

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:

«Система відстеження помилок для ІТ проєктів»

Виконав студент групи ПЗ-21-2
ПУШКАР Кирило Вікторович

Керівник роботи:
ФАНТ Микола Олександрович

Рецензент:
ТУРЕНКО Єлизавета Єгорівна

Житомир – 2025

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

«14» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ПУШКАР Кирило Вікторович**

Керівник роботи: **ФАНТ Микола Олександрович**

Тема роботи: **«Система відстеження помилок для ІТ проєктів»**,
затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: **процес використання технологій для розробки системи відстеження помилок, що забезпечує ефективне виявлення, аналіз та моніторинг дефектів у проєктах розробки програмного забезпечення.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Фант М.О.	27.02.2024р.	27.02.2024р.
2	Фант М.О.	27.02.2024р.	27.02.2024р.
3	Фант М.О.	27.02.2024р.	27.02.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з веб-додатку системи відстеження помилок та пояснювальної записки. Пояснювальна записка до випускної роботи містить 88 сторінок, 43 ілюстрацій та 23 таблиці. Метою роботи є проектування та реалізація сучасної системи відстеження помилок, що забезпечує ефективне управління життєвим циклом дефектів програмного забезпечення, взаємодію між учасниками команди розробки та аналітику даних.

Архітектура системи відстеження помилок побудована на клієнт-серверній моделі. Функціональність системи включає реєстрацію та авторизацію користувачів, створення та управління проєктами, звітами про помилки, коментування, додавання вкладень та експорт даних у зовнішні системи.

Система розроблена з урахуванням потреб користувача, інтерфейс забезпечує максимально комфортну роботу, що дозволяє користувачам легко реєструвати помилки, відстежувати їхній прогрес, фільтрувати та сортувати дані, отримувати аналітичні звіти а також здійснювати їх експорт до сторонніх комерційних систем управління.

КЛЮЧОВІ СЛОВА: СИСТЕМА ВІДСТЕЖЕННЯ ПОМИЛОК, ВЕБДОДАТОК, NODE.JS, REACT, TYPESCRIPT, POSTGRESQL, УПРАВЛІННЯ ПРОЄКТАМИ.

					ІПЗ.КР.Б – 121 – 25– ІПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Система відстеження помилок для ІТ проєктів	Літ.	Арк.	Аркуші
Розроб.		Пушкар К.В.					3	88
Керівник		Фант М.О.						
Рецензент		Туренко Є.Є.				Житомирська політехніка. ІПЗ-21-2		
Зав. каф..		Вакалюк Т.А.						

ABSTRACT

The bachelor's thesis consists of a bug tracking system web application and an explanatory note. The explanatory note to the final thesis contains 88 pages, 43 illustrations, and 23 table. The goal of the work is to design and implement a modern bug tracking system that ensures effective management of the software defect lifecycle, interaction between development team members, and data analytics.

The architecture of the bug tracking system is built on a client-server model. The system's functionality includes user registration and authorization, creating and managing projects, bug reports, commenting, adding attachments, and exporting data to external systems.

The system is designed with user needs in mind; the interface provides maximum comfort, allowing users to easily report bugs, track their progress, filter and sort data, receive analytical reports and export them to third-party commercial management systems.

KEYWORDS: BUG TRACKING SYSTEM, WEB APPLICATION, NODE.JS, REACT, TYPESCRIPT, POSTGRESQL, PROJECT MANAGEMENT.

		Пушкар К.В.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ВСТУП.....	6
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ СИСТЕМИ ВІДСТЕЖЕННЯ ПОМИЛОК.....	8
1.1. Аналіз задачі, засобів та методів її вирішення.....	8
1.2. Аналіз аналогів програмного продукту	9
1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	13
Висновки до першого розділу.....	16
РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ВІДСТЕЖЕННЯ ПОМИЛОК	17
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	17
2.2. Розробка бази даних системи.....	24
2.3. Проектування та реалізація алгоритмів роботи системи	30
2.4. Реалізація функціоналу системи відстеження помилок.....	32
Висновки до другого розділу	36
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ВІДСТЕЖЕННЯ ПОМИЛОК.....	37
3.1. Порядок встановлення та налаштування системи	37
3.2. Структура інтерфейсу системи відстеження помилок	39
3.3. Тестування системи відстеження помилок	61
Висновки до третього розділу.....	64
ВИСНОВКИ.....	65
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	66
ДОДАТКИ.....	69

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ВСТУП

Актуальність теми обумовлена зростаючою потребою в ефективному управлінні якістю програмного забезпечення у сучасних ІТ проєктах. Системи відстеження помилок дозволяють автоматизувати процес виявлення, класифікації та усунення дефектів, що забезпечує безперервний контроль над якістю розроблюваного продукту. Завдяки інтеграції з іншими інструментами управління проєктами, ці системи сприяють оптимізації робочих процесів, покращенню комунікації між учасниками команди та зменшенню часу на вирішення критичних проблем.

Сучасні рішення у сфері відстеження помилок дозволяють не лише швидко реагувати на виникаючі дефекти, але й прогнозувати потенційні ризики, забезпечуючи високу стабільність і надійність ІТ продуктів. Інноваційні підходи до автоматизації процесів виявлення та аналізу помилок стають запорукою успішної реалізації як малих стартапів, так і масштабних корпоративних проєктів.

Метою випускної кваліфікаційної роботи бакалавра є розробка системи відстеження помилок для ІТ проєктів, що дозволить ефективно управляти процесом виявлення та усунення дефектів, забезпечуючи високий рівень стабільності та надійності програмного забезпечення.

Завдання роботи. На основі мети роботи завдання можна сформулювати таким чином:

1. Провести огляд сучасних систем відстеження помилок, аналіз їх функціональності та ефективності в різних типах ІТ проєктів, включаючи методи збору та обробки статистичних даних.
2. Здійснити вибір засобів проектування та технологічних рішень.
3. Розробити систему відстеження помилок, що забезпечує автоматизацію процесів виявлення, класифікації та усунення дефектів, а також підтримує прозорість та оперативність взаємодії між членами команди.

Об'єктом дослідження є інформаційні технології управління процесами виявлення та усунення дефектів у програмному забезпеченні.

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

Предметом дослідження є використання технологій для розробки системи відстеження помилок, що забезпечує ефективне виявлення, аналіз та моніторинг дефектів у проєктах розробки ПЗ.

Дослідження охоплює аналіз сучасних підходів до збору, обробки та зберігання інформації про помилки, а також вивчення можливостей інтеграції з іншими інструментами управління проєктами. Особлива увага приділяється ролі технологій у створенні зручної системи, що сприяє прозорості процесів та оперативному реагуванню на виявлені дефекти. Розробка системи відстеження помилок має на меті підвищення ефективності комунікації між членами команди, оптимізацію процесів тестування та забезпечення стабільності програмного забезпечення.

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ СИСТЕМИ ВІДСТЕЖЕННЯ ПОМИЛОК

1.1. Аналіз задачі, засобів та методів її вирішення

У ході аналізу було виокремлено наступні умови та функціональні вимоги, які має містити програмний продукт:

- Моделювання загальної архітектури системи. Розробка концептуальної моделі системи з урахуванням різних ролей користувачів, таких як розробники та адміністратори.
- Визначення механізму взаємодії користувачів із системою. Створення зручного та зрозумілого інтерфейсу, який дозволить користувачам створювати записи про помилки, редагувати їх деталі, додавати коментарі та відслідковувати статус вирішення дефектів.
- Функціонал управління помилками. Забезпечення можливості додавання нових записів про помилки, редагування їх характеристик, а також архівування завершених задач для подальшого аналізу.
- Пошук та фільтрація записів. Реалізація механізмів для швидкого пошуку та сортування помилок за різними критеріями (тип, пріоритет, дата створення), що значно полегшить роботу з великою кількістю даних.
- Експортування звітів до системи керування проектами Atlassian Jira[1]. Забезпечення можливості автоматичного передавання даних про помилки до комерційної платформи для подальшого планування та моніторингу.

Проведений аналіз дозволив систематизувати як статичні, так і динамічні параметри роботи системи. Робоче визначення якості системи охоплює його ефективність, зручність використання та продуктивність, що враховує як об'єктивні характеристики роботи системи, так і суб'єктивну оцінку користувачів. Це дозволить проводити узагальнення результатів тестування та оперативно впроваджувати необхідні удосконалення, забезпечуючи високу стабільність та надійність процесів розробки програмного забезпечення.

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

1.2. Аналіз аналогів програмного продукту

Успішна розробка системи відстеження помилок вимагає ретельного дослідження існуючих рішень на ринку інструментів управління проектами. Було проведено комплексне порівняння провідних систем для ефективного відстеження розробки програмного забезпечення.

Одним з таких додатків є система Bugzilla (рис. 1.1).

Bugzilla[2] – відкрите програмне забезпечення, що є повністю безкоштовним та має відкритий код, що дозволяє його гнучку модифікацію. Це перевірене часом, стабільне рішення, широко використовуване в проєктах з відкритим кодом. Система дозволяє тонке налаштування відповідно до специфічних потреб проєкту. Проте інтерфейс може здатися застарілим і неінтуїтивним для нових користувачів, кількість стандартних інтеграцій із сучасними інструментами CI/CD обмежена, а спеціалізованих рішень для роботи з мобільними пристроями не надається.

Bugzilla

Search Bugs

Browse

Advanced Search

>>

Target Milestone: Firefox 78

Classification: Client Software, Developer Infrastructure, Components, Server Software, Other, Graveyard

287 bugs found.

ID	Type	Summary	Product	Comp	Assignee ^	Status ^	Resolution	Updated
1629729		Intermittent [tier2] devtools/client/responsive/test/browser/browser_container_tab.js Test timed out -	DevTools	Responsive Design Mo	mtigley	REOP	---	2020-05-17
1634054		Document trigger and button actions for CFRs	Firefox	Messaging System	andrei.br92	RESO	FIXE	Fri 14:45
1636463		Refactor ASRouter to use SpecialMessage Actions	Firefox	Messaging System	andrei.br92	RESO	FIXE	Fri 01:58
1639177		Checkbox in What's New panel is cut out on macOS in French	Firefox	Messaging System	andrei.br92	RESO	FIXE	Thu 14:56
1636229		Popup javascript window is now tabbed (v76 32 bit)	Firefox	Theme	arai.unmht	RESO	FIXE	Fri 13:30
1635401		Polish UX: minor fixes	DevTools	Application Panel	balbeza	RESO	FIXE	2020-05-14
1630844		Collect telemetry on what causes the PDF viewer fallback bar	Firefox	PDF Viewer	bdahl	RESO	FIXE	Tue 20:42
1607630		pdf.js occasionally shows incorrect symbols until page is refreshed	Firefox	PDF Viewer	bobowencode	RESO	FIXE	Wed 05:25
1620098		Can't access all items in drop down list of device sizes	DevTools	Responsive Design Mo	daisuke	RESO	FIXE	2020-05-11
1637562		Expose an API on `ResourceWatcher` in order to only listen to "future" resources	DevTools	Framework	daisuke	RESO	FIXE	Wed 08:51

Рисунок 1.1 – Інтерфейс системи Bugzilla

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

MantisBT[3] характеризується простотою використання завдяки зрозумілому інтерфейсу і легкості встановлення, що робить систему зручною для невеликих команд. Вона також є відкритою, що дозволяє адаптувати функціонал під конкретні потреби, і має вбудовану систему сповіщень для оперативної комунікації. Недоліками є менш розвинений функціонал порівняно з комерційними рішеннями, застарілий вигляд інтерфейсу і обмеженість готових інтеграцій із сторонніми інструментами управління проєктами.

На рисунку 1.2 представлено інтерфейс системи MantisBT:

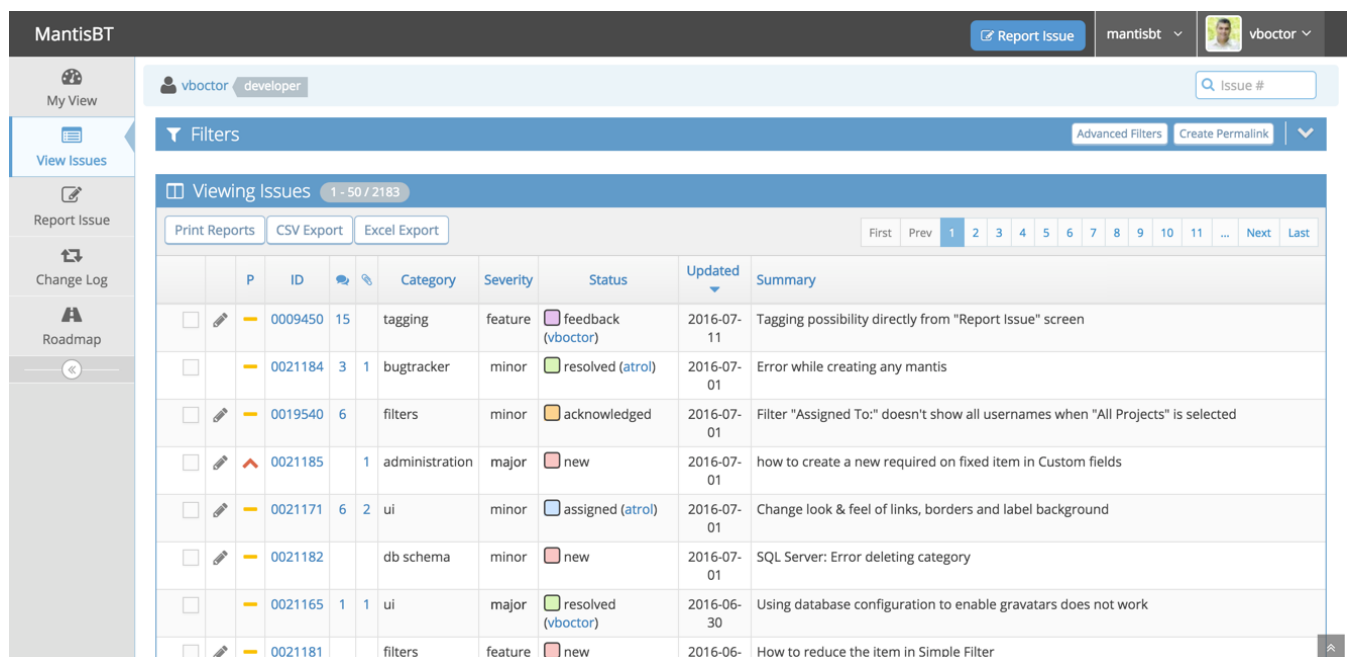


Рисунок 1.2 – Інтерфейс системи MantisBT

YouTrack[4] від JetBrains (рис. 1.3). вирізняється сучасним і інтуїтивно зрозумілим інтерфейсом, що полегшує роботу користувачів. Потужний пошук і фільтрація дозволяють швидко знаходити потрібну інформацію завдяки гнучким інструментам, а можливість налаштовувати робочі процеси, статуси і пріоритети відповідає методології конкретної команди. Інтеграція з екосистемою JetBrains сприяє ефективній роботі в середовищі розробки. Серед недоліків – висока вартість ліцензії для деяких компаній (хоча для малих команд існують безкоштовні версії), обмежена підтримка регіональних мов та необхідність додаткової конфігурації для деяких інтеграцій зі сторонніми сервісами.

