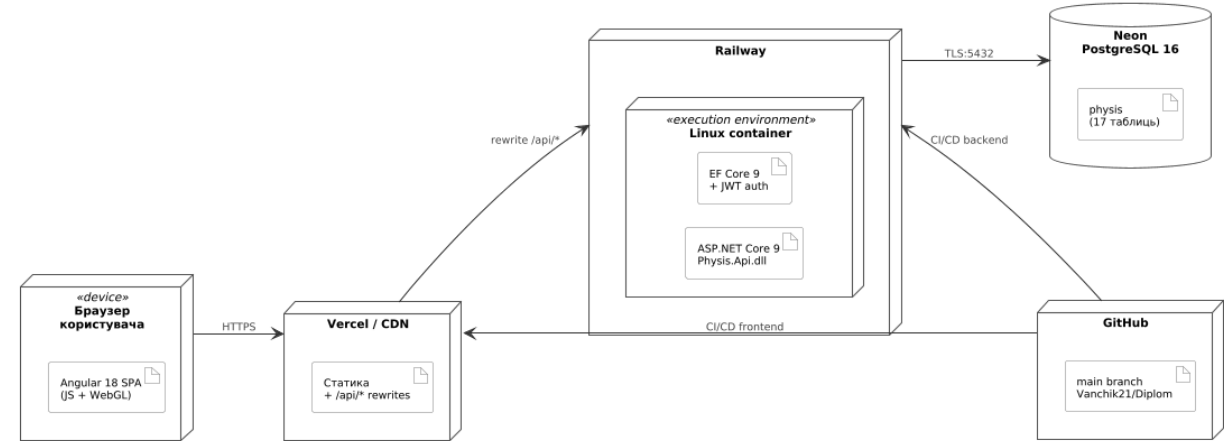


Рис. 3.1. Архітектура production-розгортання платформи

Маршрут типового запиту до платформи проходить наступним чином. Браузер користувача звертається до доменного імені Vercel, отримує статичні файли Angular-застосунку з найближчої edge-ноди. Далі клієнтський код виконує REST-запит за шляхом /api/* – оскільки шлях відносний, запит за замовчуванням адресується тому самому походженню. Vercel розпізнає це за правилами rewrites у файлі vercel.json і прозоро перенаправляє запит на Railway. Завдяки серверній природі перенаправлення для браузера обмін лишається одно-походженням, тому механізм CORS не задіюється – це суттєво спрощує конфігурацію безпеки. Railway передає запит у контейнер з ASP.NET Core застосунком, який виконує перевірку JWT-токена, звертається до Neon через захищене TLS-з’єднання на порту 5432 і повертає результат у зворотному порядку.

Налаштування платформи у production-середовищі повністю керується через змінні оточення, що відповідає підходу Twelve-Factor App [33]. Конфігураційні файли appsettings.json і appsettings.Development.json не використовуються у проді – замість них .NET автоматично читає environment variables з префіксом, що відповідає шляху налаштування у JSON. Перелік змінних оточення наведено в таблиці 3.1.

Таблиця 3.1

Змінні оточення production-середовища

Змінна оточення	Призначення
ConnectionStrings__Default	Рядок підключення до Postgres у форматі Npgsql з параметрами TLS
Jwt__Secret	64-символьний секретний ключ для підпису JWT-токенів
Jwt__Issuer, Jwt__Audience	Поля видавця і отримувача токена для додаткової валідації
Jwt__AccessTokenExpirationMinutes	Час життя access-токена в хвилинах (за замовчуванням 60)
Jwt__RefreshTokenExpirationDays	Час життя refresh-токена в днях (за замовчуванням 7)
AdminSeed__Email	Електронна адреса акаунту адміністратора, що створюється автоматично при першому запуску
AdminSeed__UserName	Ім'я облікового запису адміністратора
AdminSeed__Password	Початковий пароль адміністратора (≥12 символів)
AllowedOrigins__0	Дозволений CORS-походження – домен фронтенду на Vercel
ASPNETCORE_ENVIRONMENT	Режим роботи .NET-застосунку, встановлюється у значення Production

Процес першого розгортання складається з трьох незалежних кроків – створення бази даних, публікація бекенду та публікація фронтенду. Спочатку у консолі Neon створюється проект з регіоном AWS Europe Central 1

(Франкфурт), вибраним з огляду на географічну близькість до користувачів. Neon генерує рядок підключення у вигляді PostgreSQL URI, який перетворюється у формат Npgsql для подальшого використання у .NET. У рядок підключення додаються параметри SSL Mode=Require і Trust Server Certificate=true – перший обов'язковий, бо Neon приймає лише шифровані з'єднання, другий потрібен через те, що Linux-контейнер Railway не має оновленого CA-bundle для перевірки сертифіката Neon. (рис. 3.2).

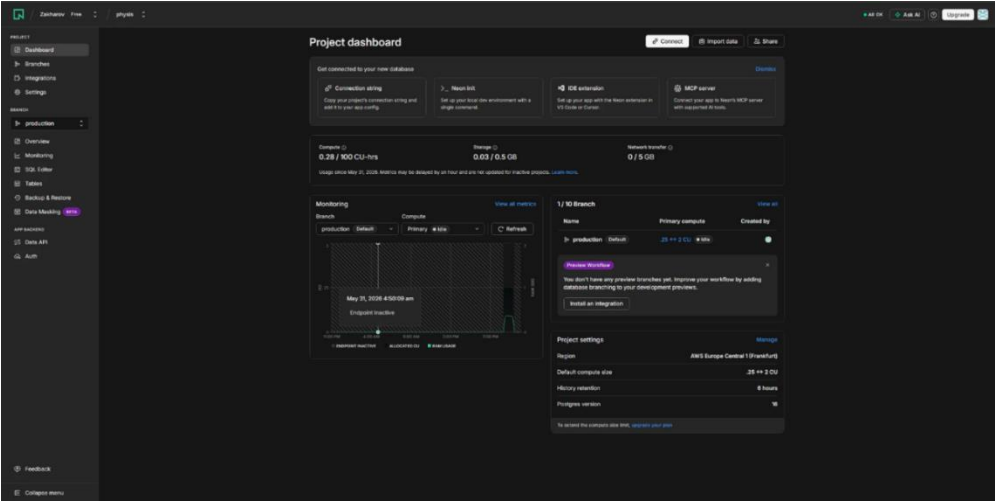


Рис. 3.2. Консоль Neon з активним проектом

Далі у Railway створюється проект з прив'язкою до GitHub-репозиторію Vanchik21/Diplom, кореневим каталогом apps/backend і набором змінних оточення з таблиці 3.1. Railpack автоматично виявляє .NET-проект і запускає збірку, після чого Railway генерує публічний домен `diplom-production-cba6.up.railway.app`, через який бекенд стає доступним для зовнішніх запитів (рис. 3.3).

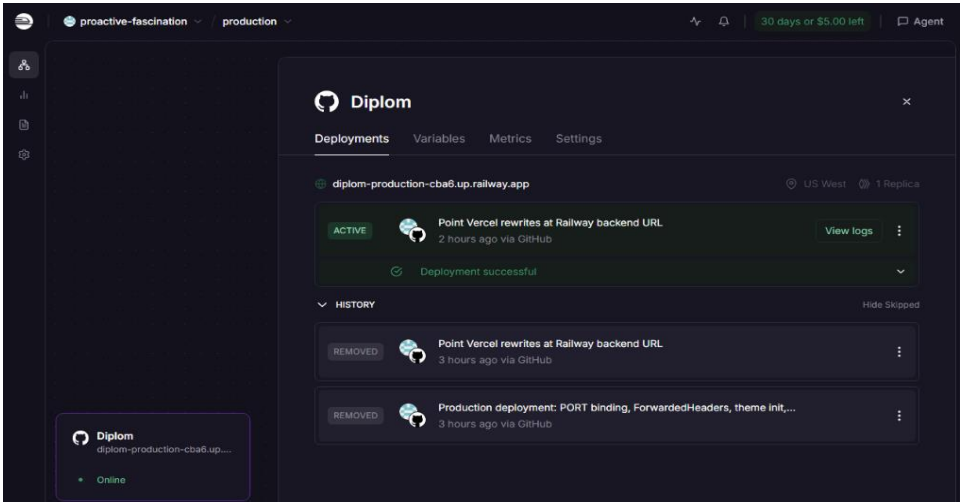


Рис. 3.3. Сервіс бекенду на Railway зі статусом Active

Третім кроком налаштовується Vercel: створюється проект, прив'язується той же GitHub-репозиторій, вказується кореневий каталог apps/frontend. Vercel автоматично визначає Angular як використовуваний фреймворк і застосовує налаштування з файлу vercel.json – команда збірки, директорія результату та команда встановлення залежностей. Файл vercel.json також містить правила rewrites, які перенаправляють запити /api/* і /uploads/* на домен Railway. Після першого деплою отримується робочий URL фронтенду – diplom-frontend-five.vercel.app, який підставляється у змінну AllowedOrigins__0 Railway-сервісу (рис. 3.4).

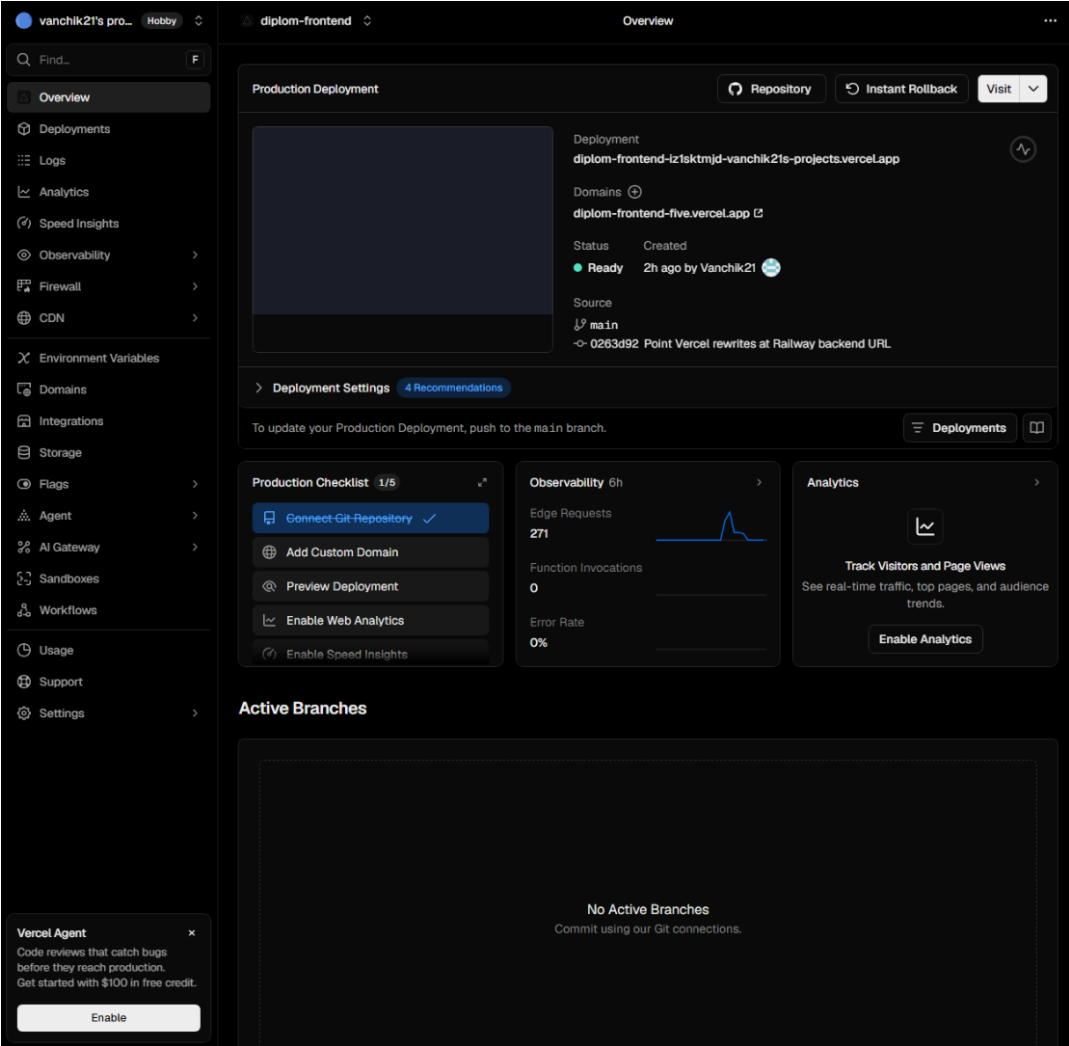


Рис. 3.4. Дашборд Vercel з активним production-деплоєм

Підтвердженням успішного розгортання слугують два знімки робочого застосунку у production-середовищі. Жива головна сторінка платформи показана на рисунку 3.5, а адміністративна панель з даними з production-бази – на рисунку 3.6.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

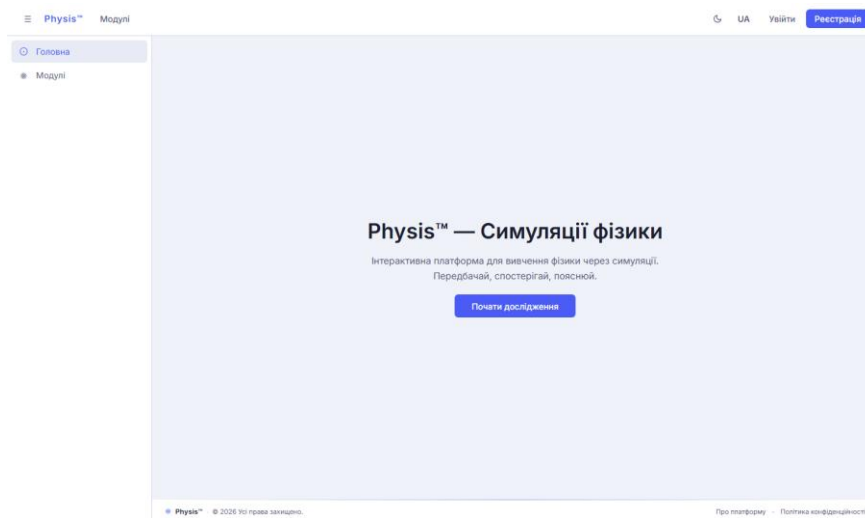


Рис. 3.5. Жива головна сторінка платформи на Vercel

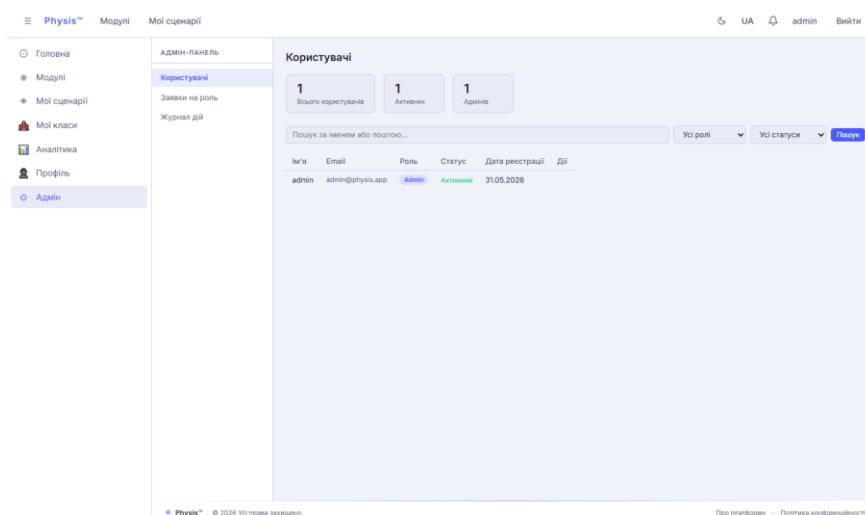


Рис. 3.6. Адміністративна панель платформи у production-середовищі

Подальші оновлення коду розповсюджуються автоматично завдяки інтеграції GitHub-репозиторію з обома платформами. При кожному push у гілку main GitHub надсилає webhook одночасно на Vercel і Railway, які паралельно запускають збірку нової версії. Перебудова при цьому відбувається лише у тому сервісі, чий код був реально змінений: Vercel реагує на зміни у apps/frontend, Railway — на зміни у apps/backend завдяки налаштуванню Watch Paths. Це дозволяє розгортати оновлення інтерфейсу без зайвого простою бекенду й навпаки. Уся історія деплоїв доступна у відповідних дашбордах, і за потреби можна відкотитися до будь-якої попередньої версії одним кліком.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

Завдяки описаному підходу платформа доступна за публічним посиланням цілодобово й не потребує додаткових витрат на серверне обладнання, що робить її придатною для дипломного демонстраційного проекту й одночасно відображає підходи, що використовуються у промислових проектах подібного масштабу.

3.2. Структура інтерфейсу платформи для моделювання складних фізичних процесів у браузері

Інтерфейс платформи побудований як односторінковий вебзастосунок з єдиним layout-каркасом – топбаром угорі, опціональним сайдбаром ліворуч та основним змістом сторінки у центрі. Топбар містить логотип з посиланням на головну, навігаційні посилання, перемикач теми (світла або темна), перемикач мови (українська або англійська), а для авторизованих користувачів – дзвоник сповіщень з випадаючим списком, ім'я користувача з посиланням на профіль і кнопку виходу. Сайдбар працює у двох станах – згорнутому та розгорнутому – і підсвічує активний пункт меню. Сторінки входу і реєстрації відображаються поза цим каркасом, на чистому тлі, для максимальної концентрації уваги.

Каркас застосунку реалізовано як standalone-компонент Shell, який підключається через router-outlet і охоплює переважну більшість маршрутів – окрім /login та /register, що рендеряться поза ним. Завдяки цьому переходи між сторінками не перемальовують топбар і сайдбар, а оновлюється лише центральна область. Активний пункт у сайдбарі визначається через директиву routerLinkActive і підсвічується одночасно з відповідним маршрутом, тож користувач завжди бачить своє поточне місце у застосунку. Для аутентифікованих користувачів у сайдбарі додатково з'являються пункти «Мої сценарії», «Мої класи», «Аналітика» та «Профіль», а адміністратору доступний окремий пункт «Адмін».

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

Перемикання теми та мови виконується миттєво без перезавантаження сторінки. Тема зберігається у localStorage під ключем physis_theme і застосовується через атрибут data-theme на елементі html, який каскадно перевизначає змінні CSS у головному файлі стилів. Темна тема встановлена за замовчуванням, що відповідає сучасним тенденціям освітніх платформ і знижує навантаження на зір під час тривалих сесій. Мова інтерфейсу керується бібліотекою ngx-translate і підвантажується з файлів i18n/uk.json та i18n/en.json; українська встановлена за замовчуванням, у топбарі відображається код активної мови у вигляді «UA» або «EN».

Дзвоник сповіщень у топбарі показує бейдж з кількістю непрочитаних повідомлень і за кліком відкриває випадаючий список з останніми десятима записами – здача роботи учнем, виставлена оцінка, прийняття заявки на роль. Кожне сповіщення містить тип, короткий опис, час події і відмічається як прочитане при кліку. Для невеликих екранів сайдбар автоматично згортається до ширини у 56 пікселів, залишаючи лише іконки, а топбар скорочує підписи навігаційних посилань до значків. Нижче розглянуто основні екрани платформи у порядку звичайного користування авторизованим відвідувачем (рис. 3.7).

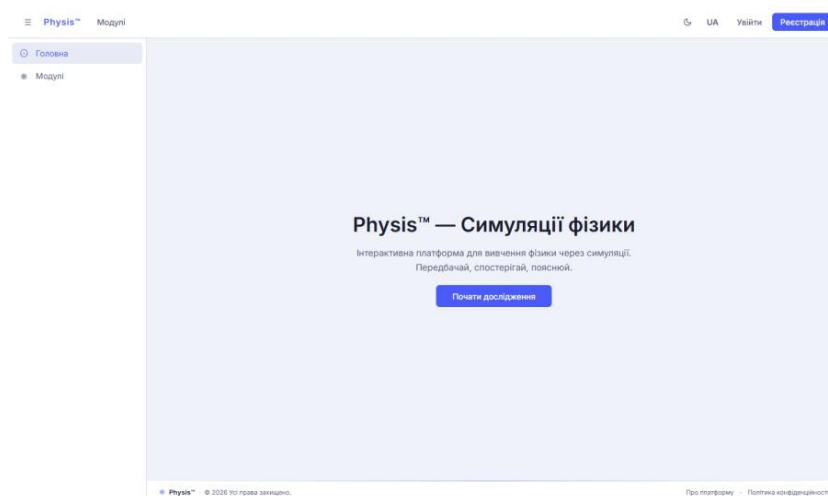


Рис. 3.7. Головна сторінка платформи

Каталог симуляцій є центральною точкою навчального контенту (рис. 3.8). На сторінці відображено сітку карток із дванадцятьма модулями, кожна картка містить мініатюру, назву, категорію та рівень складності. Угорі

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

сторінки розташовано пошук за назвою і два набори кнопок-фільтрів – за категорією та за рівнем складності. Якщо за поточним фільтром нічого не знайдено, користувач бачить порожній стан з пропозицією змінити критерії пошуку.

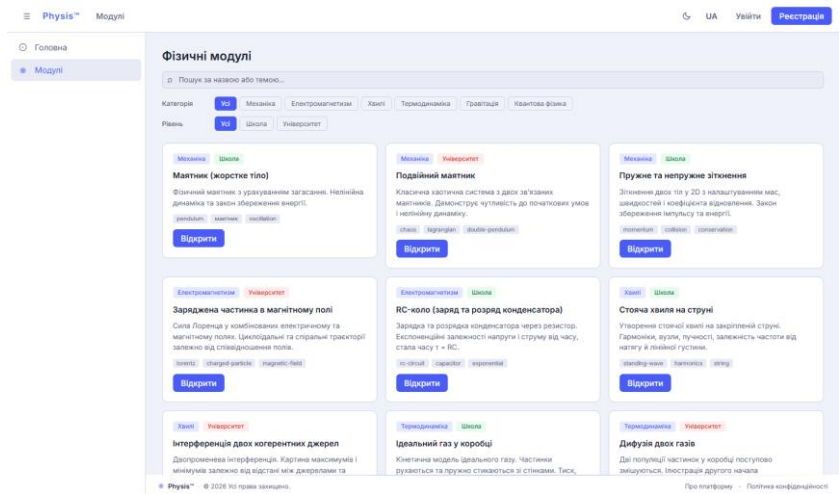


Рис. 3.8. Каталог симуляцій з фільтрами

Для роботи з персональними функціями платформи потрібна автентифікація. Сторінка входу містить компактну форму з полями email і пароллю, посилання на сторінку реєстрації та інформативні стани помилок – окремо для невірних облікових даних і окремо для деактивованих акаунтів (рис. 3.9).

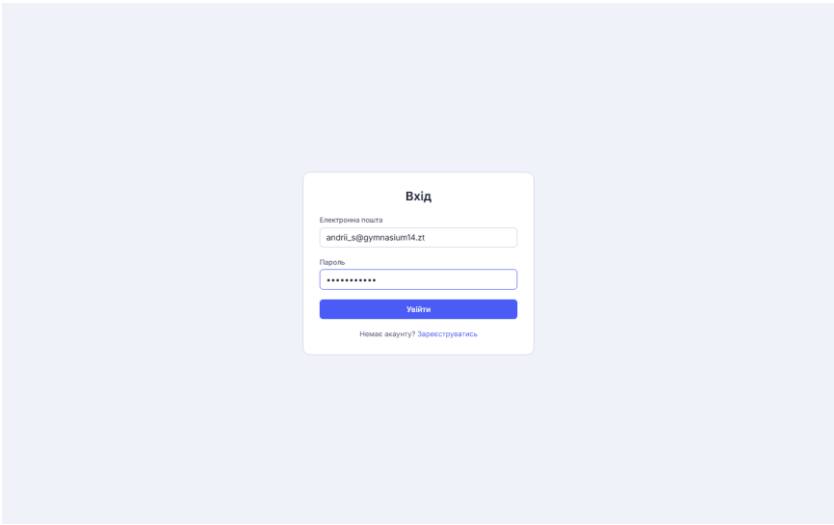


Рис. 3.9. Сторінка входу

Сторінка реєстрації побудована аналогічно і містить чотири поля – email, ім'я облікового запису, пароль та вибір стартової ролі через radio-кнопки «Учень» або «Викладач» (рис. 3.10). Валідація відбувається на стороні клієнта:

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

пароль має містити щонайменше вісім символів, ім'я облікового запису – щонайменше три. Адмінську роль через звичайну реєстрацію отримати неможливо – вона видається лише через сід або вручну іншим адміністратором.

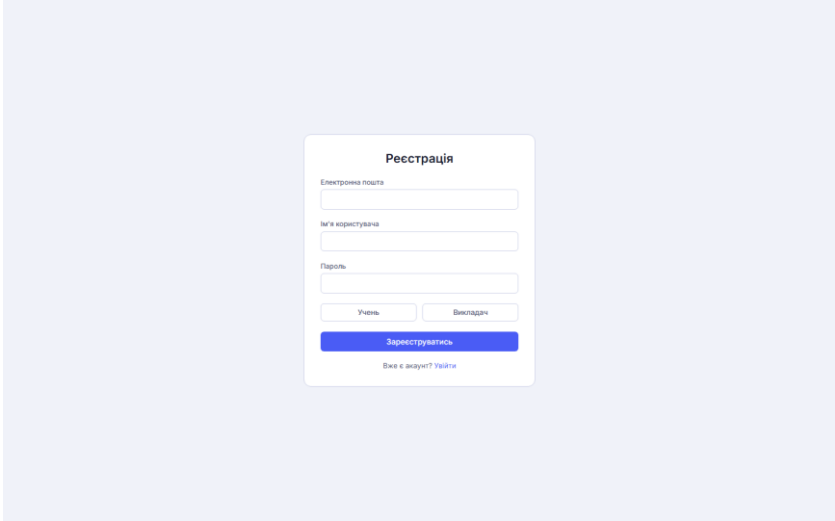


Рис. 3.10. Сторінка реєстрації з вибором ролі

Центральним елементом навчальної взаємодії є сторінка симуляції з реалізацією трьохстадійного циклу «Передбачай – Спостерігай – Пояснюй». На стадії «Передбачити» учень бачить тривимірну сцену з симуляцією на паузі та горизонтальну панель слайдерів зверху, через яку виставляє свої очікувані значення кожної метрики (рис. 3.11).

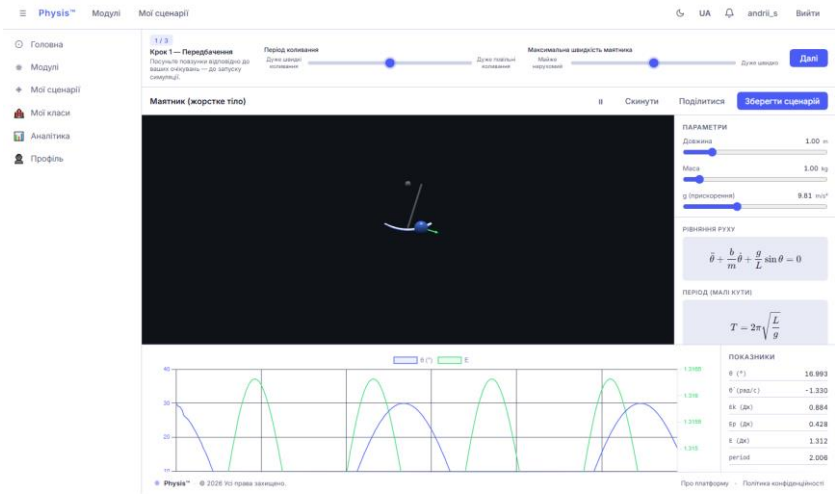


Рис. 3.11. РОЕ-цикл: стадія «Передбачити»

На стадії «Спостерігати» симуляція запускається, і над сценою з'являється прогрес-бар таймера, що відлічує реальний час спостереження від нуля до п'яти секунд (рис. 3.12). Поряд з прогрес-баром відображається

числовий лічильник у форматі «X / 5 с». До завершення інтервалу кнопка переходу до пояснення заблокована – це не дозволяє формально проскочити завдання без вникання у процес.