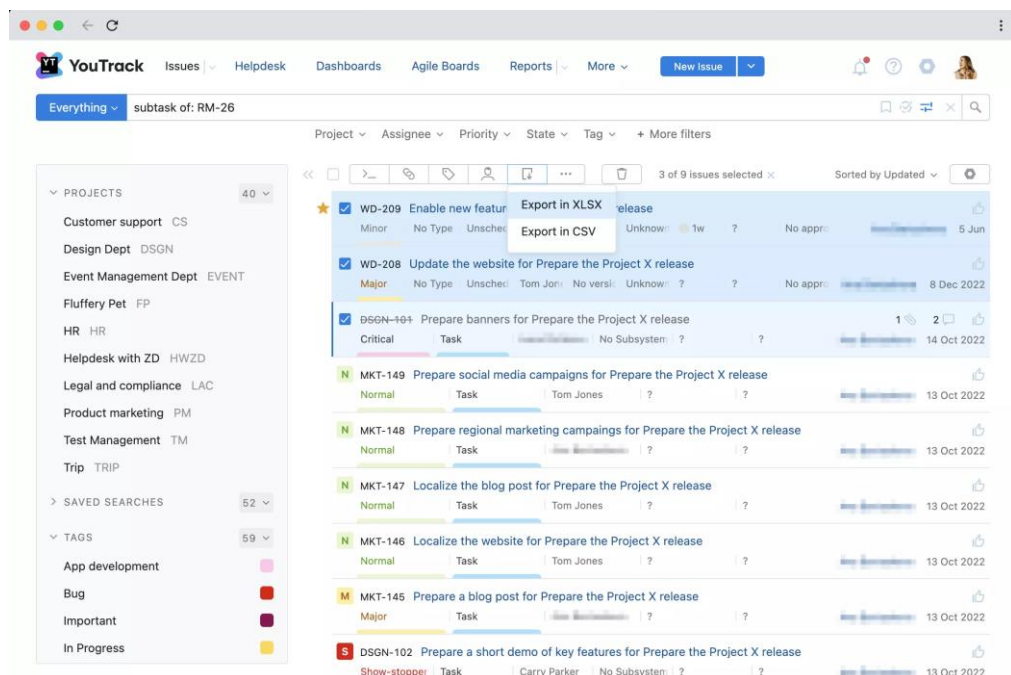


Рисунок 1.3 – Інтерфейс системи YouTrack

Redmine[5] (рис. 1.4) – це вебдодаток для управління проектами з відстеженням помилок, календарями та Gantt-діаграмами. Він забезпечує одночасну роботу з кількома проектами, інтеграцію з системами контролю версій і багатомовність, але має застарілий інтерфейс, обмежену функціональність і складну інтеграцію з сучасними CI/CD.

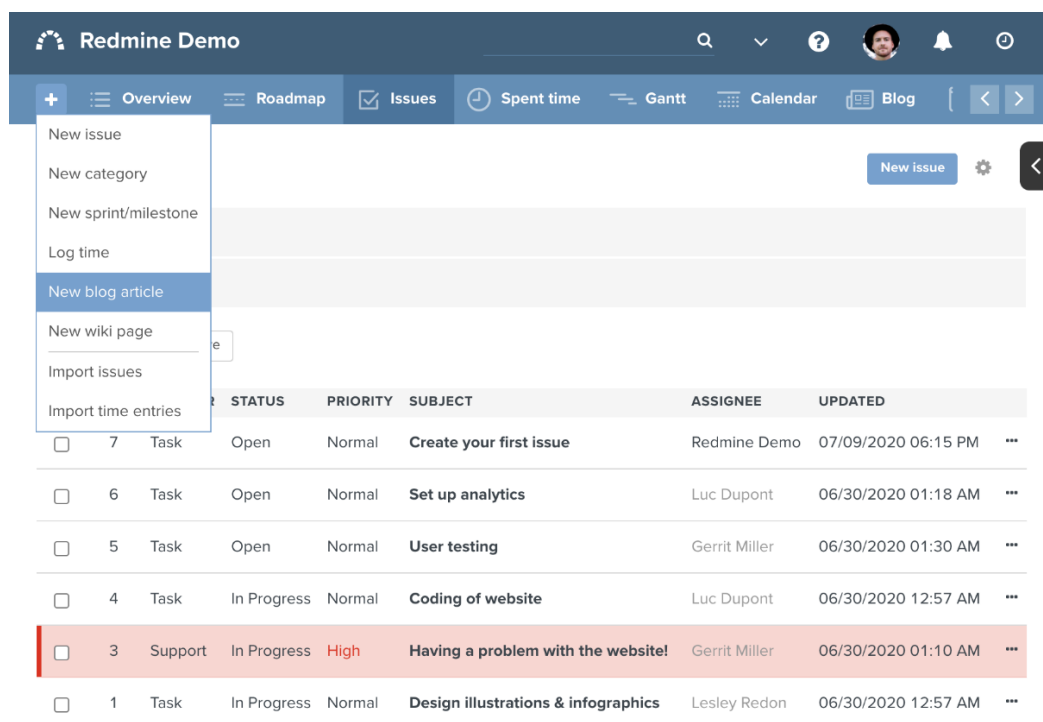


Рисунок 1.4 – Інтерфейс системи Redmine

Для більш зручного порівняння виокремимо основні характеристики засобів у таблиці 1.1.

Таблиця 1.1

Порівняльна таблиця систем відстеження помилок

Параметри	Bugzilla	MantisBT	YouTrack	Redmine	СВП-ІТ
Відкритість забезпечення	+	+	+ -	+	+
Простота використання	Інтерфейс може бути складним для нових користувачів	Простий, зрозумілий інтерфейс	Сучасний, інтуїтивно зрозумілий	Інтерфейс може здатися застарілим та вимагати налаштування	Простий, зрозумілий інтерфейс
Інтерфейс	Застарілий, не завжди інтуїтивний	Застарілий, але простий	Сучасний, привабливий	Застарілий	Сучасний, привабливий
Гнучкість налаштувань	Висока – можливість тонкого налаштування відповідно до потреб проєкту	Гнучка – налаштовується для невеликих команд	Дуже гнучка – дозволяє адаптувати робочі процеси під потреби команди	Гнучка, але з обмеженнями	Гнучка, але з обмеженнями
Українська локалізація	-	-	-	+ -	+
Функціональність	Основний функціонал, фокус на відстеженні помилок	Базовий функціонал з системою сповіщень	Повнофункціональна: відстеження, аналітика, управління завданнями	Відстеження помилок та управління проєктами	Основний функціонал, фокус на відстеженні помилок
Підтримка мобільних пристроїв	Відсутня або обмежена	Обмежена	Хороша – є мобільні додатки або адаптивний вебдизайн	Обмежена	Обмежена
Кросплатформенність	-	-	+	+ -	+ -

Таким чином, аналіз показав, що кожна з систем має свої унікальні переваги та недоліки, що дозволяє вибрати оптимальний інструмент залежно від конкретних вимог проєкту та розміру команди. Для розробки власної системи важливо враховувати досвід та функціональні особливості існуючих рішень, щоб створити продукт, що відповідає сучасним вимогам у сфері розробки ПЗ.

1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Вебдодаток буде побудований на клієнт-серверній архітектурі. Серверна частина реалізована на Node.js[6] з використанням Express[7], що дозволяє швидко створювати RESTful API для обробки запитів. Node.js, завдяки асинхронній моделі, ефективно обробляє одночасні запити від користувачів, а Express спрощує маршрутизацію, що критично для стабільності системи. Завдяки застосуванню TypeScript[8] код є строго типізованим: це зменшує кількість runtime-помилки, полегшує рефакторинг та забезпечує узгодженість даних між клієнтом і сервером (наприклад, типи для об'єктів «помилка» або «користувач»). Для системи це особливо важливо, оскільки сутності (помилки, коментарі, статуси) мають складні зв'язки, які легко описувати через інтерфейси TypeScript.

Клієнтська частина розроблена за допомогою React[9], що надає компонентний підхід для створення динамічних інтерфейсів. Наприклад, таблиця зі списком помилок може бути перевикористовуваним компонентом із можливістю сортування та фільтрації, а форма звіту про помилку — керуватися станом через хуки (useState, useEffect). Збірка додатку здійснюється за допомогою Vite[10], який забезпечує миттєве оновлення коду під час розробки та оптимізовану продуктивну збірку завдяки використанню ES-модулів. Це прискорює ітерації над інтерфейсом, де важливі деталі, як візуалізація статусів помилок. Стилзація реалізована через TailwindCSS[11], що дозволяє швидко створювати адаптивні інтерфейси без написання CSS з нуля. Наприклад, утилітарні класи Tailwind використовуються для індикації критичності помилок, а медіа-запити — для коректного відображення на пристроях.

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

Для забезпечення уніфікованого та сучасного дизайну інтерфейсу було обрано бібліотеку `shadcn/ui`[12]. Вона надає готові до використання компоненти з дотриманням сучасних UX/UI практик, які легко інтегруються з `TailwindCSS`. Використання `shadcn/ui` дозволяє значно пришвидшити розробку інтерфейсу, забезпечити єдиний стиль застосунку та зменшити час на створення власних компонентів. Крім того, бібліотека підтримує модульний підхід та надає інструменти для кастомізації, що дає змогу адаптувати компоненти під потреби проєкту.

Для реалізації локалізації інтерфейсу застосунку обрано бібліотеку `LinguiJS`[13]. `LinguiJS` забезпечує простий і зручний механізм роботи з повідомленнями та перекладами, підтримує динамічне завантаження мовних файлів і інтегрується з `React`-компонентами через макроси та `useLingui`. Це дозволяє ефективно організувати процес перекладу на кілька мов (зокрема, українську та англійську), забезпечити автоматичну перевірку відсутніх перекладів і мінімізувати повторювану роботу перекладачів. Крім того, `LinguiJS` має невеликий розмір пакету та високу продуктивність, що особливо важливо для підтримки швидкого завантаження сторінок та оптимізації користувацького досвіду.

Аутентифікація користувачів забезпечується за допомогою `Passport`[14], який підтримує різні стратегії авторизації (`JWT`, `OAuth 2.0`[15], `OpenID`). Для системи відстеження помилок це дозволяє реалізувати авторизацію та використання ролей (наприклад, адміністратор, розробник) та обмежити доступ до чутливих операцій (видалення помилок, зміна статусів). `Prisma`[16], як сучасний ORM, спрощує взаємодію з `PostgreSQL`[17] — реляційною базою даних, що ідеально підходить для структурованого зберігання даних про помилки, проєкти, користувачів та їх зв'язки. `PostgreSQL` забезпечує `ACID`-транзакції, що критично для цілісності даних під час оновлення статусів або додавання коментарів. Для адміністрування БД використовується `PgAdmin`[18] — інструмент з графічним інтерфейсом, де можна

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

перевіряти дані, налаштовувати індекси для прискорення пошуку помилок або аналізувати запити для оптимізації продуктивності.

Обраний набір технологій дозволяє створити систему відстеження помилок для ІТ-проектів з високою продуктивністю, надійністю та масштабованістю. Сучасна клієнтська частина на базі React і Tailwind забезпечує інтерактивність: користувачі можуть швидко фільтрувати помилки або сортувати їх за пріоритетом. Серверна частина на Node.js та Express обробляє запити ефективно навіть під час пікового навантаження, а Prisma і PostgreSQL гарантують швидкий доступ до даних та гнучке управління схемами. TypeScript як основа стеку запобігає типовим помилкам (наприклад, невідповідність типів у API-відповідях), а Passport і PgAdmin забезпечують безпеку та зручність адміністрування. Загалом, ця архітектура поєднує інструменти, оптимальні для розробки складних систем керування даними, де ключові вимоги — стабільність, зручність та масштабованість.

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

Висновки до першого розділу

У першому розділі було ґрунтовно досліджено предметну область систем відстеження помилок для ІТ-проектів, що дало змогу виокремити ключові вимоги до функціональності та архітектури такої платформи. Ретельний аналіз існуючих аналогів дозволив порівняти їхні переваги та недоліки, а також визначити найкращі практики та функціональні можливості, необхідні для ефективного виявлення та усунення дефектів програмного забезпечення.

На основі проведеного аналізу сформовано основні функціональні та нефункціональні вимоги до майбутньої системи, включаючи моделювання архітектури, механізми взаємодії користувачів, управління помилками, а також функції пошуку та фільтрації. Це дозволило систематизувати як статичні, так і динамічні параметри роботи системи, зосередившись на її ефективності, зручності використання та продуктивності.

Окрім того, у цьому розділі обґрунтовано вибір технологій та інструментальних засобів для реалізації системи, що забезпечує створення високопродуктивної, надійної та масштабованої платформи, яка відповідає сучасним вимогам ІТ-проектів.

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ВІДСТЕЖЕННЯ ПОМИЛОК

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

Система відстеження помилок надає зручний інтерфейс, за допомогою якого учасники проєкту можуть створювати, переглядати та оновлювати записи про виявлені помилки. Завдяки чітко структурованій моделі даних система забезпечує ефективне управління інформацією.

Основна мета системи – оперативне виявлення, глибокий аналіз та швидке усунення помилок у ІТ-проєктах. Це досягається через інтуїтивний функціонал, який охоплює весь життєвий цикл обробки технічних проблем – від їх реєстрації до повного закриття.

Користувачі отримують широкий спектр можливостей: від базової авторизації до складних аналітичних інструментів.

Особливо корисним є механізм класифікації помилок. Градація від критичних до низькопріоритетних дозволяє команді розробників чітко розставляти пріоритети та ефективно планувати роботу.

Інструменти фільтрації та пошуку забезпечують миттєвий доступ до необхідної інформації. Система не просто фіксує технічні проблеми, а перетворює процес їх обробки на структурований, прозорий та керований робочий процес. Це дозволяє командам підвищувати якість розробки, скорочувати час реакції на несправності та покращувати загальну ефективність проєктної роботи.

Опис діаграми варіантів використання наведено в таблицях додатку А.

Для наглядної взаємодії між елементами додатку було створено діаграму варіантів використання (рис 2.1).

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

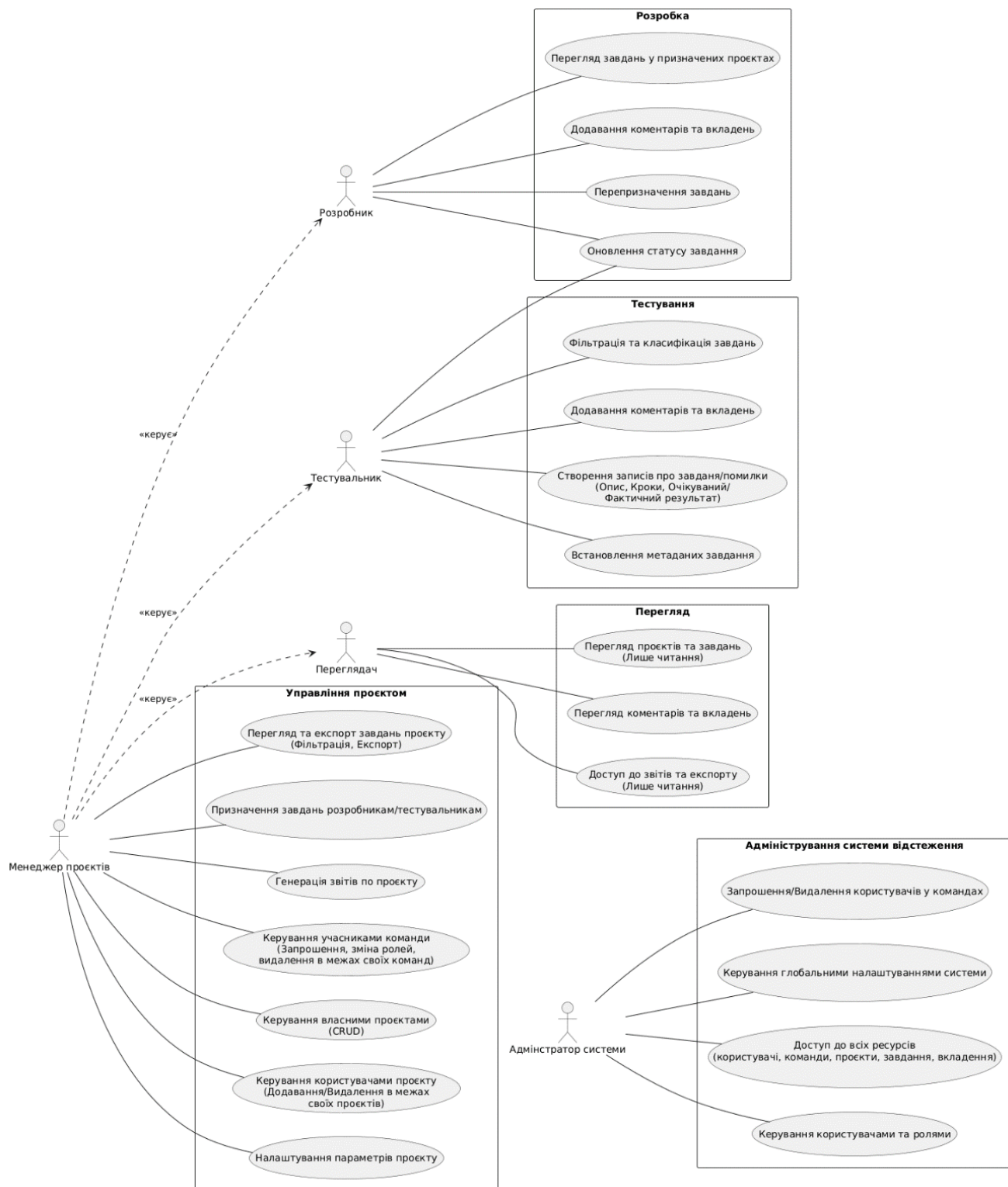


Рисунок 2.1 – Діаграма варіантів використання

Наведемо детальний опис функціональних вимог користувачів для системи відстеження помилок у ІТ-проєктах (таблиця 2.14). Ці вимоги враховують специфіку діяльності різних категорій користувачів, що беруть участь у процесі виявлення, аналізу та усунення дефектів у програмному забезпеченні.

Функціональні вимоги користувачів для системи відстеження помилок

Роль користувача	Основні вимоги
Адміністратор системи	Повний CRUD над усіма ресурсами: користувачі, команди, проєкти, помилки, вкладення. Управління користувачами та ролями: додавання, редагування, видалення, призначення/скасування ролей. Моніторинг активності системи (звіти, сповіщення про критичні події). Запрошення/видалення користувачів у будь-якій команді.
Розробник	Перегляд усіх помилок у проєктах, до яких належить. Оновлення статусів помилок. Додавання коментарів і вкладень до помилки. Переназначення помилок, над якими має права.
Тестувальник	Створення нових звітів про помилки: зазначення опису, кроків відтворення, очікуваного/фактичного результату, прикріплення скріншотів. Встановлення метаданих помилки: пріоритет, категорія, середовище. Зміна статусу помилки. Фільтрація та класифікація: пошук, сортування, призначення категорій та пріоритетів. Переназначення помилок, над якими має права.

Роль користувача	Основні вимоги
Менеджер проекту	Створення, читання, оновлення й видалення власних проєктів. Додавання/видалення користувачів у межах свого проєкту. Налаштування проектних параметрів: пріоритети, мітки, робочі процеси. Перегляд, фільтрація та експорт усіх помилок у власних проєктах. Призначення помилок розробникам чи тестувальникам. Генерація звітів про проєкт (відкриті/закриті помилки, швидкість виконання тощо). Запрошення, зміна ролей, видалення учасників у своїх командах.
Переглядач	Перегляд дозволених проєктів і відповідних помилок. Ознайомлення з коментарями та вкладеннями. Доступ до статистики і можливість експорту звітів у режимі «лише читання». Не може створювати, редагувати чи видаляти жодної помилки, не може коментувати чи змінювати налаштування.

Щодо нефункціональних вимог, користувацький інтерфейс має бути максимально зрозумілим. Це гарантує, що навіть нові користувачі зможуть легко освоїти систему за лічені хвилини, а досвідчені — працювати з нею вільно, без необхідності звертатися до будь-яких додаткових посібників.

Система повинна реагувати на дії користувача не довше ніж за 3 секунди, а для складних операцій — не більше 90 секунд.

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

Система має стабільно працювати навіть при великому навантаженні, бути доступною 24/7, забезпечувати безперервне збереження даних із мінімальними збоями. Особливу увагу слід приділяти цілісності даних під час транзакцій, оновлень і видалень.

Безпека реалізується шляхом захисту персональних даних користувачів і проєктної інформації за сучасними стандартами: шифрування, резервне копіювання, контроль доступу, багаторівнева аутентифікація та авторизація.

Система має бути масштабованою, здатною до адаптації при зростанні кількості даних (користувачів, помилок, проєктів), без втрати продуктивності. Для цього можлива реалізація розподіленої архітектури.

Щодо системних вимог, програмне забезпечення повинно працювати на пристроях із мінімум 2 Гб оперативної пам'яті, процесором із частотою не менше 1.4 GHz і стабільним підключенням до Інтернету. Система має коректно функціонувати на стаціонарних комп'ютерах використовуючи TailwindCSS.

Серверна частина працює на Node.js з Express, що дає змогу ефективно обробляти запити через RESTful API. TypeScript використовується для типізованого коду, що зменшує ризик помилок під час виконання.

Передбачено API для інтеграції з такою системою, як Jira, та можливість експорту даних у форматі CSV або JSON. Сервер підтримує технології Node.js для високої продуктивності при навантаженнях, а аутентифікація реалізується через Passport.js.

Система розроблена на основі модульної архітектури, де кожна сутність представлена відповідним класом і набором методів для реалізації бізнес-логіки.

На рисунку 2.2. наведено діаграму класів, яка використовується для його роботи.

Опишемо сутності та їхні основні методи (таблиця 2.2):

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

Основні сутності та методи в системі

Сутність	Основні методи	Опис
User	getAllUsers(), createUser(userData), findUserById(id), deleteUser(id)	Забезпечує реєстрацію, пошук та управління обліковими записами користувачів
Issue	findIssueById(id), getIssuesByUserId(userId), getIssuesByTeamId(teamId), createIssue(data), updateIssue(id, data), deleteIssueById(id), exportIssuesForProject(projectId)	Відповідає за реєстрацію, оновлення, видалення й масову обробку помилок; пошук за різними параметрами (проект, користувач, команда), експорт даних.
Project	findProjectById(id), getUsersAssociatedWithProject(projectId), createProject(data), deleteProject(id)	Управління проектами: створення/видалення, пошук за різними критеріями (за користувачем, командою), отримання статистики помилок у межах проекту.

Сутність	Основні методи	Опис
Team	createTeam(), deleteTeam(), getTeamMembersByTeamId(), inviteMemberByEmail()	Управління командами: створення, видалення, отримання списків команд, а також отримання статистики помилки (за статусами, типами, пріоритетами тощо) для конкретної команди.
TeamMember	inviteMemberByEmail(teamId, email, role), deleteTeamMember(teamMember Id)	Організовує зв'язки «користувач – команда»: запрошення, додавання, видалення учасників, отримання складу команди.
Comment	getCommentById(id), createComment(data), updateComment(id, data), deleteCommentById(id)	Керує коментарями до помилки: перегляд, створення, редагування, видалення окремих або множини коментарів для конкретної помилки.
Attachment	uploadAttachment(issueId, userId, file), deleteAttachment(attachmentId)	Здійснює операції з вкладеннями (завантаження файлів, видалення, отримання списку вкладень) у межах певної помилки.

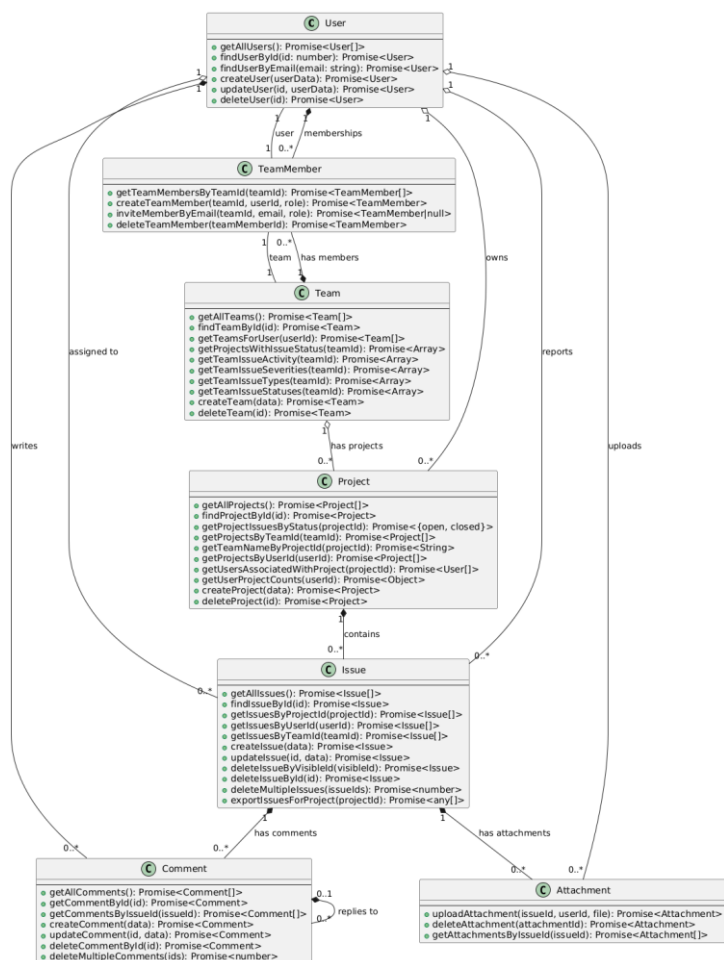


Рисунок 2.2. – Діаграма класів системи

2.2. Розробка бази даних системи

Для зберігання даних про користувачів, проекти, команди та помилки, виявлені під час розробки, застосовується система управління базами даних PostgreSQL. Завдяки реляційній структурі та можливостям SQL, PostgreSQL гарантує цілісність даних, високу продуктивність та стабільну роботу з великими обсягами інформації. Для полів, що представляють тип помилки, серйозність, пріоритет та платформу, використано тип ENUM, що забезпечує узгодженість значень та спрощує їх перевірку.

База даних системи складається з семи основних таблиць:

1. User – таблиця, яка зберігає дані про користувачів системи.
2. Team – таблиця, що містить інформацію про команди, які залучені до розробки проектів.

3. TeamMember – таблиця, яка відображає зв'язок між користувачами та командами, з зазначенням ролі учасника.
4. Project – таблиця, яка зберігає дані про ІТ-проекти, у яких використовується система відстеження помилок.
5. Issue – таблиця, що містить інформацію про помилки, їх характеристики, статус та інші параметри.
6. Comment – таблиця для збереження коментарів до помилок.
7. Attachment – таблиця для збереження вкладених файлів, пов'язаних з помилками, які слугують візуальними доказами або доповненням до опису проблеми.

Опис кожної з таблиць бази даних наведений у таблицях 2.3 – 2.9. На рисунку 2.3 наведено діаграму бази даних, яка використовуються для його роботи.