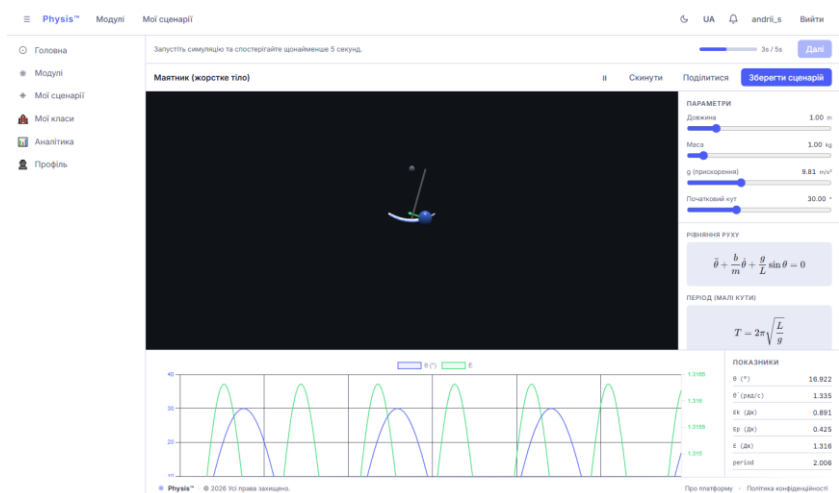


Рис. 3.12. РОЕ-цикл: стадія «Спостерігати» з таймером

На стадії «Пояснити» для кожної метрики будується картка порівняння з двома маркерами на слайдері – синій позначає прогноз учня, червоний – фактичний результат симуляції, поряд бейдж відносної похибки з кольоровим кодуванням (рис. 3.13). Нижче – опис фізичної моделі з формулами та поле для висновку учня.

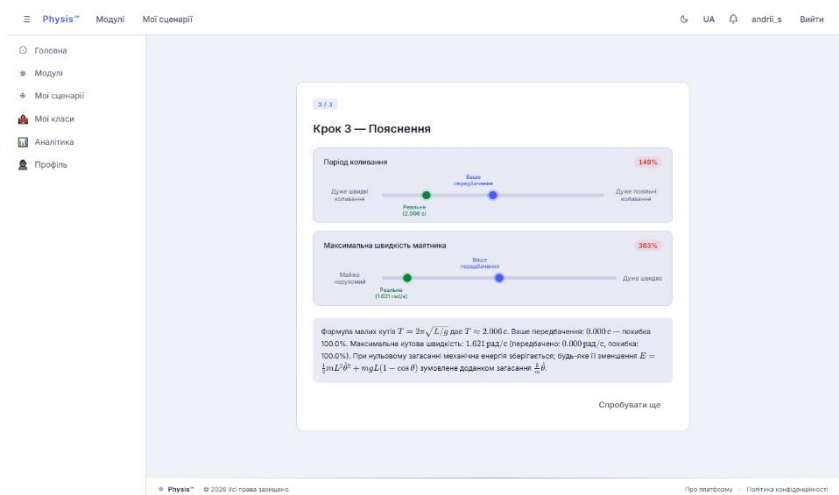


Рис. 3.13. РОЕ-цикл: стадія «Пояснити» з порівнянням метрик

Авторизований користувач отримує доступ до сторінки «Мої класи», де перелічено всі віртуальні класи, у яких він бере участь. Для викладача сторінка поділена на дві секції – «Викладаю» зі списком власних класів і «Співавтор» з

класами, до яких він приєднався як CoAuthor (рис. 3.14). Угорі сторінки є дві кнопки – «Створити клас» і «Запит на співавторство». Якщо у користувача ще немає класів, відображається пустий стан з підказкою.

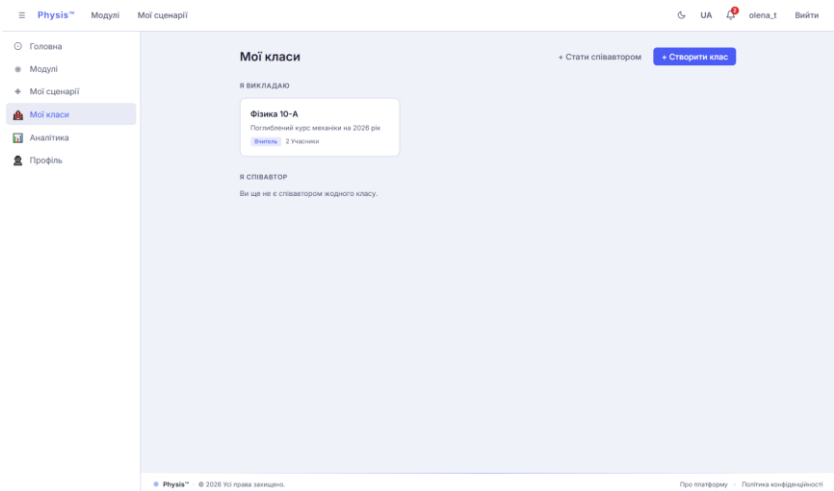


Рис. 3.14. Список класів викладача

Сторінка деталей класу містить три вкладки. Перша – «Members» зі списком учасників і можливістю видаляти їх із класу (рис. 3.15). Угорі сторінки відображено код запрошення з кнопкою копіювання у буфер обміну та кнопкою генерації нового коду на випадок, якщо старий потрапив до сторонніх осіб.

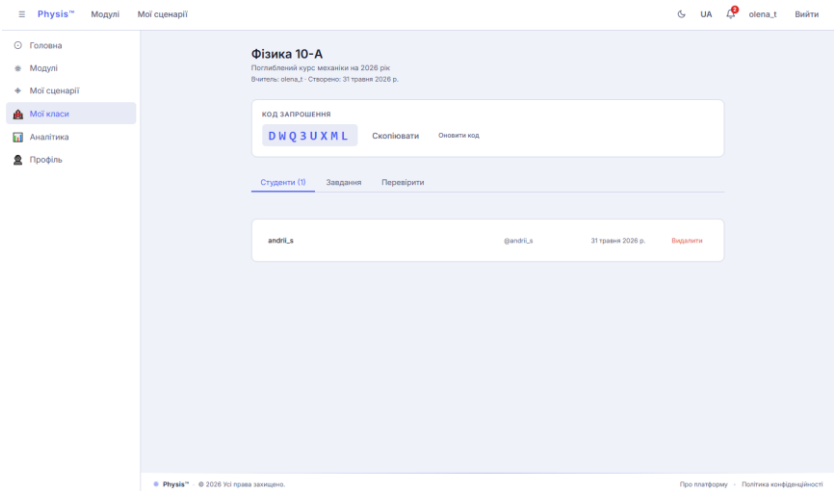


Рис. 3.15. Деталі класу: вкладка учасників

Друга вкладка – «Assignments» – містить перелік завдань класу, де кожен запис показує тип, назву і дедлайн (рис. 3.16). Викладач бачить кнопку «Створити завдання», яка відкриває модальне вікно з формою. Учень бачить тут свої здачі і вже виставлені оцінки, якщо роботи перевірені.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		70

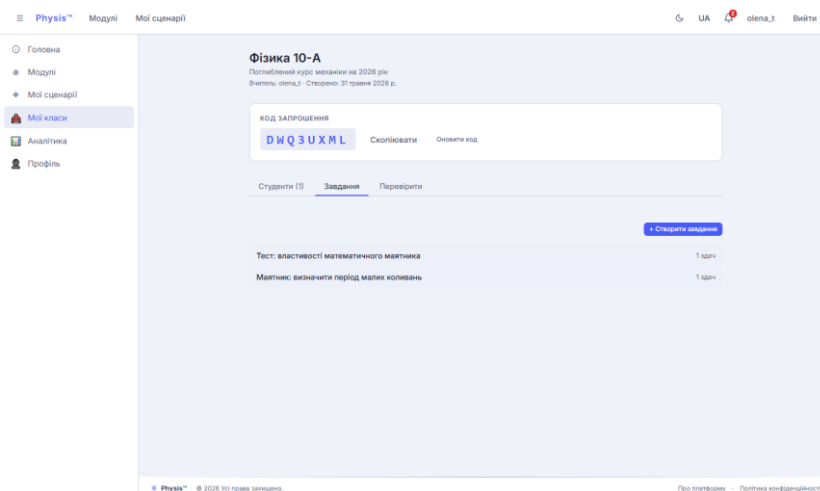


Рис. 3.16. Деталі класу: список завдань

Сторінка виконання завдання адаптується під його тип. Для тестового завдання відкривається модальне вікно з послідовністю питань і radio-кнопками вибору відповіді (рис. 3.17). Кнопка «Надіслати» активується лише після того, як учень обрав варіант для кожного питання. Після відправки відображається результат з підсвічуванням правильних і неправильних відповідей.

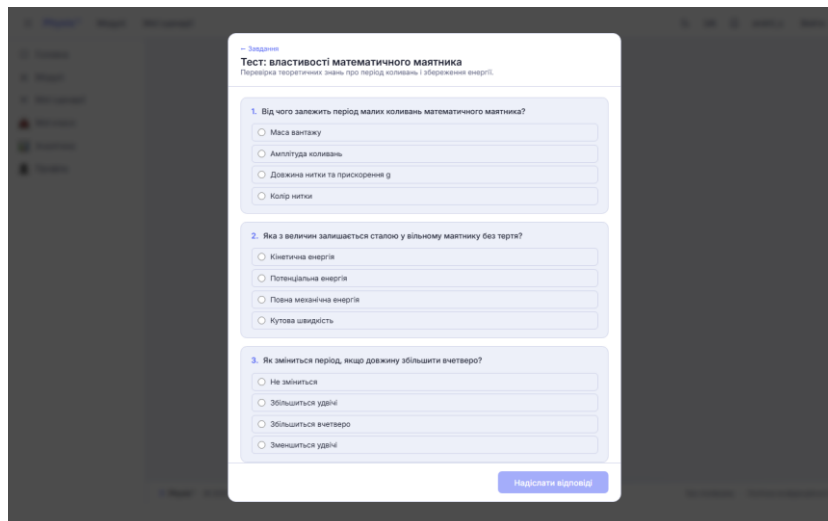


Рис. 3.17. Здача завдання типу «Тест»

Для завдання типу РОЕ сторінка виконання містить повноцінний цикл «Передбачай – Спостерігай – Пояснюй» зі симуляцією, аналогічний до режиму вільного дослідження, з додатковим полем для текстового висновку учня та кнопкою «Надіслати» наприкінці (рис. 3.18). Після відправки система обчислює оцінку за відносними похибками і формує PDF-звіт, доступний для завантаження.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

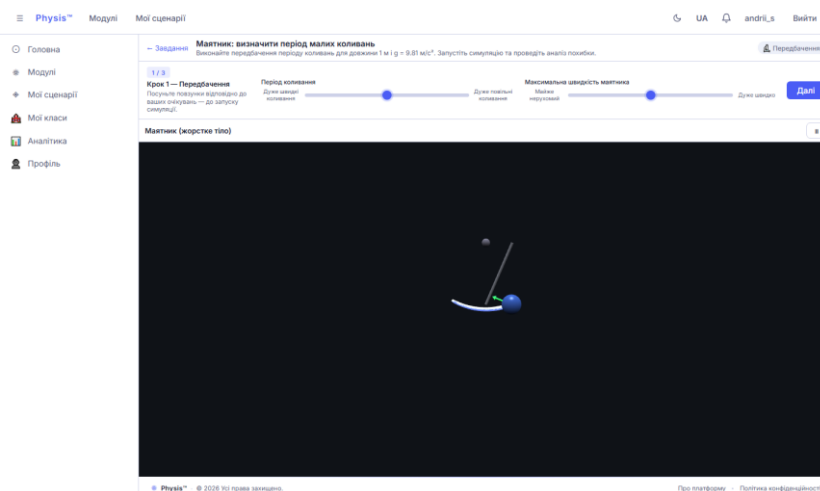


Рис. 3.18. Здача завдання типу РОЕ з полем висновку

Для викладача передбачено окрему сторінку перегляду задач (рис. 3.19). Інтерфейс розділений на дві колонки: ліворуч – список учнів з кольоровим маркером статусу задачі, праворуч – детальна інформація обраної задачі з формою ручного оцінювання. У формі викладач вводить бал від 0 до 100 і опціональний коментар, після чого натискає кнопку публікації результату – статус задачі переходить у ReviewedPublished, а учень отримує сповіщення.