Таблиця 2.3

Опис таблиці Project

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор проєкту
name	varchar	Назва проєкту
platformType	enum	Тип платформи (Desktop, Android, iOS, Web, MultiPlatform)
ownerId	int (опціонально)	Ідентифікатор власника проєкту (посилання на таблицю User)
teamId	int (опціонально)	Ідентифікатор команди, яка відповідає за проєкт
createdAt	date/time	Дата створення проєкту
updatedAt	date/time	Дата останнього оновлення даних про проєкт

Таблиця 2.4

Опис таблиці TeamMember

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор запису про участь користувача у команді
teamId	int	Ідентифікатор команди (посилання на таблицю Team)
userId	int	Ідентифікатор користувача (посилання на таблицю User)
role	varchar	Роль учасника в команді (наприклад, "member", "lead")
joinedAt	date/time	Дата приєднання користувача до команди

Таблиця 2.5

Опис таблиці Comment

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор коментаря
body	varchar	Текст коментаря
createdAt	date/time	Дата й час створення коментаря
updatedAt	date/time	Дата й час останнього оновлення коментаря
userId	int	Ідентифікатор користувача
issueId	int	Ідентифікатор помилки
parentId	int (nullable)	Ідентифікатор батьківського коментаря або NULL, якщо це кореневий коментар

Таблиця 2.6

Опис таблиці Team

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор команди
name	varchar	Назва команди
description	varchar	Опис команди (опціонально)
createdAt	date/time	Дата створення команди
updatedAt	date/time	Дата останнього оновлення даних про команду

Таблиця 2.7

Опис таблиці Attachment

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор вкладки
issueId	int	Ідентифікатор помилки
userId	int (nullable)	Ідентифікатор користувача
filename	varchar	Назва файлу вкладки
url	varchar	URL або шлях до файлу
mimeType	varchar	MIME-тип файлу
size	int	Розмір файлу в байтах
createdAt	date/time	Дата й час створення запису про вкладення
updatedAt	date/time	Дата й час останнього оновлення запису про вкладення

Опис таблиці Issue

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор запису про помилку
visibleId	varchar	Унікальний читабельний ідентифікатор (з використанням nanoid)
projectId	int	Посилання на проєкт (ідентифікатор з таблицею Project)
reporterId	int (опціонально)	Ідентифікатор користувача, який повідомив про помилку
assignedTo	int (опціонально)	Ідентифікатор користувача, якому призначено вирішення помилки
title	varchar	Заголовок помилки
description	varchar (опціонально)	Опис помилки
severity	enum	Пріоритет помилки (Low, Medium, High, Critical)
type	enum	Тип помилки (Bug, Observation, Suggestion, Improvement, NewFeature)
status	enum	Статус помилки (New, InProgress, Fixed, NotFixed, Cancelled, Closed)
isPinned	boolean	Позначка важливості або закріплення помилки
dateAdded	date/time	Дата створення запису про помилку
updatedAt	date/time	Дата останнього оновлення запису

Опис таблиці Users

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор користувача
username	varchar	Унікальне ім'я користувача
email	varchar	Email користувача
role	varchar	Роль користувача (наприклад, «user», «admin»)
hashedPassword	varchar	Захищений пароль користувача
firstName	varchar	Ім'я користувача
lastName	varchar	Прізвище користувача
createdAt	date/time	Дата реєстрації
updatedAt	date/time	Дата останнього оновлення запису

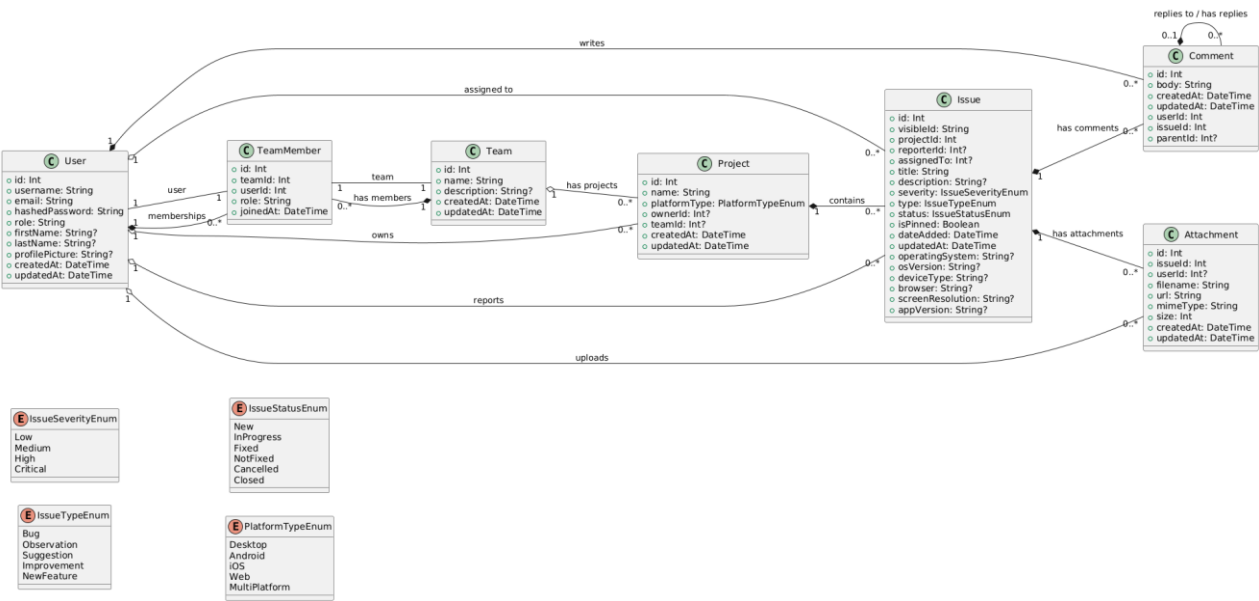


Рисунок 2.3 – Діаграма бази даних для системи

Проектування бази даних враховує потенційне збільшення обсягів даних у майбутньому, пов'язане зі зростанням кількості проєктів, помилок і користувачів. Розподілена архітектура PostgreSQL у поєднанні з Prisma дозволяє ефективно обробляти великі навантаження, підтримуючи високу продуктивність.

2.3. Проектування та реалізація алгоритмів роботи системи

Система відстеження помилок для IT-проєктів спроектована так, щоб бути зручною для всіх типів користувачів.

Після входу в систему розробник бачить перелік звітів, що стосуються проєктів, у яких він бере участь. Він може переглядати детальну інформацію про кожен звіт, включаючи опис, коментарі та історію змін. Також розробник має можливість змінювати статуси звітів (наприклад, з «новий» на «в роботі» або «виправлено»), додавати власні коментарі, повідомляти про завершення завдання та приймати нові задачі від менеджера або адміністратора.

Тестувальник має змогу створювати нові звіти про помилки, описуючи, як їх відтворити, що очікується в нормальному випадку, і що сталося насправді. Після створення звіту він може додавати уточнення, підтверджувати чи спростовувати відтворюваність проблеми, а також отримувати повідомлення про зміну статусу помилок, які він додав.

Менеджер проєкту бачить загальну аналітичну картину: кількість активних, нових, вирішених та закритих помилок. Він може переглядати призначені завдання, оцінювати ефективність команди, змінювати розподіл навантаження залежно від пріоритетів, а також планувати подальші дії для вдосконалення процесу розробки та співпраці між учасниками проєкту.

Адміністратор системи є суперкористувачем і має необмежені права. Він може виконувати повний CRUD над усіма ресурсами, керувати глобальними налаштуваннями, призначати або скасовувати будь-які ролі, а також переглядати журнали аудиту системи. Така роль зазвичай призначається технічному спеціалісту з повним рівнем довіри, що відповідає за підтримку платформи.

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

Переглядач має доступ до системи у режимі лише читання. Він може переглядати дозволені проекти, ознайомлюватися з наявними звітами про помилки, їхніми статусами, коментарями та вкладеннями, а також переглядати аналітичні панелі й експортувати звіти. Переглядач не має права створювати, редагувати або коментувати помилки. Ця роль ідеально підходить для стейкхолдерів, таких як замовники, представники бізнесу або зовнішні контролери, які зацікавлені в моніторингу прогресу, але не беруть активної участі у розробці.

Для реалізації цих сценаріїв було спроектовано ключові алгоритми, що визначають основні бізнес-процеси системи. Центральним є життєвий цикл звіту про помилку. Спочатку тестувальник ініціює створення звіту, вводячи заголовок, опис, пріоритет та тип помилки. Система перевіряє повноту й коректність введених даних, генерує унікальний читабельний ідентифікатор, присвоює запису новий статус та зберігає його в таблиці «Issue» із посиланнями на проект та автора. Далі менеджер проекту або інший уповноважений користувач призначає відповідального виконавця — система оновлює відповідне поле. Коли розробник бере завдання в роботу, він змінює статус на «В процесі», а система фіксує час цієї зміни. Після виправлення дефекту розробник встановлює статус «Виправлено» і додає коментар із описом реалізованого рішення. Тестувальник перевіряє виправлення: якщо воно успішне, статус змінюється на «Закрито», а якщо ні — повертається до «В прогресі» із додатковими коментарями.

Окремо передбачено алгоритм скасування помилки. Якщо звіт про помилку став неактуальним, був створений помилково або є дублікатом іншого, користувач з відповідними правами може змінити його статус на «Скасовано». Ця дія виключає завдання з активних робочих процесів, однак запис не видаляється з бази даних, а зберігається для цілісності історії проекту та можливості подальшого аналізу.

На будь-якому етапі користувачі можуть додавати коментарі чи вкладення. Алгоритм створення коментаря додає запис до таблиці «Comment» із посиланням на конкретну помилку та користувача, а алгоритм завантаження файлів передає їх у хмарне сховище і зберігає URL-адресу та метадані в таблиці «Attachment».

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

Окрім життєвого циклу помилки, важливими є механізми управління доступом. Користувач із відповідними правами може запросити нового члена команди, ввівши відповідну пошту користувача: система перевіряє наявність такого користувача в базі та створює зв'язок у таблиці «TeamMember», призначаючи відповідну роль. Адміністратор або менеджер проєкту може змінювати роль учасника.

Функціональність аналітичного модуля також базується на серії алгоритмів обробки даних. При виборі команди або проєкту система ініціює виконання агрегуючих запитів до бази даних. Для побудови кругових діаграм алгоритм групує всі помилки за полями, підраховуючи кількість записів у кожній групі. Для теплової карти активності виконується агрегація записів за датою створення та оновлення, що дозволяє візуалізувати інтенсивність роботи по днях. Гістограмний графік вимагає складнішого запиту, який групує помилки спочатку за проєктами, а потім за статусами всередині кожного проєкту. Результати цих операцій повертаються на клієнтську частину у вигляді структурованих даних для їх візуалізації.

2.4. Реалізація функціоналу системи відстеження помилок

Розглянемо основні етапи користування системою. Щоб почати роботу з системою, користувачеві необхідно зареєструвати новий обліковий запис. Після цього він отримає доступ до всіх можливостей системи.

Для забезпечення безпечного доступу до системи використовується аутентифікація через Google з підтримкою Passport.js. При зверненні за маршрутом «/auth/google» викликається функція Passport, яка забезпечує отримання прав доступу до email та профілю користувача. Після успішного логіну користувач перенаправляється на сторінку проєктів, або у випадку невдалого входу – на сторінку логіну.

Маршрут «/currentUser» дозволяє з'ясувати, чи користувач автентифікований, і повертає дані про нього або помилку статусу 401. Функціональність виходу з системи реалізована маршрутом «/logout», що

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				32
Змн.	Арк.	№ докум.	Підпис	Дата		

використовує вбудовану функцію «req.logout» для очищення сесійних даних. Для взаємодії з системою відстеження помилок реалізовано окремі маршрути для обробки запитів, пов'язаних зі звітами про помилки, проєктами, командами та користувачами. Логіка роботи з базою даних інкапсульована в цих маршрутах, а доступ до бази даних здійснюється за допомогою ORM Prisma.

Всі маршрути, що реалізують функціонал системи, інтегровані у серверний додаток на Express. Для забезпечення коректної обробки запитів налаштовано використання модулів cors та express-session для забезпечення підтримки крос-доменних запитів та зберігання сесійних даних.

Додавання вкладень до звітів про помилки є ключовою можливістю, що дозволяє користувачам надавати додаткові контекстні дані. Цей функціонал реалізовано за допомогою інтеграції з Google Cloud Storage (GCS)[19] для зберігання файлів та бібліотеки Multer[20] для обробки багаточастинних формних даних серверної частини.

Коли користувач завантажує файл, запит надходить на маршрут «/issues/:issueId/attachments». Тут Multer перехоплює вхідний файл, парсить його та зберігає у пам'яті сервера як буфер. Після цього викликається контролер «uploadAttachmentController», який отримує ідентифікатор помилки, ідентифікатор користувача та об'єкт файлу.

У функції «uploadAttachment» дані файлу передаються до допоміжної функції «uploadFileToGCS». Ця функція, використовуючи бібліотеку «@google-cloud/storage», ініціалізує з'єднання з GCS, вказуючи ідентифікатор проєкта з конфігурації середовища. Вона генерує унікальне ім'я файлу, щоб уникнути конфліктів. Потім буфер файлу завантажується в зазначений бакет GCS з відповідним типом. Після успішного завантаження GCS повертає публічну URL-адресу збереженого файлу.

Отримана URL-адреса, разом з іншими метаданими файлу та ідентифікаторами, зберігається у базі даних через ORM Prisma в таблиці Attachment. Це дозволяє системі зв'язати завантажений файл з конкретною

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

проблемою. У разі успішного збереження даних, сервер повертає об'єкт вкладки з HTTP статусом 201. Якщо ж виникають проблеми під час завантаження до GCS або збереження в базі даних, повертається відповідний HTTP статус 500 з повідомленням про помилку.

Функцію видалення вкладень реалізовано через маршрут «/attachments/:attachmentId». Спочатку контролер шукає запис у таблиці вкладень за вказаним ідентифікатором. Якщо такий запис існує, викликається допоміжна функція, яка за URL файлу визначає його ім'я та відправляє запит на видалення з хмарного сховища. Після успішного видалення зберігається узгоджений стан — запис вилучається з бази даних. Якщо ж вкладки не знайдено, система повертає код 404, а в разі інших збоїв — код 500 із повідомленням про помилку.

Щоб отримати всі файли, прикріплені до певної помилки, передбачено маршрут «/issues/:issueId/attachments». Контролер за допомогою ідентифікатора помилки звертається до бази даних і вибирає всі записи з таблиці вкладень, що належать до цього issue. Результат у вигляді списку URL та метаданих повертається клієнту з кодом 200.

Для забезпечення гнучкості робочих процесів у системі реалізовано функціонал експорту звітів про помилки до зовнішньої системи управління проектами Jira. Ця можливість дозволяє командам розробників переносити дані для подальшого опрацювання у звичному для них середовищі. Серверна частина інкапсульована у двох основних контролерах: один для отримання списку проєктів з Jira, інший — для безпосереднього експорту.

Процес експорту ініціюється POST-запитом на ендпоінт «/project/:id/jira/export-issues». У тілі запиту клієнт передає адресу сервера Jira та ключ проєкту, до якого будуть додані задачі. Аутентифікація відбувається через заголовок Authorization за схемою Basic, що містить закодовані в Base64 електронну пошту та токен доступу користувача Jira. Сервер розкодує ці дані для подальшої взаємодії з Jira API.

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Після валідації вхідних даних та успішного отримання облікових даних, система ініціалізує клієнт Jira за допомогою бібліотеки jira-client[21]. Налаштування клієнта включають вказання протоколу, адреси хоста, облікових даних користувача та версії API.

Наступним кроком є вибірка всіх звітів про помилки, пов'язаних з поточним проєктом, з локальної бази даних. Цю операцію виконує сервісна функція «exportIssuesForProject», яка використовує Prisma для формування запиту. Далі система в циклі обробляє кожну отриману помилку. Для кожного запису формується спеціальний об'єкт, структура якого відповідає вимогам Jira API для створення нової задачі. Відбувається перехід внутрішніх полів системи на поля Jira.

Наприклад, допоміжні функції «mapIssueTypeToJira» та «mapSeverityToJiraPriority» перетворюють типи та рівні критичності помилок (Bug, Critical) на відповідні аналоги в Jira (Bug, Highest). Опис проблеми форматується згідно зі структурою документа Jira. Сформований об'єкт відправляється в Jira через метод «jira.addNewIssue()».

Якщо створення пройшло успішно, система перевіряє, чи відрізняється статус локальної помилки від початкового статусу в Jira. Якщо так, виконується додатковий запит для зміни статусу. Для цього спочатку завантажується список можливих переходів (transitions) для створеної задачі, знаходиться відповідний до потрібного статусу і виконується зміна статусу через метод «jira.transitionIssue()». Якщо задача закривається, до неї додається поле визначення (resolution).