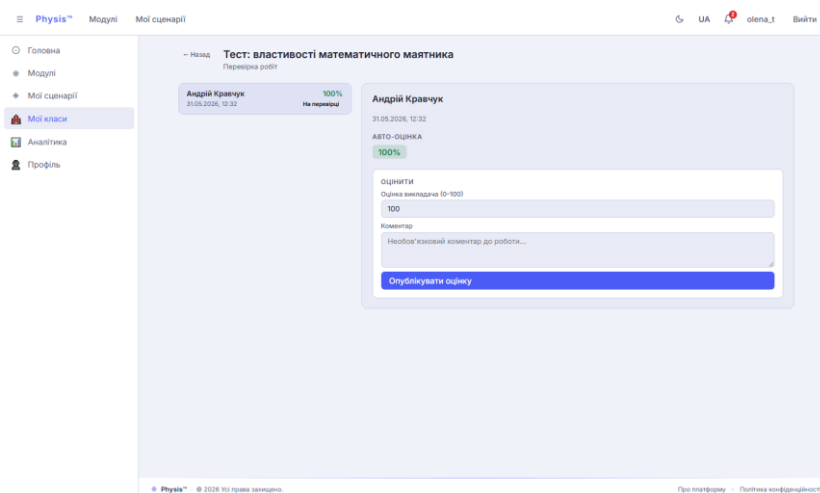


Рис. 3.19. Сторінка перевірки задач викладачем

Сторінка «Мої сценарії» містить персональну колекцію збережених конфігурацій симуляцій (рис. 3.20). Кожен сценарій представлений карткою з назвою, датою створення і двома кнопками – «Завантажити» для повернення до тієї самої конфігурації симуляції та «Видалити» для прибирання сценарію зі списку. Якщо колекція порожня, користувач бачить підказку.

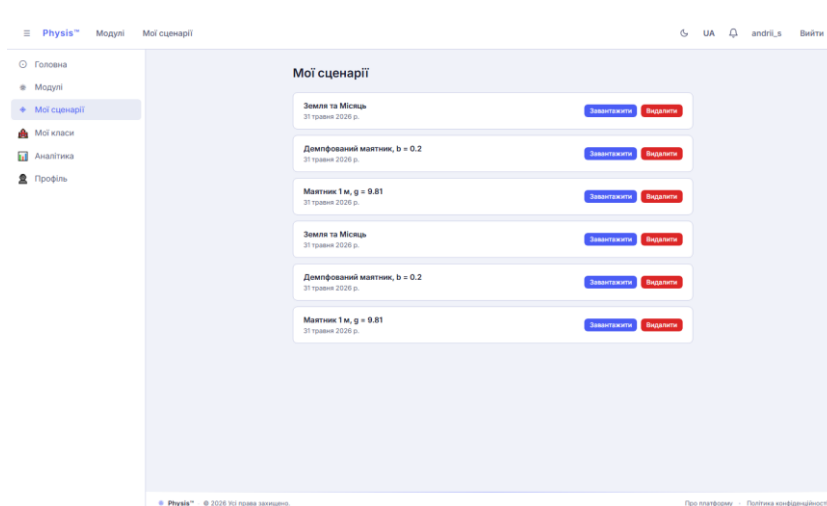


Рис. 3.20. Сторінка збережених сценаріїв

Сторінка аналітики має дві вкладки – особисту і класи. На вкладці «Особисте» учень бачить три картки зі ключовими показниками – загальна кількість здач, середній бал у відсотках і відсоток виконаних завдань, – а під ними теплову карту майстерності за шістьма категоріями фізики з градієнтом від червоного до зеленого (рис. 3.21).

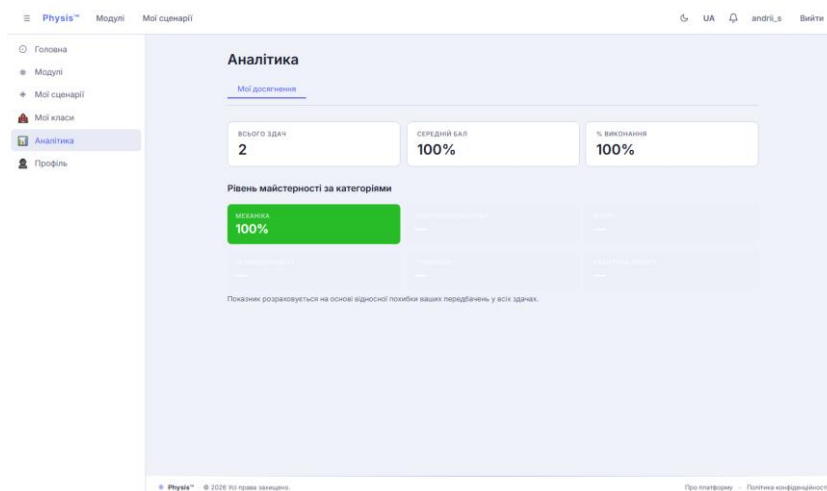


Рис. 3.21. Особиста аналітика учня з тепловою картою

На вкладці «Класи», доступній лише викладачам, у верхній частині розташовано перемикач класів у вигляді кнопок-пігулок, нижче – стовпчастий графік середніх балів за завданнями цього класу і таблиця учнів з їхньою успішністю (рис. 3.22). Клік по учневі підвантажує лінійний графік динаміки його оцінок у часі та персональну теплову карту майстерності.

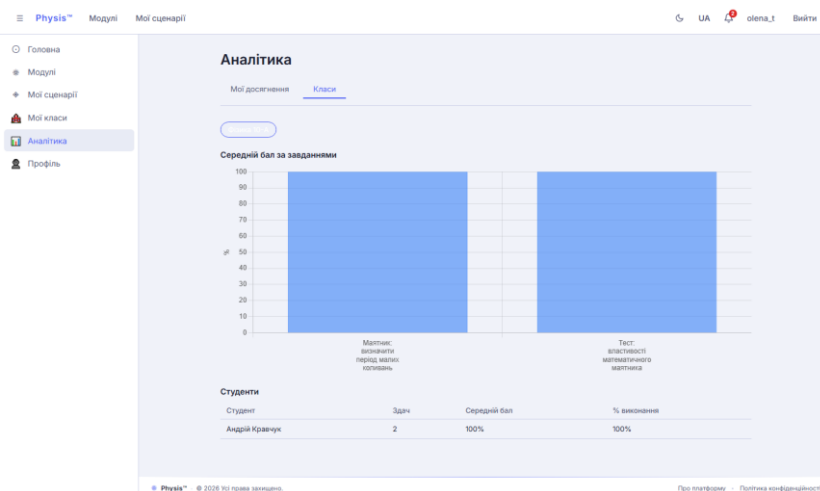


Рис. 3.22. Аналітика викладача зі статистикою класу

Сторінка профілю надає інструменти редагування особистих даних та керування акаунтом (рис. 3.23). Користувач завантажує аватар у форматах JPEG, PNG або WebP розміром до 5 МБ, редагує ім'я, прізвище, заклад освіти, факультет, курс і коротку біографію з обмеженням у 500 символів. Окремо виділено секцію роботи з ролями – поточна роль і кнопка подання заявки на її зміну. Внизу сторінки розміщено блок GDPR з двома діями: «Експортувати дані» формує JSON-файл з усією інформацією користувача, а «Видалити акаунт» запускає процедуру безповоротного видалення з підтвердженням у вигляді inline-діалогу.

Мій профіль

Андрій Кравчук
student@physis.local
Житомирська гімназія №14
Учасник з: 31.05.2020

Редигувати профіль

Ім'я: Андрій Прізвище: Кравчук
Навчальний заклад (ІНЗ): Житомирська гімназія №14 Факультет: 10-A
Про себе: Цікавлюся механікою та астрофізикою, готуюся до ЗНО.
Ім'я користувача: andrii_s Електронна пошта: student@physis.local

Роль: Учень
Подати заявку на зміну ролі

Дані та конфіденційність
Завантажте копію своїх даних або налаштуйте конфіденційність.
Завантажити мої дані Видалити акаунт

Рис. 3.23. Сторінка профілю користувача

Адміністративна частина інтерфейсу доступна лише користувачам з відповідними правами і об'єднана у спільний layout зі своєю навігацією – список користувачів, заявки на роль і журнал дій. Сторінка користувачів

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

містить таблицю з пагінацією по 15 записів, угорі – стат-картки з загальною кількістю користувачів, активних акаунтів і адміністраторів, нижче – три фільтри: текстовий пошук, фільтр за роллю та за статусом (рис. 3.24). Для кожного запису доступні дії «Деактивувати», «Активувати» і «Видалити» з inline-підтвердженням.

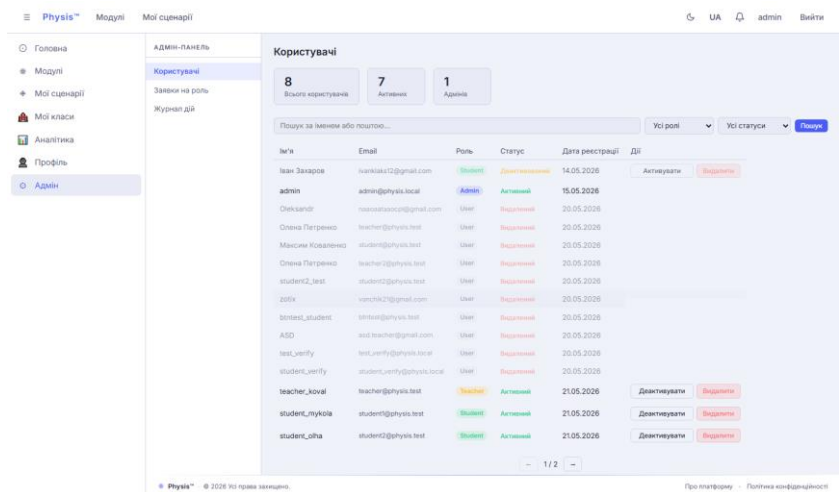


Рис. 3.24. Адмін-панель: керування користувачами

Окрема сторінка обробляє заявки на зміну ролі (рис. 3.25). Адміністратор бачить таблицю активних заявок з кнопками «Переглянути» (відкриває модальне вікно з повними даними та підтвердженням з опцією змінити цільову роль) і «Відхилити»; кожна дія фіксується у журналі.

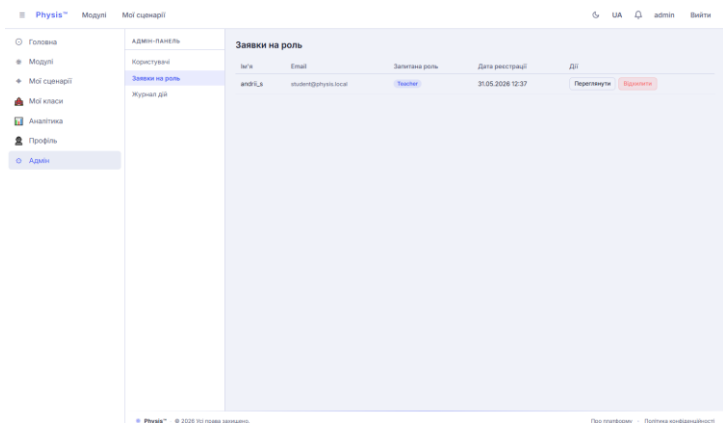


Рис. 3.25. Адмін-панель: заявки на зміну ролі

Журнал адміністративних дій надає повну історію втручань у систему (рис. 3.26). Таблиця містить п'ять колонок – адміністратор, тип дії з кольоровим бейджем, цільовий користувач, додатковий контекст у форматі

JSON і час дії – а зверху доступний фільтр за типом дії, що миттєво звужує вибірку.

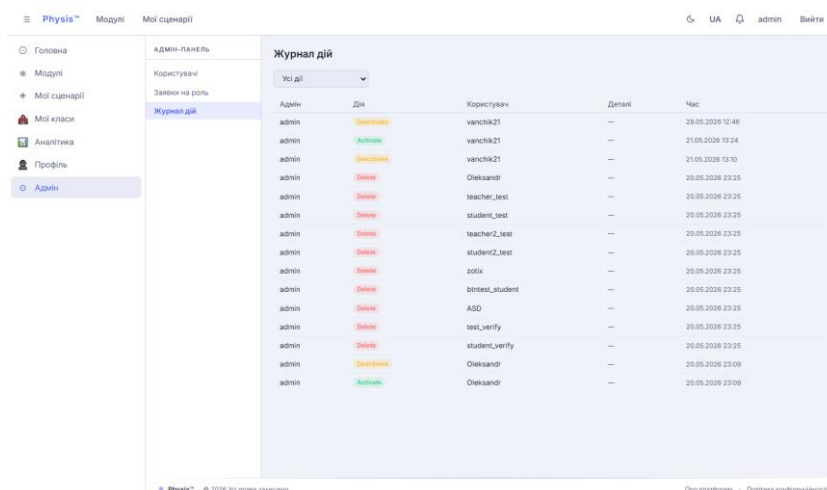


Рис. 3.26. Адмін-панель: журнал дій

Описаний інтерфейс охоплює всі ключові сценарії використання платформи з боку трьох типів користувачів – учня, викладача і адміністратора – і дотримується принципів послідовності й мінімалізму: ті самі візуальні шаблони повторюються на різних сторінках, що зменшує когнітивне навантаження і пришвидшує освоєння системи новим користувачем.

3.3. Тестування платформи

Тестування розробленої платформи проводилося безпосередньо на production-розгортанні з реальною базою даних і реальними мережевими затримками, що відображає умови експлуатації застосунку кінцевими користувачами. Свідомо було обрано підхід інтеграційного й end-to-end тестування замість модульного: для діапазону задач, які виконує платформа, перевірка коректності окремих C# та TypeScript-методів у відриві від їхнього контексту не дає реальної гарантії того, що система працює як єдине ціле. Натомість кожен з описаних нижче сценаріїв проходить наскрізь – через клієнтський фронтенд, мережу Vercel-rewrite, серверний контейнер на Railway та базу Neon, – тож позитивний результат водночас підтверджує і коректність окремої логіки, і працездатність розгортання.