Конфігураційний код серверного застосунку, аутентифікації, роботи з вкладеннями та експортування даних наведено в додатку Б.

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

Висновки до другого розділу

У другому розділі було детально розглянуто та спроектовано архітектуру та основні компоненти системи відстеження помилок. Завдяки визначенню варіантів використання та об'єктно-орієнтованої структури, вдалося чітко окреслити функціональні можливості системи для різних категорій користувачів, таких як адміністратори, розробники, тестувальники, менеджери проєктів та переглядачі. Кожен варіант використання був детально описаний, що дозволило зрозуміти взаємодію користувачів із системою та їхні ключові потреби.

Особлива увага приділялася розробці бази даних системи, де було обрано PostgreSQL як основну СУБД. Це рішення забезпечує надійне зберігання даних, високу продуктивність та масштабованість, що є критично важливим для системи, яка працюватиме з великими обсягами інформації про помилки, проєкти та користувачів. Детальне проектування семи основних дозволило оптимізувати структуру даних та забезпечити їх цілісність.

Крім того, у цьому розділі було розглянуто проектування та реалізацію алгоритмів роботи системи, що включає логіку взаємодії користувачів з інтерфейсом, обробку запитів на серверній частині та інтеграцію з зовнішніми сервісами. Зокрема, було детально описано реалізацію таких ключових функціональних можливостей, як аутентифікація через Google, управління вкладеннями за допомогою Google Cloud Storage, а також експортування даних про помилки до Jira. Ці реалізації підтверджують здатність системи надавати гнучкий інструментарій для ефективного управління процесом виправлення помилок у IT-проєктах.

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ВІДСТЕЖЕННЯ ПОМИЛОК

3.1. Порядок встановлення та налаштування системи

Для встановлення та запуску розробленого додатку не потрібно жодного спеціального обладнання чи додаткової конфігурації: досить стандартного набору інструментів, який застосовується для більшості сучасних вебсайтів на будь-якій операційній системі. Це дозволяє швидко розгорнути систему та забезпечує зручність у використанні, що позитивно впливає на досвід кінцевих користувачів.

Переходячи до наступного етапу - розробки діаграми розгортання, яка описує архітектуру програмного комплексу. Діаграма розгортання є ключовим інструментом для візуалізації фізичної архітектури програмного комплексу. Вона відображає, як програмні компоненти розміщуються та взаємодіють на різних апаратних пристроях у реальному часі. Ця діаграма допомагає зрозуміти топологію розподіленої системи, її конфігурацію, а також фізичні зв'язки та шляхи передачі інформації між елементами. На ній зображуються вузли виконання, процеси та об'єкти, які є виконуваними файлами або динамічними бібліотеками.

Компоненти, що не використовуються під час виконання, на діаграмі розгортання не відображаються. Це забезпечує повну картину реалізації системи, демонструючи її особливості та відмінності від діаграм логічного представлення.

Діаграма розгортання є заключним етапом специфікації моделі об'єктно-орієнтованого програмування для конкретної програмної системи, надаючи графічне зображення процесорів, пристроїв, процесів та їхніх зв'язків.

Розгортання системи відстеження помилок починається з клієнтського пристрою — персонального комп'ютера, де працює веббраузер. Він завантажує клієнтську частину застосунка — саме тут відбувається вся взаємодія користувача: маршрутизація між розділами, локалізація українською та англійською мовами, візуалізація аналітичних даних. Вхідні та вихідні дані обробляє модуль управління даними, який передає запити до серверної частини.

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

Для опису архітектури системи було створено діаграму розгортання (рис. 3.1).

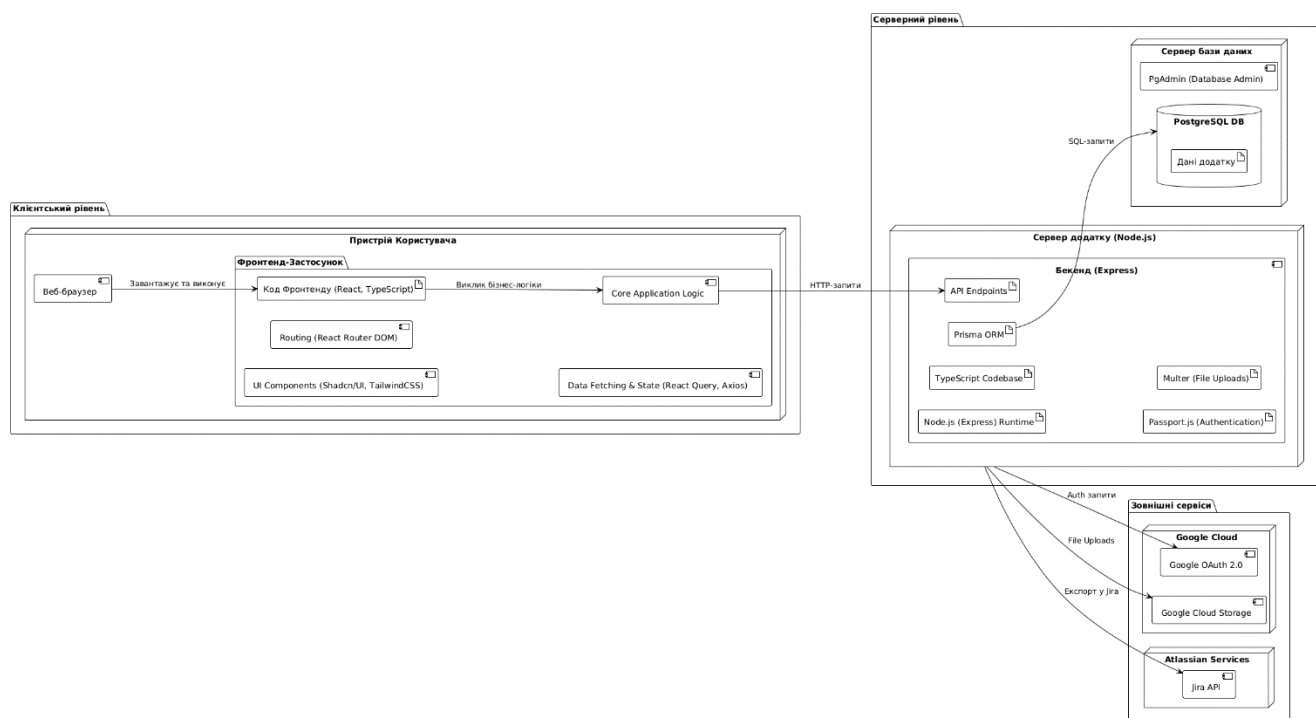


Рисунок 3.1. – Діаграма розгортання системи

Серверна частина складається з Node.js/Express-серверу та коду на TypeScript, що реалізує бізнес-логіку, — тут же працює Prisma ORM для доступу до PostgreSQL. За авторизацію відповідає Passport.js, за обробку завантажених файлів — Multer. Сервер формує REST-API, до якого клієнтська частина звертається для зчитування і запису даних у базу.

Окремий сервер PostgreSQL адмініструється через PgAdmin і зберігає всю необхідну інформацію. Для користувацької аутентифікації та роботи з файлами система інтегрована з Google Cloud (OAuth 2.0 та Cloud Storage), а для передачі помилок у робочі процеси розробників — з Jira API від Atlassian.

У підсумку браузер користувача постійно взаємодіє з клієнтом, який надсилає API-запити на сервер додатку. Сервер опрацьовує їх, обмінюється даними з базою, звертається до зовнішніх сервісів і повертає результати. Така послідовна взаємодія забезпечує безперебійну та ефективну роботу системи відстеження помилок.

3.2. Структура інтерфейсу системи відстеження помилок

При першому переході на вебдодаток користувач потрапляє на головну сторінку, яка виконує роль вітрини та точки входу до системи. Тут зібрано найважливіші елементи, що знайомлять з можливостями системи, а також надають доступ до авторизації. Основна мета головної сторінки — швидко показати користувачу цінність додатку та перенаправити його до відповідної дії: увійти або зареєструватися (рис 3.2 – рис. 3.3).

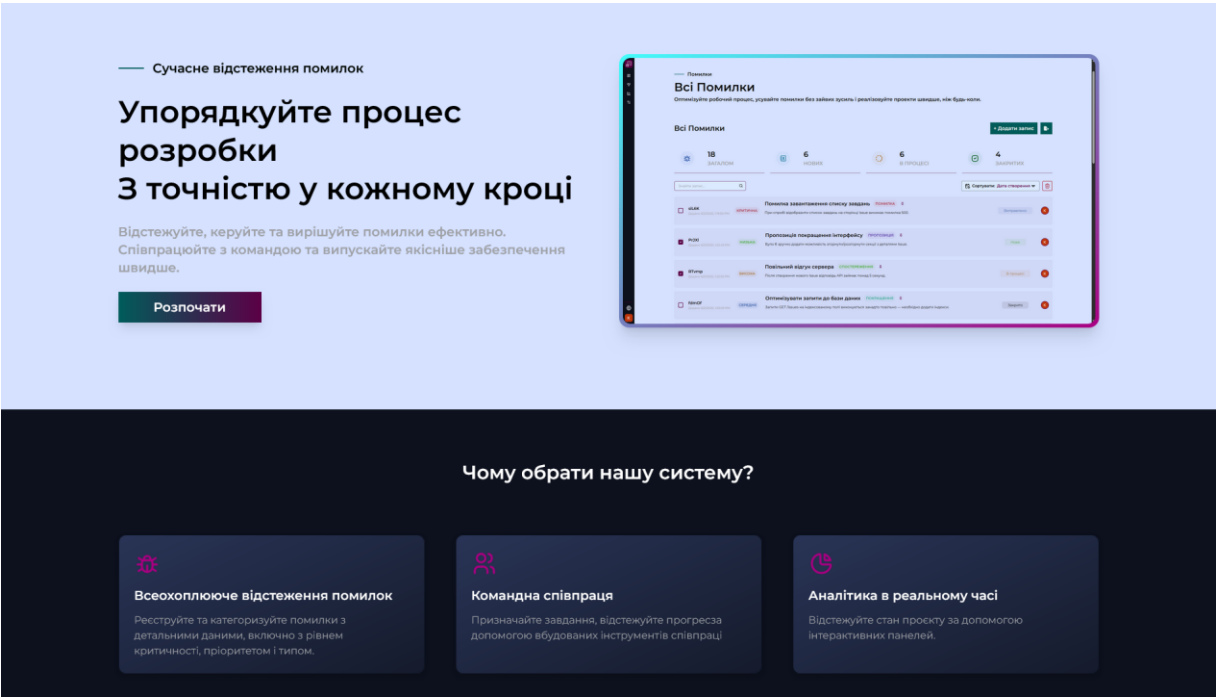


Рисунок 3.2. – Головна сторінка системи

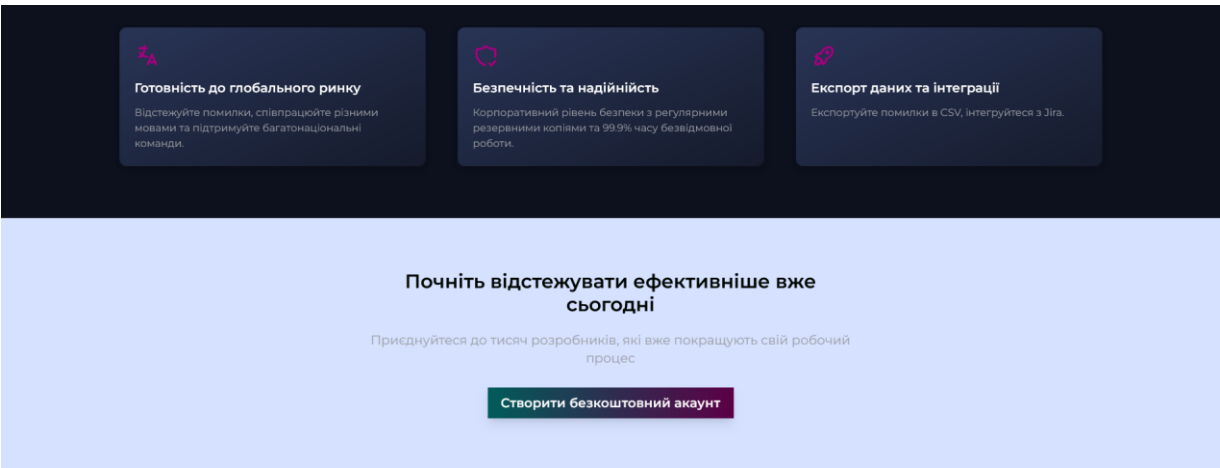


Рисунок 3.3. – Головна сторінка системи

Для доступу до функціоналу системи користувач повинен пройти процес авторизації. На сторінці входу користувачеві пропонується ввести свої облікові дані. Проте, для максимальної зручності та безпеки, система надає можливість авторизації або реєстрації через Google OAuth (рис 3.4).

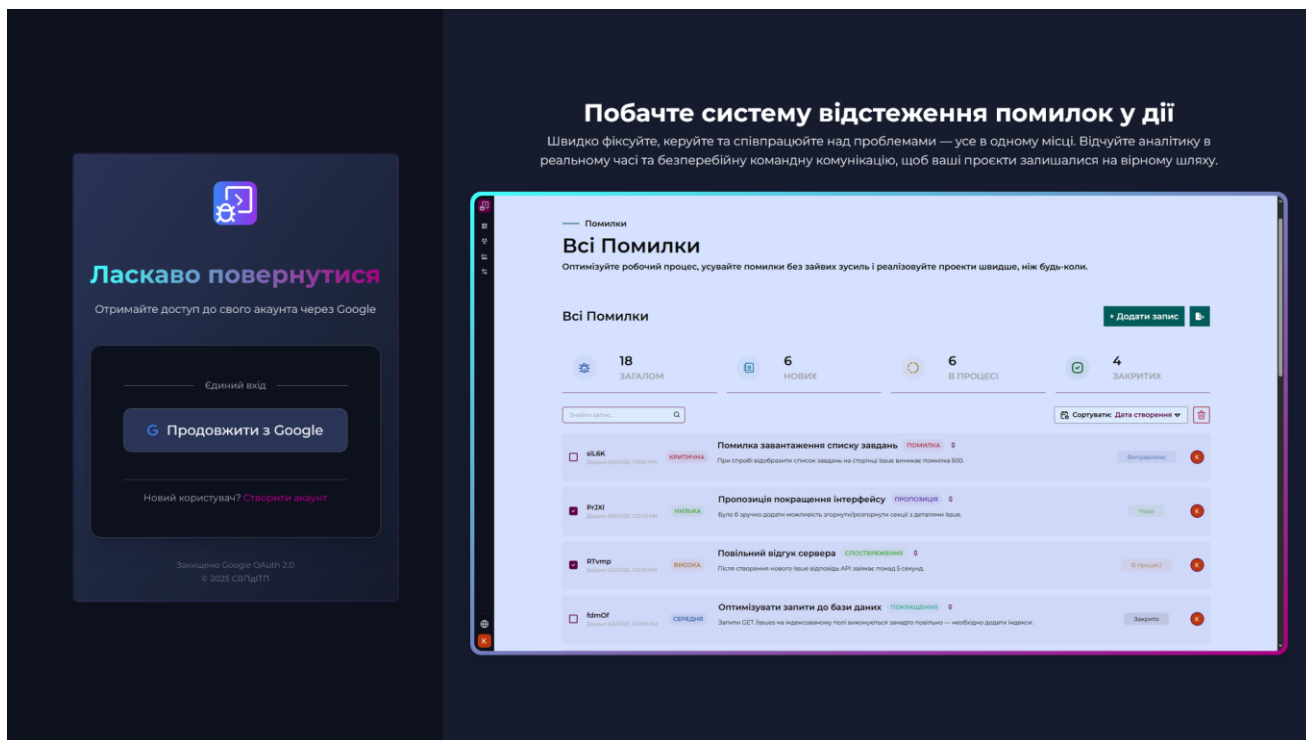


Рисунок 3.4. – Сторінка авторизації

При виборі опції "Продовжити з Google" користувач перенаправляється на сторінку авторизації Google, де він може підтвердити свою особу через свій обліковий запис Google. Це дозволяє уникнути необхідності запам'ятовувати додаткові логіни та паролі, значно спрощуючи процес доступу до системи (рис. 3.5).

В разі успішної аутентифікації користувача, йому буде надано доступ до всіх функцій та можливостей вебзастосунку та перенаправлено на головну сторінку з відображення бічної панелі.

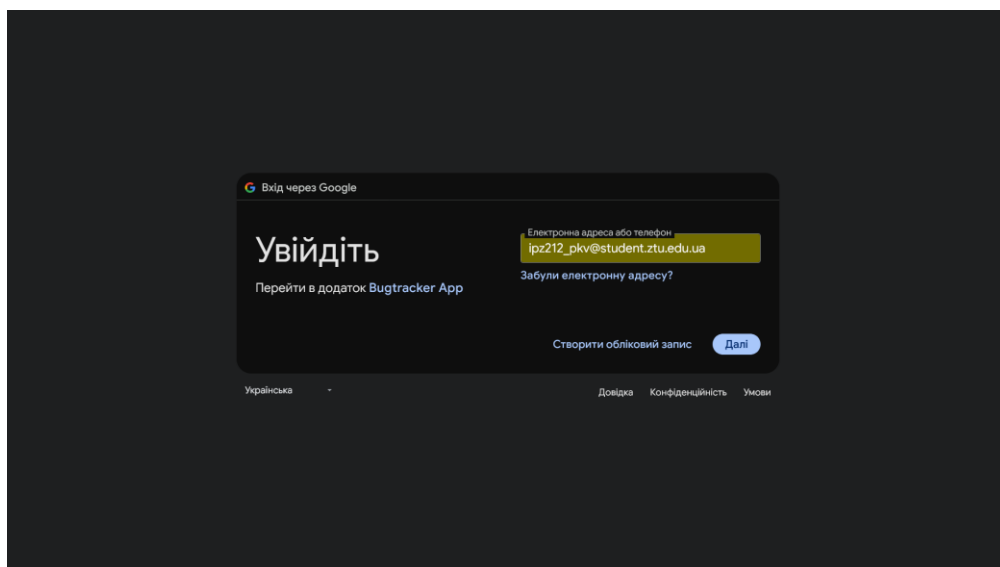


Рисунок 3.5. – Авторизація через Google

Після успішного входу до системи можна через бічне меню перейти на сторінку «Команди». У вигляді карток відображаються всі команди, до яких користувачу належить або має доступ.

Кожна картка показує назву команди та містить кнопку для миттєвого переходу до перегляду чи редагування деталей (рис 3.6). Створення нової команди показано на рисунку 3.7 та 3.8.

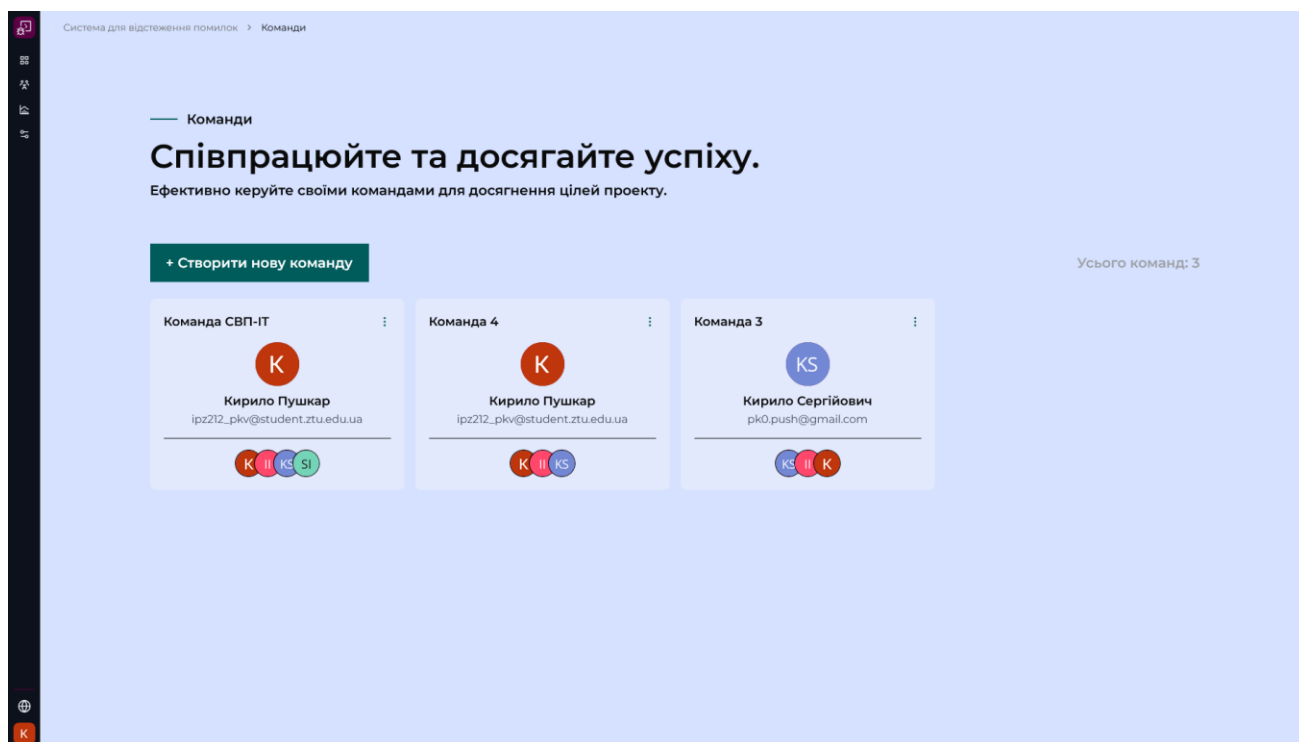


Рисунок 3.6. – Сторінка «Команди»

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

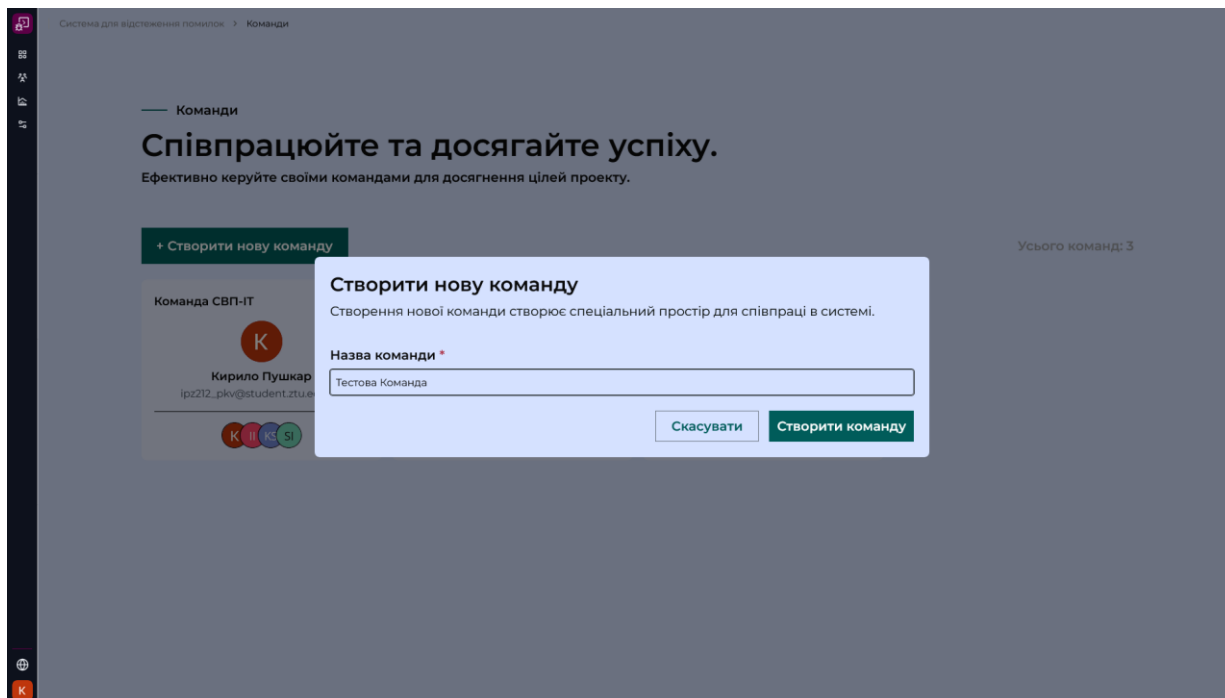


Рисунок 3.7. – Створення нової команди

На сторінці управління командами зосереджена вся необхідна функціональність для організації спільної роботи над проєктами: тут можна швидко створити, налаштувати та адмініструвати команди, забезпечуючи злагоджену взаємодію.

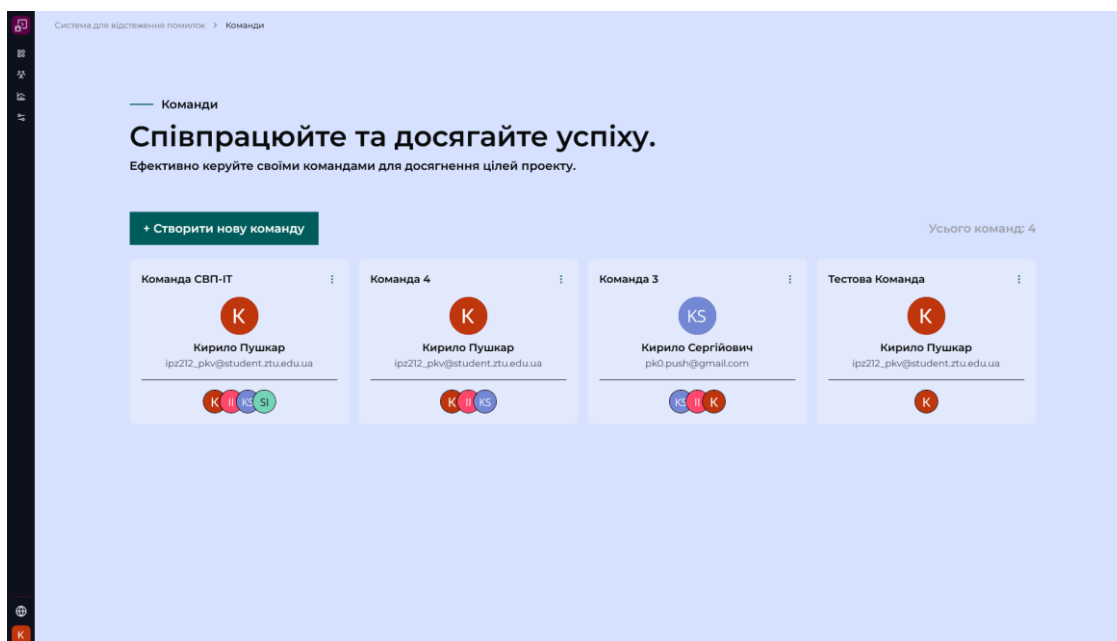


Рисунок 3.8. – Створення нової команди

Після успішного створення або вибору вже існуючої команди, користувач може перейти до детального управління її учасниками. Для цього передбачений

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

окремий розділ, який може бути доступний через клік на картці команди або через бічне меню.

На сторінці управління учасниками відображається повний список всіх, хто належить до даної команди. Кожен учасник представлений у вигляді окремої картки.

Крім того, на цій сторінці користувач з відповідними правами може видаляти учасників з команди за допомогою кнопки "Вилучити" (рис. 3.9), а також запрошувати нових учасників (рис. 3.10).

Для запрошення нового учасника користувач натискає кнопку "Запросити нового учасника". Після цього відкривається модальне вікно, де необхідно ввести електронну адресу користувача, якого хочуть запросити. Система перевіряє наявність такого користувача в базі даних.

У разі успішного запрошення, новий учасник з'являється у списку команди, а система сповіщає про успішне запрошення (рис. 3.11).