Перевірка платформи розділена на чотири блоки. Перший блок присвячено коректності фізичних симуляцій: для кожного з дванадцяти модулів вибрано величину, для якої існує аналітична формула або фундаментальний закон збереження, після чого виміряне у симуляції значення зіставлено з теоретичним. Другий блок верифікує алгоритми оцінювання GradingService на десяти крайових випадках, які охоплюють усі чотири типи завдань і граничні ситуації – точне попадання, повну помилку, нульове очікуване значення, кілька метрик одночасно. Третій блок оцінює продуктивність застосунку у production-середовищі за допомогою аудиту Google Lighthouse, замірів частоти кадрів на найскладнішому фізичному модулі та таймінгу запитів до REST API. Четвертий блок підтверджує сумісність з основними браузерними рушіями шляхом виконання сценаріїв smoke-тестування у трьох незалежних engine.

Автоматизація переважної більшості тестів реалізована через бібліотеку Playwright: один скрипт послідовно відкриває кожен модуль на production-доміні `diplom-frontend-five.vercel.app`, дочікується усталення симуляції, зчитує значення метрик з DOM і порівнює їх з теоретичними. Тестування GradingService виконується через прямі виклики API за схемою «створити викладача → створити учня → створити клас → створити завдання → надіслати задачу → перевірити повернутий бал». Це дозволяє покрити сценарії, які з UI вимагали б десятків кліків.

Першим напрямком тестування стала перевірка коректності фізичних симуляцій, у межах якої усі дванадцять модулів платформи запускалися автоматизовано з параметрами за замовчуванням, після чого протягом фіксованого вікна спостереження зчитувалися експоновані ними метрики. Для нехаотичних систем виміряне значення зіставлялося з аналітичною формулою; для хаотичних (подвійний маятник, задача трьох тіл) – перевірялося збереження повної механічної енергії впродовж проміжку спостереження; для дифузії газу – монотонне зростання індексу змішування відповідно до другого

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

закону термодинаміки; для конструктора атома – закони збереження заряду й масового числа. Результати наведено у таблиці 3.2.

Таблиця 3.2

Порівняння вимірних значень з аналітичними формулами

Модуль	Перевіряється	Теоретичне	Виміряне	Похибка
Маятник (жорстке тіло)	$T = 2\pi \cdot \sqrt{L/g}$	2.006 с	2.006 с	0.003%
Подвійний маятник	Збереження Е повної	20.268 Дж сталий	20.268 Дж	< 0.001%
Пружне зіткнення (2D)	Збереження р та Е _к	p = 8 кг·м/с, Е _к = 16 Дж	p = 8, Е _к = 16	0% / 0%
Заряджена частинка (Лоренц)	$r = v/(B \cdot q/m)$, $T = 2\pi/(B \cdot q/m)$	r = 20 м, T = 6.283 с	r = 20, T = 6.283	0% / 0.003%
RC-коло	$\tau = R \cdot C$	1.000 с	1.000 с	< 0.001%
Стояча хвиля на струні	$v = \sqrt{T/\mu}$, $\lambda = 2L/n$, $f = v/\lambda$	v = 63.246, $\lambda = 4$, f = 15.811	63.246 / 4 / 15.811	0.001% / 0% / 0.002%
Інтерференція хвиль	$N_{\max} \approx 2 \cdot [d/\lambda] + 1$	5	5	0%
Ідеальний газ	$v_{\text{rms}} = \sqrt{2T}$	2.828	2.828	0.015%
Дифузія двох газів	Зростання індексу змішування	mixing → 1	mixing = 0.84	–
Орбіта планети	$T^2 = 4\pi^2 \cdot a^3/(G \cdot M)$	T = 6.283 с, e = 0	6.283 / 0	0.003% / 0%
Задача трьох тіл	Відхилення $\Delta E \leq 5\%$	$\Delta E = 0\%$	$\Delta E = 0.001\%$	< 0.001%
Конструктор атома	$q = Z - n_e$, $A = Z + N$	0 = 0 – 0	0 = 0 – 0	0%

Усі дванадцять модулів пройшли перевірку. Для модулів з прямою аналітичною формулою максимальна спостережена похибка склала 0.015% (ідеальний газ), що відповідає очікуваному рівню точності інтеграторів з фіксованим кроком $dt = 1\text{--}16$ мс. Для маятника, RC-кола, циклотронного руху та орбіти планети похибка не перевищила 0.003% – це означає, що використані напівнеявна схема Ейлера та інтегратор Рунге–Кутти на практиці зберігають

точність на рівні 4–5 значущих цифр. Для задачі трьох тіл відносний дрейф повної енергії за 5 секунд інтегрування становить 0.001%, що підтверджує симплектичні властивості реалізованого інтегратора. Для модуля дифузії газу за 8 секунд індекс змішування зріс з 0 до 0.84, що відповідає прогнозованому перебігу до термодинамічної рівноваги.

Окремий приклад наочної перевірки коректності розрахунку метрик показано на скріншоті результату оцінювання задачі (рис. 3.27) – для маятникового завдання з очікуваним періодом 2.006 с виміряний модулем період відрізняється від теоретичного на одиниці третьої значущої цифри, що повністю узгоджується з табличним результатом.

Другим напрямком тестування стала верифікація алгоритмів автоматичного оцінювання задач, реалізованих у класі GradingService, для чого розроблено набір з десяти крайових тестових випадків, що охоплюють усі чотири типи завдань – РОЕ, тест з варіантами, числова задача та сценарій з ручною перевіркою. Кожен тест виконує повний цикл «створити завдання → надіслати задачу від імені учня → отримати оцінку від сервера» проти production-середовища, після чого фактичний бал порівнюється з очікуваним аналітично. Результати наведено в таблиці 3.3.

Таблиця 3.3

Крайові випадки роботи GradingService

Код	Вхідні умови	Очікуваний бал	Фактичний бал	Статус
РОЕ-1	Точне передбачення метрики	1.000	1.000	+
РОЕ-2	Похибка одної метрики 50%	0.500	0.500	+
РОЕ-3	Похибка одної метрики $\geq 100\%$	0.000	0.000	+
РОЕ-4	Очікуване ≈ 0 , виміряне ≈ 0 (особливий випадок)	1.000	1.000	+
РОЕ-5	Дві метрики, середнє (1.0 і 0.5)	0.750	0.750	+
QUIZ-1	Тест 0 правильних з 3	0.000	0.000	+
QUIZ-2	Тест 2 правильних з 3	0.667	0.667	+
QUIZ-3	Тест 3 правильних з 3	1.000	1.000	+
PROB-1	Числова задача в межах допуску	1.000	1.000	+

PROB-2	Числова задача за межами допуску	0.000	0.000	+
--------	----------------------------------	-------	-------	---

Усі десять випадків пройшли з нульовою розбіжністю між очікуваним і фактичним балом. Зокрема, особливий випадок POE-4 (очікуване значення метрики дорівнює нулю) перевіряє коректність обробки порогу 10^{-12} , який зашитий у код GradingService для уникнення ділення на нуль і повертає 1.0 за умови, що виміряне значення також близьке до нуля. Випадок POE-5 з двома метриками (одна точно передбачена, друга з помилкою 50%) підтверджує, що підсумкова оцінка обчислюється як середнє арифметичне частинних балів – $(1.0 + 0.5) / 2 = 0.75$. Quiz-3 з усіма правильними відповідями повертає 1.0, Quiz-1 з усіма неправильними – 0.0, Quiz-2 з двома з трьох – $2/3 \approx 0.667$ без проміжного округлення. Числові задачі (PROB-1, PROB-2) коректно реагують на параметр абсолютного допуску: відповідь 1.05 при правильному значенні 1.0 та допуску 0.1 зараховується (різниця $0.05 \leq 0.1$), відповідь 1.5 – не зараховується.

Візуальне підтвердження роботи алгоритму показано на рисунку 3.27 – сторінка перевірки задач викладачем для POE-завдання з ненульовою похибкою. Платформа відображає метрика-по-метриці порівняння очікуваних і фактичних значень з обчисленою відносною похибкою для кожної з них, а підсумкова автооцінка обчислюється як середнє арифметичне частинних балів і виводиться у бейджі з кольоровим кодуванням.

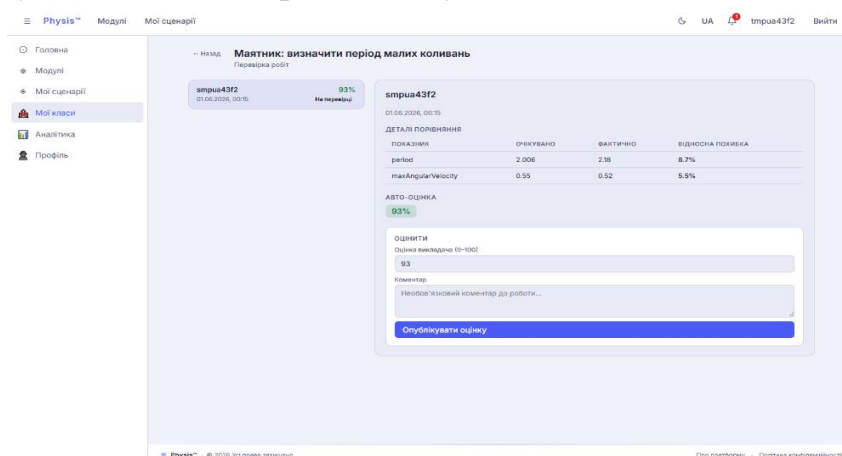


Рис. 3.27. Робота GradingService на POE-здачі з частковою похибкою

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		80

Третім і четвертим напрямком тестування стали відповідно оцінювання продуктивності клієнтської частини й перевірка кросбраузерної сумісності розгорнутого застосунку. Для виміру продуктивності використано офіційний інструмент Google Lighthouse у режимі desktop, який обчислює час до першого змістовного відображення, час до інтерактивності, сумарне блокування основного потоку та низку інших ключових показників якості клієнтського застосунку. Тестування проведено проти production-домену `diplom-frontend-five.vercel.app`. Отримані оцінки за чотирма основними категоріями показано на рисунку 3.28.

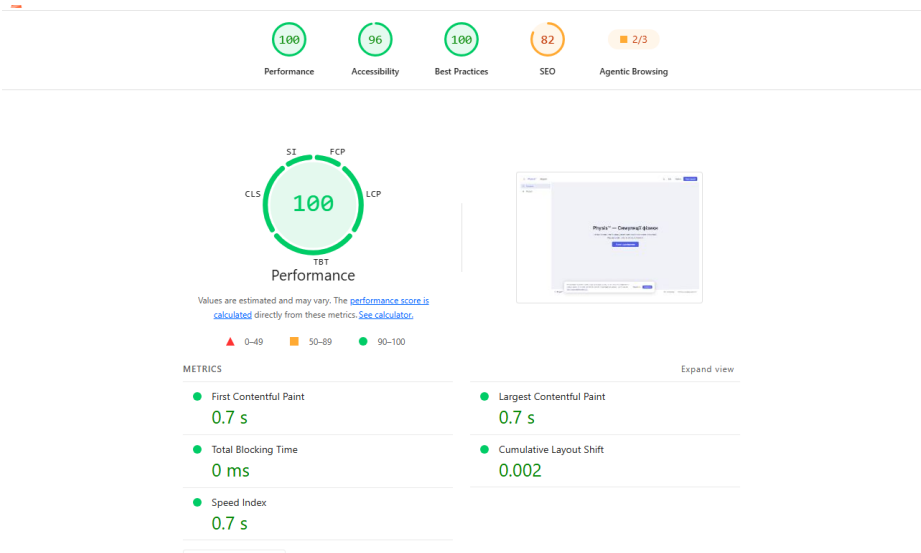
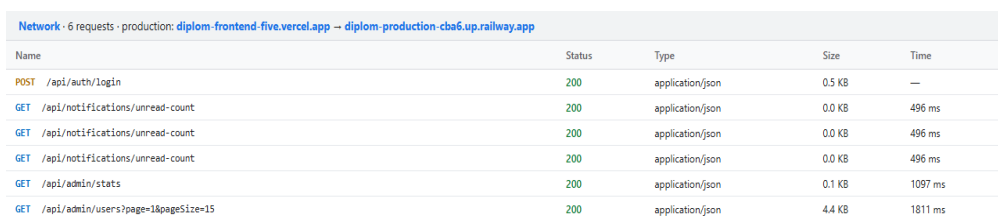


Рис. 3.28. Підсумкові оцінки Lighthouse для production-розгортання

Платформа отримала максимальну оцінку 100/100 за категорією Performance, що свідчить про відповідність найвищим вимогам до швидкості завантаження для веб-застосунків з тривимірною графікою. Перше змістовне відображення (FCP) відбулося за 0.7 с, найбільший елемент сторінки (LCP) – також за 0.7 с при рекомендованому Google порозі у 2.5 с. Кумулятивне зміщення макета (CLS) склало 0.002, що значно нижче порогу 0.1; сумарне блокування основного потоку (TBT) дорівнювало 0 мс. Категорії Best Practices та Accessibility отримали 100 і 96 балів відповідно, що підтверджує дотримання сучасних стандартів веб-розробки та доступності для людей з обмеженими можливостями.

Окремо виміряно частоту кадрів безпосередньо у браузері під час роботи найважливого з модулів – задачі трьох тіл, де одночасно інтегруються рівняння руху трьох гравітаційно взаємодіючих тіл і малюються три траєкторії-сліди. За 10-секундне вікно спостереження зібрано понад 600 семплів інтервалів між кадрами; середня частота склала 60.2 fps, мінімальна – 44 fps, 5-й перцентиль – 55 fps. Це означає, що на середньостатистичному пристрої симуляція рухається плавно й не падає нижче порогу візуальної комфортності у 30 fps навіть у найгірших випадках. Для модуля жорсткого маятника, що є типовим за обчислювальним навантаженням, середня частота також становить 60.2 fps з мінімумом 44 fps, що підтверджує – навіть у складніших модулях рендеринг не стає вузьким місцем.