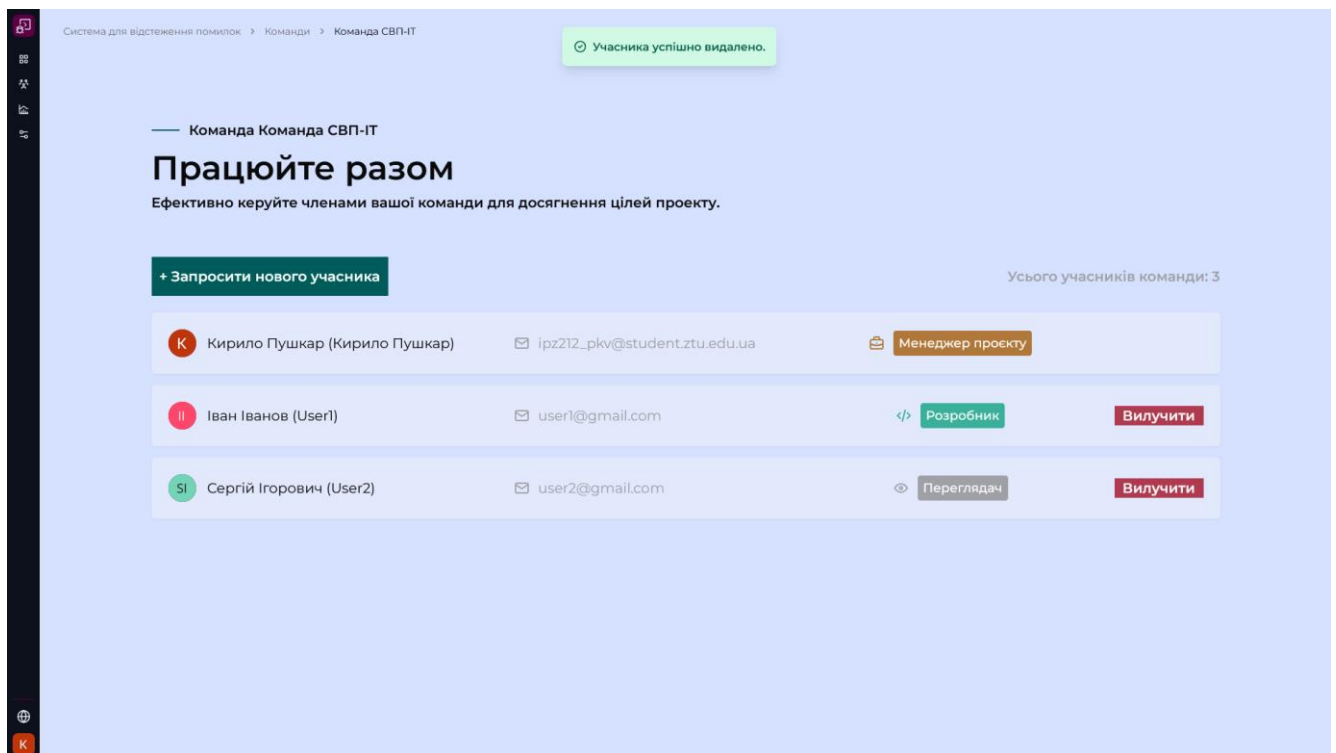


Рисунок 3.9 – Вилучення учасника команди

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

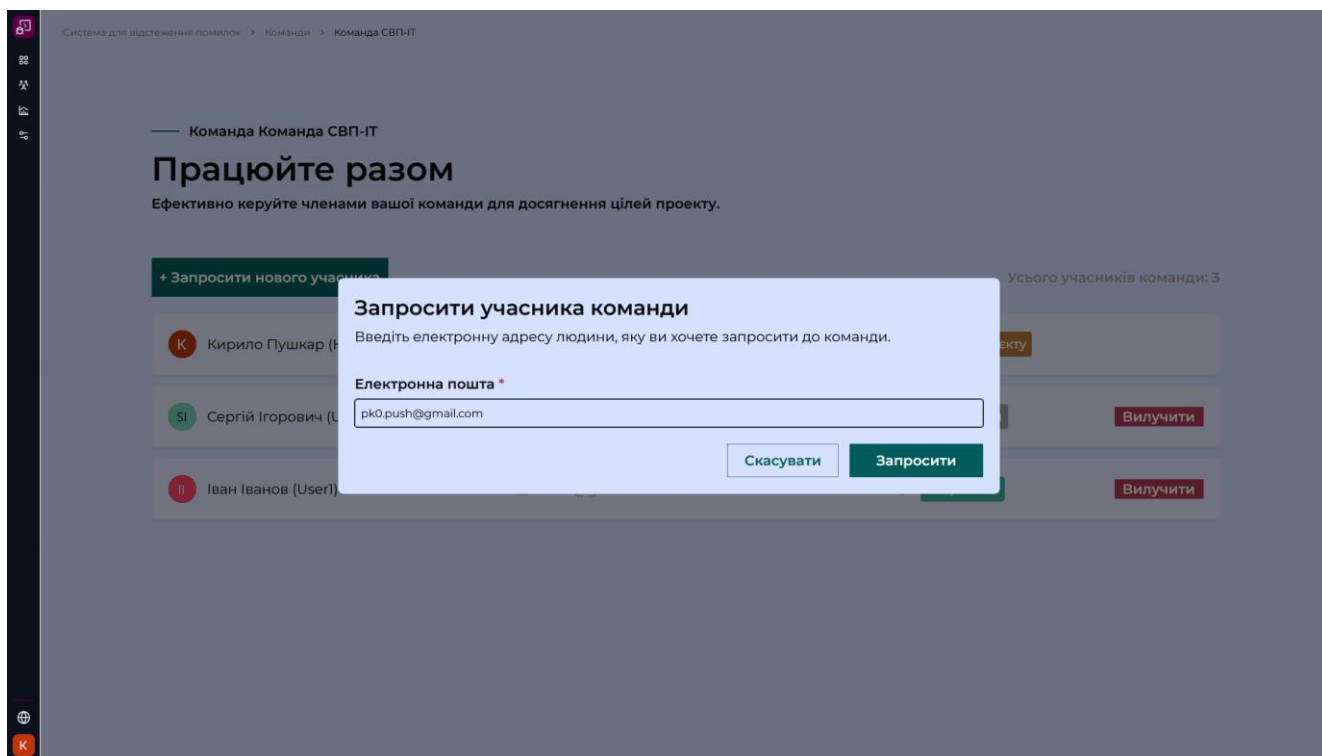


Рисунок 3.10 – Запрошення нового учасника команди

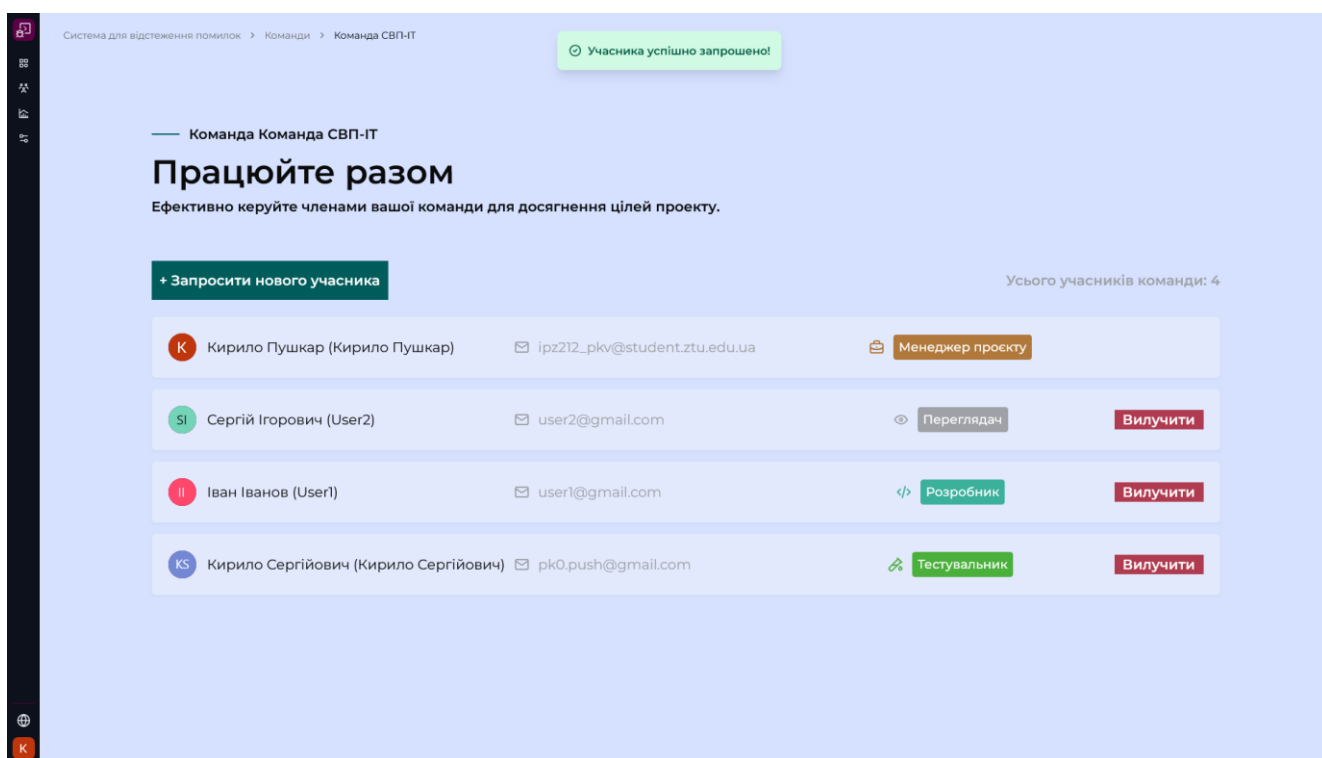


Рисунок 3.11 – Запрошення нового учасника команди

Користувач має особисту сторінку профілю, доступ до якої здійснюється через бічне меню (рис. 3.12). На головній вкладці профілю він може редагувати свої дані, а також переглядати записи про помилки, що пов'язані з ним.

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

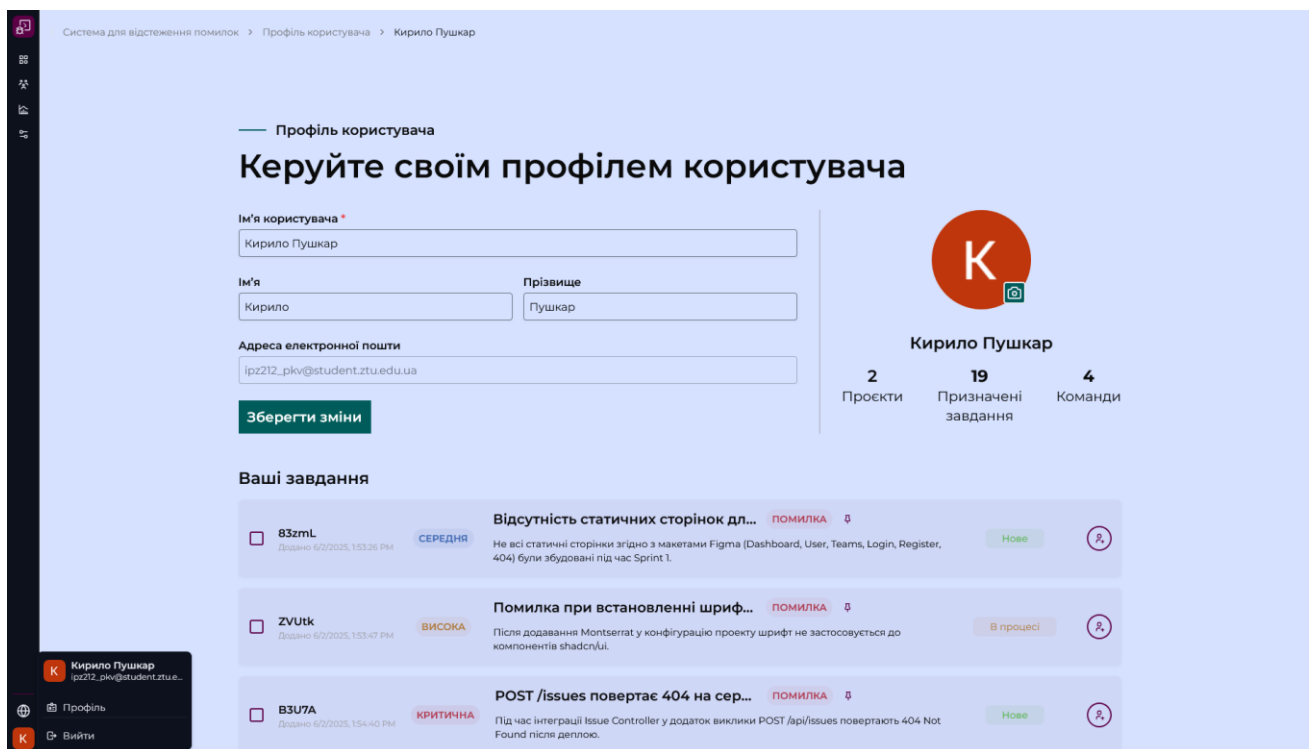


Рисунок 3.12 – Сторінка «Профіль користувача»

Через бічне меню користувач може перейти до сторінки "Проекти" (рис. 3.13). Ця сторінка слугує головним центром керування всіма проектами, в яких користувач задіяний або до яких має доступ. Основне призначення сторінки — забезпечити швидкий доступ та зручні інструменти для роботи з проектами.

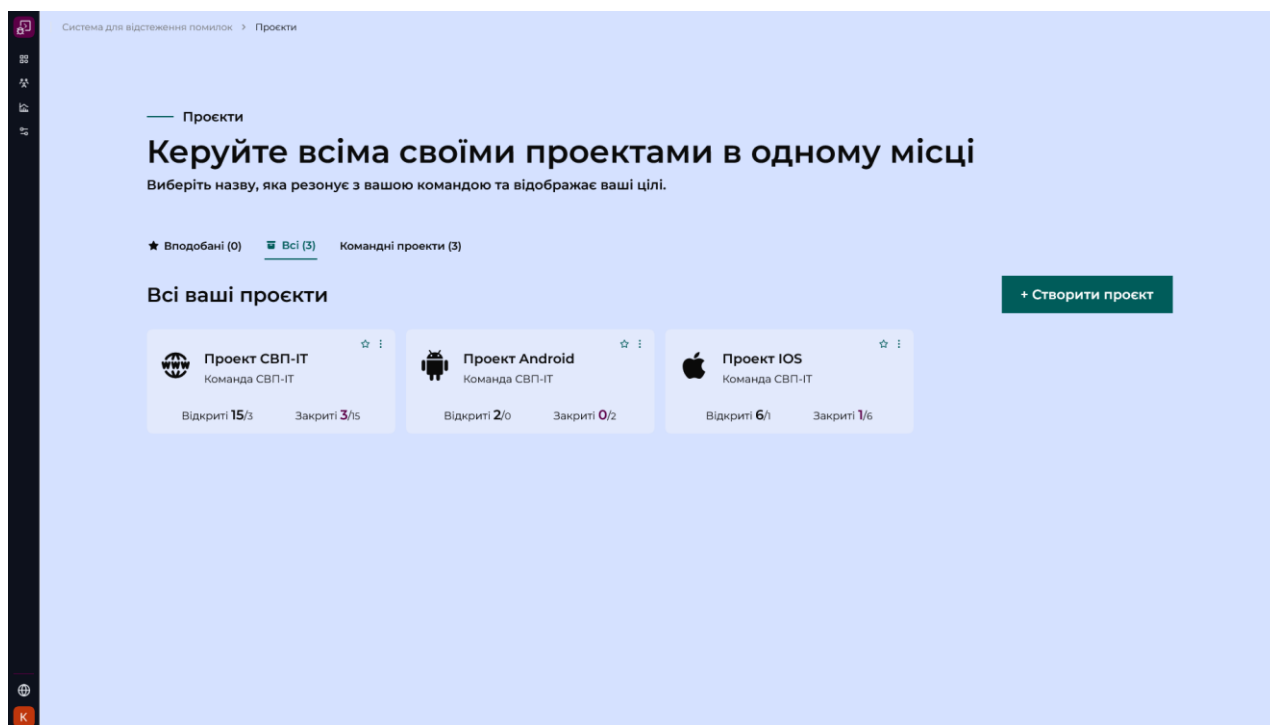


Рисунок 3.13 – Сторінка «Проекти»

Після натискання кнопки "Створити проєкт" (рис. 3.14) відкривається модальне вікно з формою для заповнення.