Швидкість відповіді серверного API також виміряна з production-середовища. На скріншоті панелі мережевих запитів (рис. 3.29) показано вибірку реальних HTTP-запитів, які виконує клієнтський застосунок під час типового користувацького сеансу – автентифікація, перевірка нових сповіщень, отримання списку користувачів адмін-панелі. Час відповіді коливається в межах 500–1800 мс і включає у себе подвійний географічний перехід Європа–США–Європа, оскільки контейнер Railway розгорнуто у регіоні US West. Для production-демонстрації дипломної роботи такий рівень затримки прийнятний, а для реального впровадження він може бути зменшений у кілька разів простою зміною регіону Railway на найближчий до користувачів.



Name	Status	Type	Size	Time
POST /api/auth/login	200	application/json	0.5 KB	—
GET /api/notifications/unread-count	200	application/json	0.0 KB	496 ms
GET /api/notifications/unread-count	200	application/json	0.0 KB	496 ms
GET /api/notifications/unread-count	200	application/json	0.0 KB	496 ms
GET /api/admin/stats	200	application/json	0.1 KB	1097 ms
GET /api/admin/users?page=1&pageSize=15	200	application/json	4.4 KB	1811 ms

Рис. 3.29. Реальні HTTP-запити до API під час сеансу користувача

Для перевірки сумісності з різними браузерними рушіями було розроблено smoke-тест, що послідовно проходить п'ять ключових сценаріїв – завантаження лендингу, відкриття каталогу модулів, рендеринг симуляції з

ініціалізацією Babylon.js-канвасу, автентифікація адміністратора та перехід до адмін-панелі.

Таблиця 3.4

Результати кросбраузерного тестування

Рушій	Браузер	Лендинг	Каталог	Симуляція	Логін	Адмін-панель
Chromium 141	Chrome, Edge	+	+	60 fps	+	+
WebKit 26.4	Safari 17+	+	+	60 fps	+	+
Gecko 131	Firefox	+	+	60 fps	+	+

Усі п'ять сценаріїв успішно пройшли в усіх трьох браузерних рушіях, що покривають практично весь сучасний ринок десктопних і мобільних браузерів. Зокрема, рендеринг 3D-канвасу через Babylon.js працює як на WebGL-стеці Chrome/Edge, так і на власній реалізації WebKit для Safari, а Gecko-рушій Firefox теж не виявив жодних регресій. Завдяки тому, що в кодовій базі не використовуються пропрієтарні API окремих браузерів, а сам Angular і Babylon.js підтримують повний набір вебстандартів, додаткова робота з адаптацією під різні браузери виявилася непотрібною.

Підсумовуючи результати чотирьох блоків тестування – фізичні симуляції коректно відтворюють поведінку реальних систем з похибкою не більше сотих часток відсотка, алгоритми оцінювання працюють згідно з документованими формулами на всіх крайових випадках, продуктивність відповідає максимальному рівню за метриками Google Lighthouse, а сумісність охоплює усі сучасні браузерні рушії. Це підтверджує готовність платформи до використання у навчальному процесі і завершує цикл верифікації, описаний у третьому розділі.

Висновки до третього розділу

В даному розділі була наведена детальна схема розгортання платформи моделювання фізичних процесів. Кожен структурний елемент системи був

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

описаний, а також було продемонстровано його взаємодію з іншими частинами платформи. Це дозволило зрозуміти архітектуру системи та взаємозв'язки між її компонентами.

Далі було надано опис інтерфейсу розробленого програмного продукту. Були розглянуті всі функціональні можливості додатку, залежно від ролі користувача. Кожен тип користувача отримав свої унікальні функції та можливості в системі. Це допомагає кожному користувачу зорієнтуватись у функціоналі додатку та використовувати його відповідно до своїх потреб.

Окрім того, було проведено тестування реалізованої платформи у production-середовищі. Була перевірена коректність роботи усіх фізичних модулів, верифіковано алгоритми автоматичного оцінювання здач, виміряно продуктивність клієнтської частини та підтверджено сумісність з основними браузерними рушіями. Це дозволило переконатися у точності обчислень, надійності системи оцінювання та готовності платформи до використання у навчальному процесі на пристроях різних типів і в різних браузерах.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		84

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра було спроектовано та реалізовано вебплатформу для моделювання складних фізичних процесів у браузері з реалізацією методики «Передбачай – Спостерігай – Пояснюй». Проведено аналіз чотирьох провідних аналогів – PhET, Labster, Algodoo та Brilliant, – на основі якого визначено основні вимоги до власної розробки й обрано технологічний стек: ASP.NET Core 9, Angular 18, рушій Babylon.js та PostgreSQL 16.

Проектування платформи включало аналіз варіантів використання для п'яти ролей користувачів і побудову об'єктної моделі з дев'яти доменних сутностей та десяти сервісів. Спроектовано базу даних за підходом Code First з Entity Framework Core й описано основний функціонал: реєстрацію з вибором ролі, проходження трьохстадійного РОЕ-циклу, створення віртуальних класів, формування завдань чотирьох типів та автоматичне оцінювання здач.

Наведено схему розгортання у production-середовищі на трьох незалежних провайдерах – Vercel, Railway та Neon, – описано інтерфейс для різних типів користувачів і роботу аналітики у вигляді теплової карти майстерності та графіків успішності класу. Архітектуру побудовано на принципі винесення усіх фізичних обчислень на сторону клієнта через рушій Babylon.js, що дозволяє горизонтально масштабувати серверну частину; безпеку сесій реалізовано через пару JWT-токенів з механізмом TokenVersion для миттєвої інвалідації при зміні ролі або деактивації акаунту.

Вебплатформа має інтуїтивний інтерфейс з підтримкою світлої й темної тем, двох мов та дотриманням вимог GDPR. Учні запускають симуляції одним кліком, проходять РОЕ-цикл і одразу бачать порівняння прогнозів з реальністю, а викладачі створюють класи, формують завдання та оцінюють роботи. Це робить платформу придатною для впровадження у навчальний процес закладів загальної середньої та вищої освіти.

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		85

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. White R., Gunstone R. Probing Understanding. London : Falmer Press, 1992. 232 p.
2. Freeman A. Pro ASP.NET Core 9 : Develop Cloud-Ready Web Applications Using Controllers, Views, and EF Core. New York : Apress, 2025. 1132 p.
3. Bampakos A., Deeleman P. Learning Angular : A no-nonsense guide to building real-world apps with Angular. 5th ed. Birmingham : Packt Publishing, 2023. 430 p.
4. Moreau-Mathis J. Babylon.js Essentials. Birmingham : Packt Publishing, 2016. 196 p.
5. Obe R. O., Hsu L. S. PostgreSQL : Up and Running : A Practical Guide to the Advanced Open Source Database. 4th ed. Sebastopol : O'Reilly Media, 2025. 314 p.
6. Smith J. P. Entity Framework Core in Action. 2nd ed. Shelter Island : Manning Publications, 2021. 624 p.
7. Wieman C. E., Adams W. K., Perkins K. K. PhET: Simulations That Enhance Learning. Science. 2008. Vol. 322, No 5902. P. 682–683. DOI: <https://doi.org/10.1126/science.1161948>.
8. Perkins K., Adams W., Dubson M., Finkelstein N., Reid S., Wieman C., LeMaster R. PhET: Interactive Simulations for Teaching and Learning Physics. The Physics Teacher. 2006. Vol. 44, No 1. P. 18–23. DOI: <https://doi.org/10.1119/1.2150754>.
9. Adams W. K. Student engagement and learning with PhET interactive simulations. Il Nuovo Cimento C. 2010. Vol. 33, No 3. P. 21–32. DOI: <https://doi.org/10.1393/ncc/i2010-10623-0>.
10. Coştu B., Ayas A., Niaz M. Investigating the effectiveness of a POE-based teaching activity on students' understanding of condensation. Research in

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

Science Education. 2012. Vol. 42, No 6. P. 1135–1160. DOI: <https://doi.org/10.1007/s11165-011-9229-2>.

11. Liew C.-W., Treagust D. F. A predict-observe-explain teaching sequence for learning about understanding of heat and expansion of liquids. Australian Science Teachers' Journal. 1995. Vol. 41, No 1. P. 68–71.

12. Bonde M. T., Makransky G., Wandall J., Larsen M. V., Morsing M., Jarmer H., Sommer M. O. A. Improving biotech education through gamified laboratory simulations. Nature Biotechnology. 2014. Vol. 32, No 7. P. 694–697. DOI: <https://doi.org/10.1038/nbt.2955>.

13. de Jong T., Linn M. C., Zacharia Z. C. Physical and virtual laboratories in science and engineering education. Science. 2013. Vol. 340, No 6130. P. 305–308. DOI: <https://doi.org/10.1126/science.1230579>.

14. Tao P.-K., Gunstone R. F. The process of conceptual change in force and motion during computer-supported physics instruction. Journal of Research in Science Teaching. 1999. Vol. 36, No 7. P. 859–882.

15. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT) : RFC 7519. Internet Engineering Task Force, 2015. URL: <https://datatracker.ietf.org/doc/html/rfc7519> (дата звернення: 01.06.2026).

16. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : doctoral dissertation. University of California, Irvine, 2000. 162 p. URL: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm> (дата звернення: 01.06.2026).

17. HTML Living Standard. WHATWG. URL: <https://html.spec.whatwg.org> (дата звернення: 01.06.2026).

18. WebGL 2.0 Specification. Khronos Group. URL: <https://registry.khronos.org/webgl/specs/latest/2.0/> (дата звернення: 01.06.2026).

19. ASP.NET Core documentation. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/aspnet/core/> (дата звернення: 01.06.2026).

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		87

20. ASP.NET Core Identity documentation. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity/> (дата звернення: 01.06.2026).
21. Entity Framework Core documentation. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/ef/core/> (дата звернення: 01.06.2026).
22. Angular Documentation. URL: <https://angular.dev/overview> (дата звернення: 01.06.2026).
23. Babylon.js Documentation. URL: <https://doc.babylonjs.com> (дата звернення: 01.06.2026).
24. PostgreSQL 16 Documentation. PostgreSQL Global Development Group. URL: <https://www.postgresql.org/docs/16/index.html> (дата звернення: 01.06.2026).
25. Npgsql Documentation : .NET Access to PostgreSQL. URL: <https://www.npgsql.org/doc/index.html> (дата звернення: 01.06.2026).
26. PhET Interactive Simulations. University of Colorado Boulder. URL: <https://phet.colorado.edu> (дата звернення: 01.06.2026).
27. Labster Virtual Labs. URL: <https://www.labster.com> (дата звернення: 01.06.2026).
28. Algodoo. URL: <https://www.algodoo.com> (дата звернення: 01.06.2026).
29. Brilliant : Learn interactively. URL: <https://brilliant.org> (дата звернення: 01.06.2026).
30. Vercel Documentation. URL: <https://vercel.com/docs> (дата звернення: 01.06.2026).
31. Railway Documentation. URL: <https://docs.railway.app> (дата звернення: 01.06.2026).
32. Neon Documentation : Serverless Postgres. URL: <https://neon.tech/docs/introduction> (дата звернення: 01.06.2026).

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				88
Змн.	Арк.	№ докум.	Підпис	Дата		

33. Wiggins A. The Twelve-Factor App : A methodology for building software-as-a-service apps. URL: <https://12factor.net> (дата звернення: 01.06.2026).

34. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 (General Data Protection Regulation). Official Journal of the European Union. 2016. L 119. P. 1–88. URL: <https://eur-lex.europa.eu/eli/reg/2016/679/oj> (дата звернення: 01.06.2026).

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		89

ДОДАТКИ

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		90

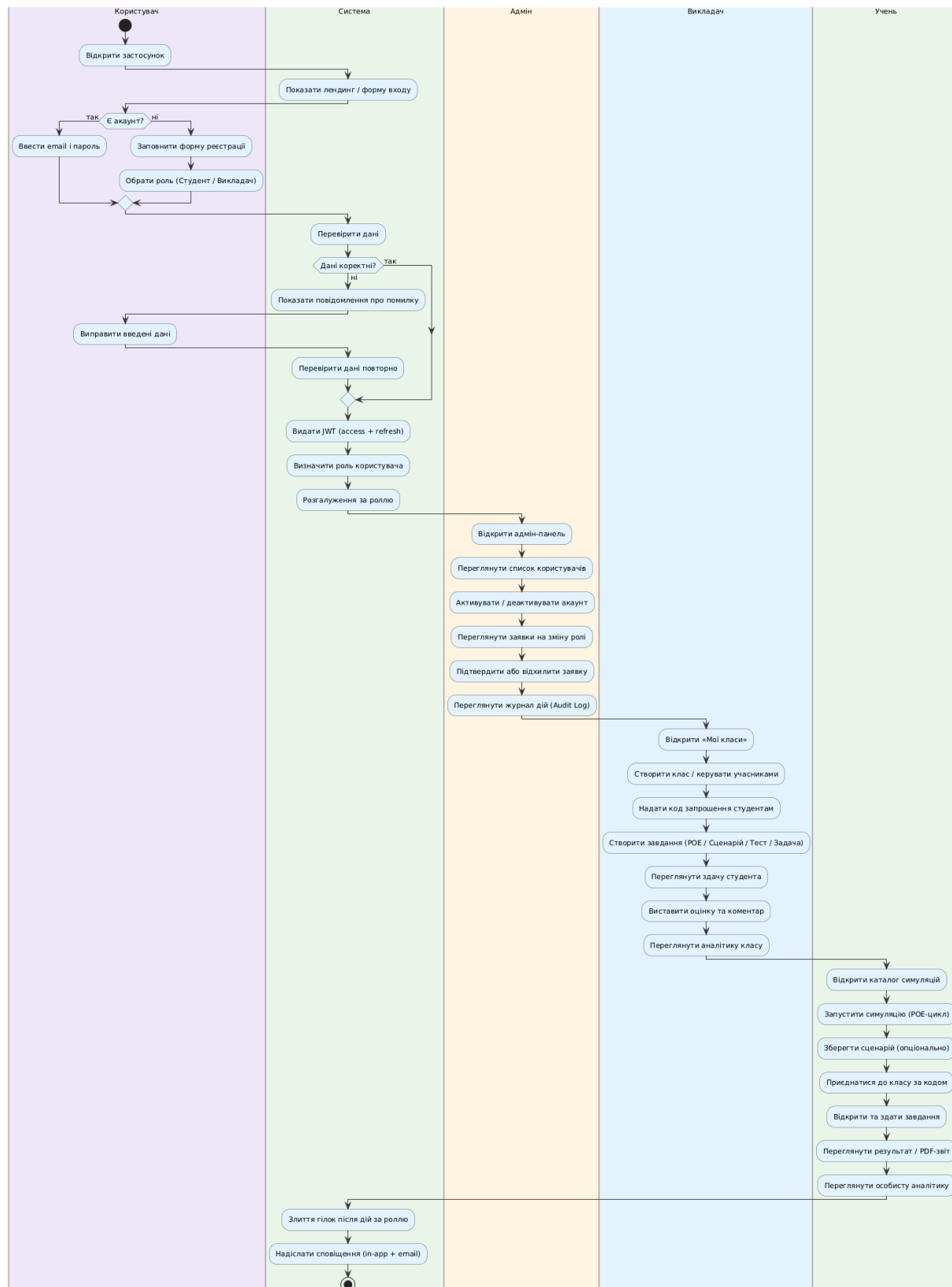


Рис. А.1 – Загальна діаграма активності системи

РОЕ-цикл (Predict - Observe - Explain)

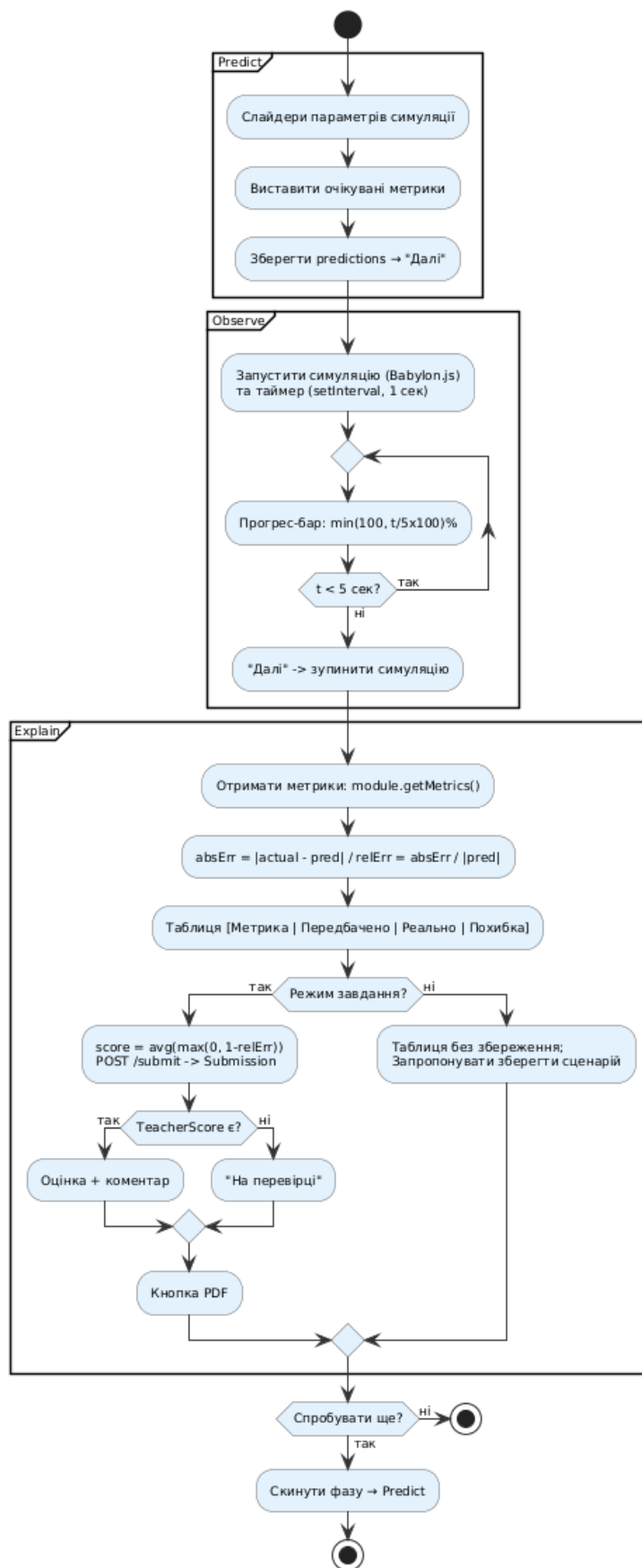


Рис. А.2 – Діаграма активності РОЕ-циклу

GradingService.cs

```

using System.Text.Json;
using Physis.Api.DTOS;

namespace Physis.Api.Services;

public static class GradingService
{
    private static readonly JsonSerializerOptions JsonOpts =
        new() { PropertyNameCaseInsensitive = true };

    public static (double Score, List<ComparisonRowDto> Rows) Grade(
        Dictionary<string, double> expected,
        Dictionary<string, double> observed)
    {
        if (expected.Count == 0)
            return (1.0, []);

        var rows = new List<ComparisonRowDto>();

        foreach (var (key, exp) in expected)
        {
            if (!observed.TryGetValue(key, out var actual))
                continue;

            var absError = Math.Abs(actual - exp);
            var relError = Math.Abs(exp) < 1e-12
                ? (Math.Abs(actual) < 1e-12 ? 0.0 : 1.0)
                : absError / Math.Abs(exp);

            rows.Add(new ComparisonRowDto(key, exp, actual, absError, relError));
        }

        if (rows.Count == 0)
            return (1.0, rows);

        var score = rows.Average(r => Math.Max(0.0, 1.0 - r.RelError));
        return (score, rows);
    }

    public static (double Score, List<ComparisonRowDto> Rows) GradeProblem(
        List<AnswerFieldDto> fields,
        Dictionary<string, double> studentAnswers)
    {
        if (fields.Count == 0)
            return (1.0, []);

        var rows = new List<ComparisonRowDto>();
    }

```

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		93

```

        int correct = 0;

        foreach (var field in fields)
        {
            if (!studentAnswers.TryGetValue(field.Label, out var actual))
                continue;

            var absError = Math.Abs(actual - field CorrectValue);
            var relError = Math.Abs(field CorrectValue) < 1e-12
                ? (Math.Abs(actual) < 1e-12 ? 0.0 : 1.0)
                : absError / Math.Abs(field CorrectValue);

            rows.Add(new ComparisonRowDto(field.Label, field CorrectValue,
            actual, absError, relError));

            if (absError <= field.Tolerance)
                correct++;
        }

        var score = rows.Count > 0 ? (double)correct / fields.Count : 1.0;
        return (score, rows);
    }

    public static List<AnswerFieldDto>? DeserializeAnswerFields(string? json)
    {
        if (string.IsNullOrEmpty(json)) return null;
        return JsonSerializer.Deserialize<List<AnswerFieldDto>>(json, JsonOpts);
    }
}

```

Poe-workflow.component.ts

```

import {
    ChangeDetectionStrategy,
    Component,
    OnDestroy,
    computed,
    effect,
    inject,
    input,
    output,
    signal,
} from '@angular/core';
import { DecimalPipe } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { DomSanitizer, type SafeHtml } from '@angular/platform-browser';
import { TranslateModule, TranslateService } from '@ngx-translate/core';
import type { PhysicsModule, PredictionTarget, SliderConfig } from '@physis/sdk';
import katex from 'katex';
import { LocalizedPipe } from '../core/pipes/localized.pipe';

```

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		94

```

type PoePhase = 'predict' | 'observe' | 'explain';

interface ComparisonRow {
  target: PredictionTarget;
  predicted: number;
  actual: number;
  absoluteError: number;
  relativeError: number | null;
  predictedPos: number | null;
  actualPos: number | null;
}

const MIN_OBSERVE_SECONDS = 5;

@Component({
  selector: 'app-poe-workflow',
  standalone: true,
  changeDetection: ChangeDetectionStrategy.OnPush,
  imports: [DecimalPipe, FormsModule, TranslateModule, LocalizedPipe],
  templateUrl: './poe-workflow.component.html',
  styleUrls: ['./poe-workflow.component.scss'],
})
export class PoeWorkflowComponent implements OnDestroy {
  readonly module = input.required<PhysicsModule>();
  readonly initialPredictions = input<Record<string, number>>({});

  readonly predictionsChange = output<Record<string, number>>();
  readonly explained          = output<Record<string, number>>();

  private readonly sanitizer = inject(DomSanitizer);
  private readonly translate = inject(TranslateService);

  protected readonly phase = signal<PoePhase>('predict');
  protected readonly predictions = signal<Record<string, number>>({});
  protected readonly observedSeconds = signal(0);
  protected readonly comparisonRows = signal<ComparisonRow[]>([]);
  protected readonly explanationHtml = signal<SafeHtml>('');

  protected readonly targets = computed<PredictionTarget[]>(() =>
    this.module().getPredictionTargets(),
  );

  protected readonly canContinueObserve = computed(
    () => this.observedSeconds() >= MIN_OBSERVE_SECONDS,
  );

  protected readonly observeProgress = computed(() =>
    Math.min(100, (this.observedSeconds() / MIN_OBSERVE_SECONDS) * 100),
  );

```

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		95

```

private timer: ReturnType<typeof setInterval> | null = null;

constructor() {
  effect(() => {
    const initial = this.initialPredictions();
    if (Object.keys(initial).length > 0) {
      this.predictions.set(initial);
    }
  }, { allowSignalWrites: true });
}

protected getPrediction(key: string): number {
  return this.predictions()[key] ?? 0;
}

protected setPrediction(key: string, value: number): void {
  this.predictions.update(prev => ({ ...prev, [key]: Number(value) }));
  this.predictionsChange.emit({ ...this.predictions() });
}

protected getSliderPos(target: PredictionTarget): number {
  const slider = target.slider!;
  const val = this.predictions()[target.metricKey];
  if (val === undefined) return 50;
  return Math.round(this.toSliderPos(slider, val));
}

protected setSliderPrediction(target: PredictionTarget, pos: number | string):
void {
  const slider = target.slider!;
  const val = slider.min + (Number(pos) / 100) * (slider.max - slider.min);
  this.setPrediction(target.metricKey, val);
}

protected toObserve(): void {
  this.observedSeconds.set(0);
  this.phase.set('observe');
  this.timer = setInterval(() => {
    this.observedSeconds.update(s => s + 1);
  }, 1000);
}

protected toExplain(): void {
  this.stopTimer();
  const metrics = this.module().getMetrics();
  const preds = this.predictions();

  const rows: ComparisonRow[] = this.targets().map(target => {
    const slider = target.slider;
    const predicted = preds[target.metricKey]
  });
}

```

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		96

```

    ?? (slider ? slider.min + 0.5 * (slider.max - slider.min) : 0);
    const actual = metrics.scalars[target.metricKey] ?? 0;
    const absoluteError = Math.abs(actual - predicted);
    const relativeError = Math.abs(actual) > 1e-9
      ? (absoluteError / Math.abs(actual)) * 100
      : null;
    const predictedPos = slider ? this.toSliderPos(slider, predicted) : null;
    const actualPos = slider ? this.toSliderPos(slider, actual) : null;
    return { target, predicted, actual, absoluteError, relativeError,
predictedPos, actualPos };
  });

  this.comparisonRows.set(rows);
  this.explanationHtml.set(this.renderExplanation(preds));
  this.phase.set('explain');
  this.explained.emit(metrics.scalars);
}

protected tryAgain(): void {
  this.stopTimer();
  this.predictions.set({});
  this.observedSeconds.set(0);
  this.module().reset();
  this.phase.set('predict');
}

private toSliderPos(slider: SliderConfig, value: number): number {
  return Math.max(0, Math.min(100, ((value - slider.min) / (slider.max -
slider.min)) * 100));
}

private renderExplanation(preds: Record<string, number>): SafeHtml {
  const raw = this.module().getExplanation(preds, this.translate.currentLang);
  const html = raw.replace(/\$([\^$]+)\$/g, (_match, latex: string) => {
    try {
      return katex.renderToString(latex, { throwOnError: false });
    } catch {
      return latex;
    }
  });
  return this.sanitizer.bypassSecurityTrustHtml(html);
}

private stopTimer(): void {
  if (this.timer !== null) {
    clearInterval(this.timer);
    this.timer = null;
  }
}

```

```

ngOnDestroy(): void {
    this.stopTimer();
}
}

```

LabReportService.cs

```

using System.Text.Json;
using Microsoft.EntityFrameworkCore;
using QuestPDF.Fluent;
using QuestPDF.Helpers;
using QuestPDF.Infrastructure;
using Physis.Api.Data;
using Physis.Api.Models;

namespace Physis.Api.Services;

public class LabReportService(AppDbContext db)
{
    private static readonly JsonSerializerOptions JsonOpts =
        new() { PropertyNameCaseInsensitive = true };

    public async Task<byte[]?> GenerateAsync(Guid submissionId, string
requestUserId)
    {
        var submission = await db.Submissions
            .Include(s => s.Student)
            .Include(s => s.Artifact)
            .Include(s => s.Assignment)
            .ThenInclude(a => a.Classroom)
            .FirstOrDefaultAsync(s => s.Id == submissionId);

        if (submission is null) return null;

        var isOwner = submission.StudentId == requestUserId;
        var isTeacher = await db.ClassroomMemberships.AnyAsync(m =>
            m.ClassroomId == submission.Assignment.ClassroomId
            && m.UserId == requestUserId
            && m.Role == ClassroomRole.Teacher);

        if (!isOwner && !isTeacher) return null;

        var gradingRows = ParseRows(submission.GradingRows);
        var moduleEntry = ModuleCatalog.Entries
            .FirstOrDefault(e => e.Id == submission.Assignment.ModuleId);
        var moduleName = moduleEntry?.NameUk ?? submission.Assignment.ModuleId;

        var studentName = $"{submission.Student.FirstName}
{submission.Student.LastName}.Trim() is
        { Length: > 0 } n ? n : submission.Student.UserName ??
submission.StudentId;

```

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				98
Змн.	Арк.	№ докум.	Підпис	Дата		

```

return Document.Create(container =>
{
    container.Page(page =>
    {
        page.Size(PageSizes.A4);
        page.Margin(2, Unit.Centimetre);
        page.DefaultTextStyle(t => t.FontSize(11).FontFamily("Arial"));

        page.Header().Column(col =>
        {
            col.Item().Row(row =>
            {
                row.RelativeItem().Text("Physis™")
                    .FontSize(22).Bold();
                row.ConstantItem(200).AlignRight()
                    .Text(submission.SubmittedAt.ToString("dd.MM.yyyy"))
                    .FontColor(Colors.Grey.Medium);
            });
            col.Item().Text("Звіт про виконання лабораторної роботи")
                .FontSize(13).FontColor(Colors.Grey.Darken2);
            col.Item().PaddingTop(4).LineHorizontal(1).LineColor(Colors.Grey.Lighten2);
        });

        page.Content().PaddingTop(16).Column(col =>
        {
            // Meta
            col.Item().Table(t =>
            {
                t.ColumnsDefinition(c => { c.RelativeColumn(3);
                c.RelativeColumn(5); });
                MetaRow(t, "Модуль", moduleName);
                MetaRow(t, "Студент", studentName);
                MetaRow(t, "Клас", submission.Assignment.Classroom.Name);
                MetaRow(t, "Завдання", submission.Assignment.Title);
            });

            // Description
            if
(!string.IsNullOrWhiteSpace(submission.Assignment.Description))
            {
                col.Item().PaddingTop(14).Text("Умова").FontSize(13).Bold();
                col.Item().PaddingTop(4).Text(submission.Assignment.Description);
            }
        });
    });
}

```

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		99

```

col.Item().PaddingTop(14).Text("Прогнози студента проти
спостережених значень")
    .FontSize(13).Bold();

if (gradingRows.Count > 0)
{
    col.Item().PaddingTop(6).Table(t =>
    {
        t.ColumnsDefinition(c =>
        {
            c.RelativeColumn(3);
            c.RelativeColumn(2);
            c.RelativeColumn(2);
            c.RelativeColumn(2);
        });

        t.Header(h =>
        {
            foreach (var label in new[] { "Метрика",
"Очікувано", "Отримано", "Відн. похибка" })
                h.Cell().Background(Colors.Grey.Lighten3)
                    .Padding(4).Text(label).Bold();
        });

        foreach (var row in gradingRows)
        {
            t.Cell().Padding(4).Text(row.Key);
            t.Cell().Padding(4).Text(row.Expected.ToString("G
4"));

            t.Cell().Padding(4).Text(row.Actual.ToString("G4"
));

            t.Cell().Padding(4)
                .Text($"{row.RelError:P1}")
                .FontColor(row.RelError < 0.1 ?
Colors.Green.Darken2 : Colors.Red.Darken1);
        }
    });
}
else
{
    col.Item().PaddingTop(4).Text("Дані оцінювання
відсутні.")
        .FontColor(Colors.Grey.Medium);
}

// Score badge
col.Item().PaddingTop(10).Row(row =>
{
    row.AutoItem().Text("Загальний бал: ").Bold();
    row.AutoItem()

```

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		100

```

        .Text($"{submission.Score * 100:F0}%")
        .Bold()
        .FontColor(submission.Score >= 0.9 ?
Colors.Green.Darken2 : Colors.Orange.Darken1);
    });

    if (submission.Artifact is { Kind: "screenshot" } art)
    {
        col.Item().PaddingTop(14).Text("Скріншот
симуляції").FontSize(13).Bold();
        col.Item().PaddingTop(6).MaxHeight(220).Image(art.Data);
    }

    // Conclusion
    col.Item().PaddingTop(14).Text("Висновок").FontSize(13).Bold(
);
    col.Item().PaddingTop(4).Background(Colors.Grey.Lighten4).Pad
ding(8)
        .Text(string.IsNullOrEmpty(submission.ConclusionText
)
        ? "Висновок не надано."
        : submission.ConclusionText);
    });

    page.Footer().AlignCenter()
        .Text(t =>
        {
            t.Span("Physis™ · ").FontColor(Colors.Grey.Medium);
            t.CurrentPageNumber();
            t.Span(" / ");
            t.TotalPages();
        });
    });
}).GeneratePdf();
}

private static void MetaRow(TableDescriptor t, string label, string value)
{
    t.Cell().Padding(3).Text(label).FontColor(Colors.Grey.Darken2);
    t.Cell().Padding(3).Text(value);
}

private static List<GradingRowData> ParseRows(string json)
{
    try { return JsonSerializer.Deserialize<List<GradingRowData>>(json,
JsonOpts) ?? []; }
    catch { return []; }
}

private record GradingRowData(

```

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		101

```

        string Key, double Expected, double Actual, double AbsError, double
RelError);
    }

```

AppDbContext.cs

```

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using Physis.Api.Models;

namespace Physis.Api.Data;

public class AppDbContext(DbContextOptions<AppDbContext> options)
    : IdentityDbContext<ApplicationUser>(options)
{
    public DbSet<SavedScenario> SavedScenarios =>
Set<SavedScenario>();
    public DbSet<Classroom> Classrooms => Set<Classroom>();
    public DbSet<ClassroomMembership> ClassroomMemberships =>
Set<ClassroomMembership>();
    public DbSet<Assignment> Assignments =>
Set<Assignment>();
    public DbSet<Submission> Submissions =>
Set<Submission>();
    public DbSet<SubmissionArtifact> SubmissionArtifacts =>
Set<SubmissionArtifact>();
    public DbSet<RoleChangeRequest> RoleChangeRequests =>
Set<RoleChangeRequest>();
    public DbSet<AdminAuditLog> AdminAuditLogs =>
Set<AdminAuditLog>();
    public DbSet<Notification> Notifications =>
Set<Notification>();

    protected override void OnModelCreating(ModelBuilder builder)
    {
        base.OnModelCreating(builder);

        builder.Entity<ApplicationUser>()
            .HasQueryFilter(u => !u.IsDeleted);

        builder.Entity<ApplicationUser>(e =>
        {
            e.Property(u =>
u.FirstName).HasMaxLength(100).HasDefaultValue(string.Empty);
            e.Property(u =>
u.LastName).HasMaxLength(100).HasDefaultValue(string.Empty);
            e.Property(u =>
u.University).HasMaxLength(200).HasDefaultValue(string.Empty);
            e.Property(u => u.Faculty).HasMaxLength(200);
            e.Property(u => u.Bio).HasMaxLength(500);
            e.Property(u => u.CreatedAt).HasDefaultValueSql("now()");
        });
    }

```

		Захаров І. А.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		102

```

        e.Property(u => u.UpdatedAt).HasDefaultValueSql("now()");
    });

    builder.Entity<SavedScenario>(e =>
    {
        e.HasKey(s => s.Id);
        e.Property(s => s.Id).ValueGeneratedOnAdd();
        e.Property(s => s.CreatedAt).HasDefaultValueSql("now()");
        e.HasOne(s => s.User)
            .WithMany()
            .HasForeignKey(s => s.UserId)
            .OnDelete(DeleteBehavior.Cascade);
    });

    builder.Entity<Classroom>(e =>
    {
        e.HasKey(c => c.Id);
        e.Property(c => c.Id).ValueGeneratedOnAdd();
        e.Property(c => c.Name).HasMaxLength(100).IsRequired();
        e.Property(c => c.Description).HasMaxLength(500);
        e.Property(c => c.InviteCode).HasMaxLength(8).IsRequired();
        e.Property(c => c.CreatedAt).HasDefaultValueSql("now()");
        e.HasIndex(c => c.InviteCode).IsUnique();
        e.HasOne(c => c.Owner)
            .WithMany()
            .HasForeignKey(c => c.OwnerId)
            .OnDelete(DeleteBehavior.Restrict);
    });

    builder.Entity<ClassroomMembership>(e =>
    {
        e.HasKey(m => m.Id);
        e.Property(m => m.Id).ValueGeneratedOnAdd();
        e.Property(m => m.JoinedAt).HasDefaultValueSql("now()");
        e.HasIndex(m => new { m.ClassroomId, m.UserId }).IsUnique();
        e.HasOne(m => m.Classroom)
            .WithMany(c => c.Memberships)
            .HasForeignKey(m => m.ClassroomId)
            .OnDelete(DeleteBehavior.Cascade);
        e.HasOne(m => m.User)
            .WithMany()
            .HasForeignKey(m => m.UserId)
            .OnDelete(DeleteBehavior.Cascade);
    });

    builder.Entity<Assignment>(e =>
    {
        e.HasKey(a => a.Id);
        e.Property(a => a.Id).ValueGeneratedOnAdd();
        e.Property(a => a.ModuleId).HasMaxLength(50).IsRequired();
    });

```

```

        e.Property(a => a.Title).HasMaxLength(200).IsRequired();
        e.Property(a => a.Description).HasMaxLength(1000);
        e.Property(a =>
a.ExpectedMetrics).HasColumnType("text").HasDefaultValue("{}");
        e.Property(a => a.AnswerFieldsJson).HasColumnType("text");
        e.Property(a => a.CreatedAt).HasDefaultValueSql("now()");
        e.HasOne(a => a.Classroom)
            .WithMany()
            .HasForeignKey(a => a.ClassroomId)
            .OnDelete(DeleteBehavior.Cascade);
        e.HasOne(a => a.CreatedBy)
            .WithMany()
            .HasForeignKey(a => a.CreatedById)
            .OnDelete(DeleteBehavior.Restrict);
    });

    builder.Entity<Submission>(e =>
    {
        e.HasKey(s => s.Id);
        e.Property(s => s.Id).ValueGeneratedOnAdd();
        e.Property(s =>
s.ObservedMetrics).HasColumnType("text").HasDefaultValue("{}");
        e.Property(s =>
s.GradingRows).HasColumnType("text").HasDefaultValue("");
        e.Property(s => s.ConclusionText).HasMaxLength(2000);
        e.Property(s => s.ProblemAnswers).HasColumnType("text");
        e.Property(s =>
s.Status).HasDefaultValue(SubmissionStatus.Submitted);
        e.Property(s => s.SubmittedAt).HasDefaultValueSql("now()");
        e.HasIndex(s => new { s.AssignmentId, s.StudentId }).IsUnique();
        e.HasOne(s => s.Assignment)
            .WithMany(a => a.Submissions)
            .HasForeignKey(s => s.AssignmentId)
            .OnDelete(DeleteBehavior.Cascade);
        e.HasOne(s => s.Student)
            .WithMany()
            .HasForeignKey(s => s.StudentId)
            .OnDelete(DeleteBehavior.Cascade);
        e.HasOne(s => s.Artifact)
            .WithOne(a => a.Submission)
            .HasForeignKey<SubmissionArtifact>(a => a.SubmissionId)
            .OnDelete(DeleteBehavior.Cascade);
    });

    builder.Entity<SubmissionArtifact>(e =>
    {
        e.HasKey(a => a.Id);
        e.Property(a => a.Id).ValueGeneratedOnAdd();
        e.Property(a => a.Kind).HasMaxLength(50).IsRequired();
        e.Property(a => a.ContentType).HasMaxLength(100).IsRequired();
    });

```

```

        e.HasKey(a => a.SubmissionId).IsUnique();
    });

    builder.Entity<RoleChangeRequest>(e =>
    {
        e.HasKey(r => r.Id);
        e.Property(r => r.Id).ValueGeneratedOnAdd();
        e.Property(r => r.RequestedRole).HasMaxLength(20).IsRequired();
        e.Property(r => r.Status).HasMaxLength(20).IsRequired();
        e.Property(r => r.CreatedAt).HasDefaultValueSql("now()");
        e.HasOne(r => r.User)
            .WithMany()
            .HasForeignKey(r => r.UserId)
            .OnDelete(DeleteBehavior.Cascade);
    });

    builder.Entity<AdminAuditLog>(e =>
    {
        e.HasKey(a => a.Id);
        e.Property(a => a.Id).ValueGeneratedOnAdd();
        e.Property(a => a.Action).HasMaxLength(50).IsRequired();
        e.Property(a => a.TargetUserId).IsRequired();
        e.Property(a => a.Details).HasMaxLength(1000);
        e.Property(a => a.CreatedAt).HasDefaultValueSql("now()");
        e.HasOne(a => a.Admin)
            .WithMany()
            .HasForeignKey(a => a.AdminId)
            .OnDelete(DeleteBehavior.Restrict);
    });

    builder.Entity<Notification>(e =>
    {
        e.HasKey(n => n.Id);
        e.Property(n => n.Id).ValueGeneratedOnAdd();
        e.Property(n => n.Message).HasMaxLength(500).IsRequired();
        e.Property(n => n.Link).HasMaxLength(200);
        e.Property(n => n.CreatedAt).HasDefaultValueSql("now()");
        e.CreateIndex(n => new { n.UserId, n.IsRead });
        e.HasOne(n => n.User)
            .WithMany()
            .HasForeignKey(n => n.UserId)
            .OnDelete(DeleteBehavior.Cascade);
    });
}
}

```