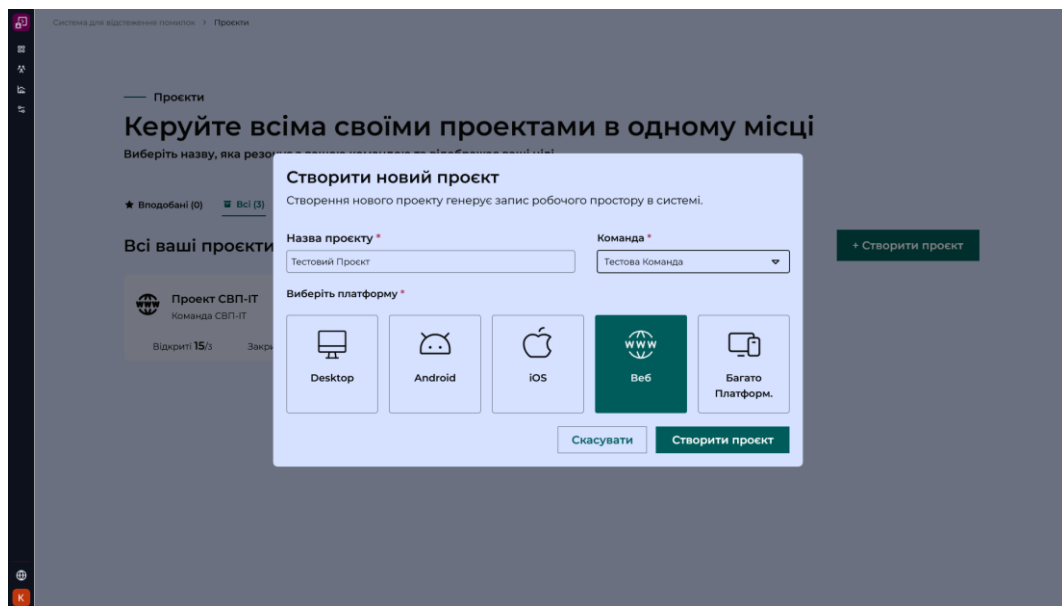


Рисунок 3.14 – Створення нового проєкту

У ньому передбачені такі поля: назва проєкту, вибір команди та вибір платформи. Користувач заповнює всі обов'язкові поля й натискає кнопку підтвердження створення (рис. 3.15).

Після обробки запиту система, у разі успіху, додає новий проєкт до списку на сторінці "Проекти" та інформує користувача відповідним повідомленням.

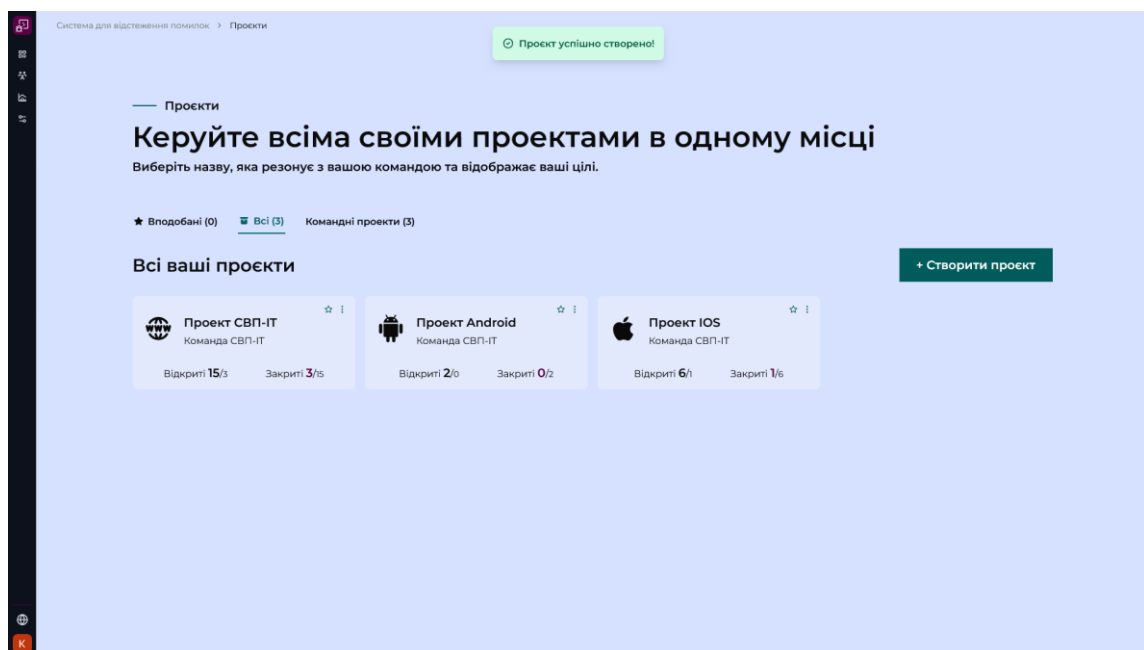


Рисунок 3.15 – Створення нового проєкту

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				46
Змн.	Арк.	№ докум.	Підпис	Дата		

Після вибору конкретного проєкту (кліком на його картку), користувач переходить на сторінку "Помилки" цього проєкту, де він зможе детально переглядати, створювати та керувати всіма записами, пов'язаними з обраним проєктом.

Сторінка «Помилки» містить заголовок, опис, кнопки для створення нової помилки та експорту списку. Далі розміщені статистичні картки зі статусами (загалом, в процесі тощо), які фільтрують список при натисканні. Основну частину займає список інтерактивних карток помилок. Над ним — панель інструментів із полем пошуку за назвою чи описом, випадаючим списком для сортування (за датою, типом, статусом тощо) та кнопкою «Видалити», яка з'являється при виборі однієї чи кількох помилок (рис. 3.16, 3.17).

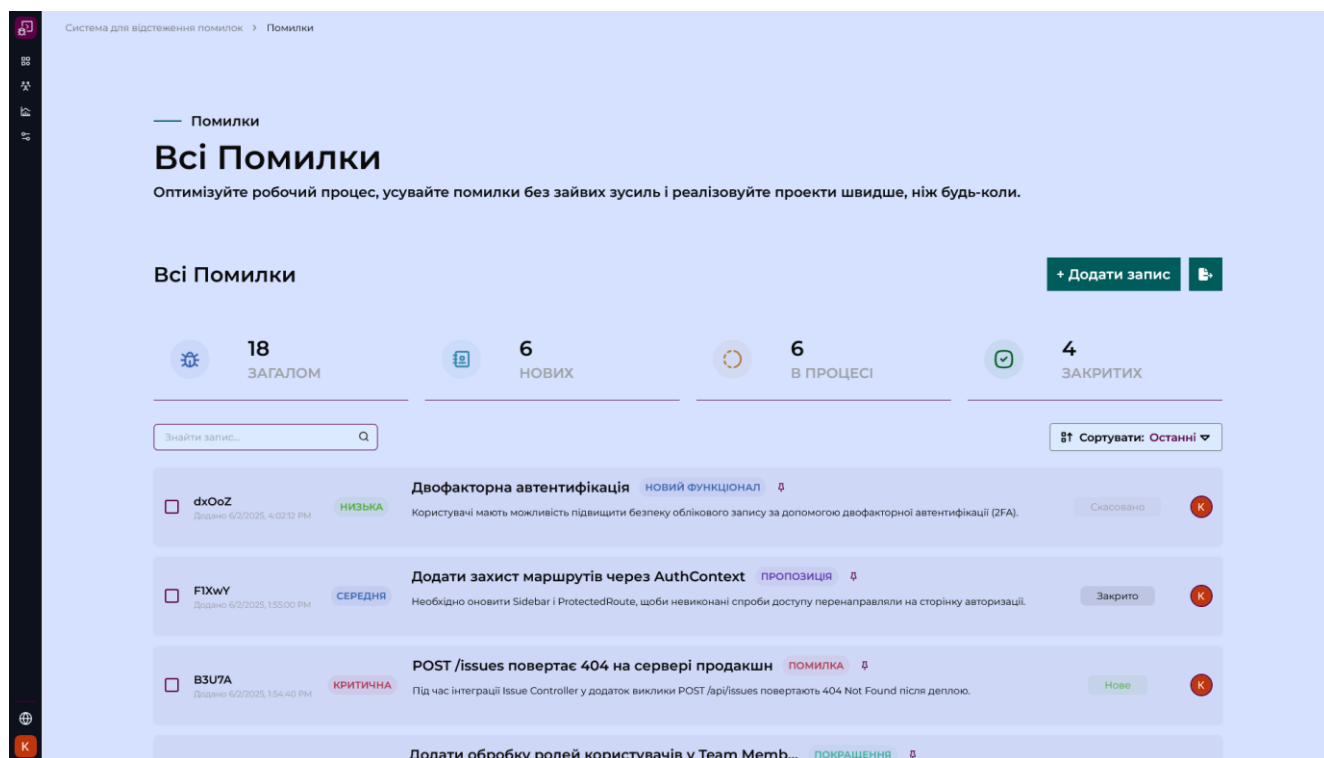


Рисунок 3.16 – Сторінка «Помилки»

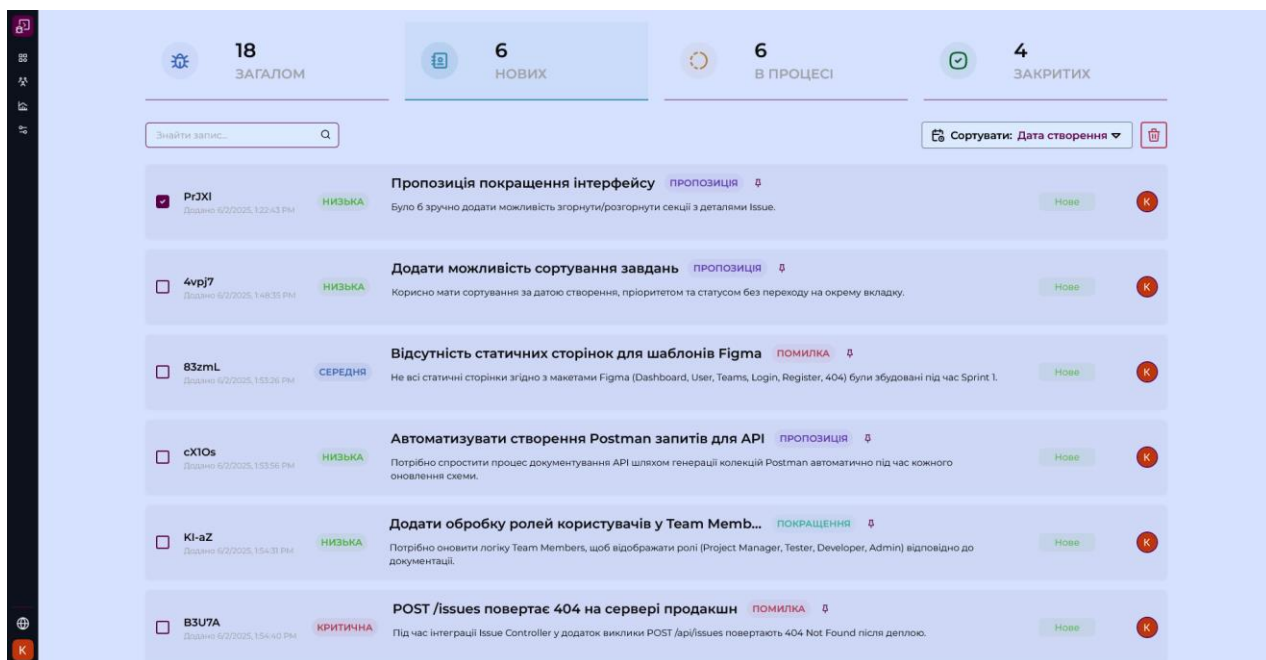


Рисунок 3.17 – Фільтрування та сортування помилок

Після натискання кнопки «Додати запис» відкривається модальне вікно, що складається з двох блоків. У лівій частині розташовані поля для введення заголовка, опису проблеми, технічних деталей та виконавця, якого можна обрати з випадаючого списку. Правий блок призначений для вибору класифікацій: типу помилки, рівня критичності та статусу. Після заповнення полів користувач натискає кнопку «Створити запис». Процес створення нового запису про помилку проілюстровано на рисунках 3.18-3.21.

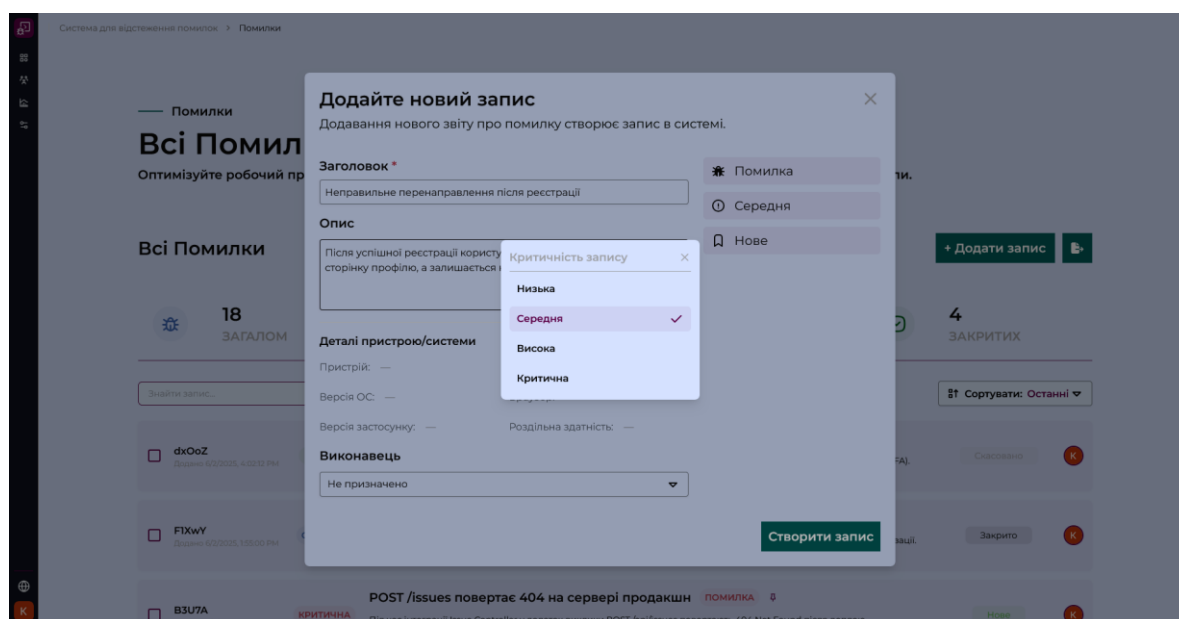


Рисунок 3.18 – Зміна критичності запису

У разі успішного створення з’являється відповідне повідомлення, в іншому випадку — повідомлення про помилку.

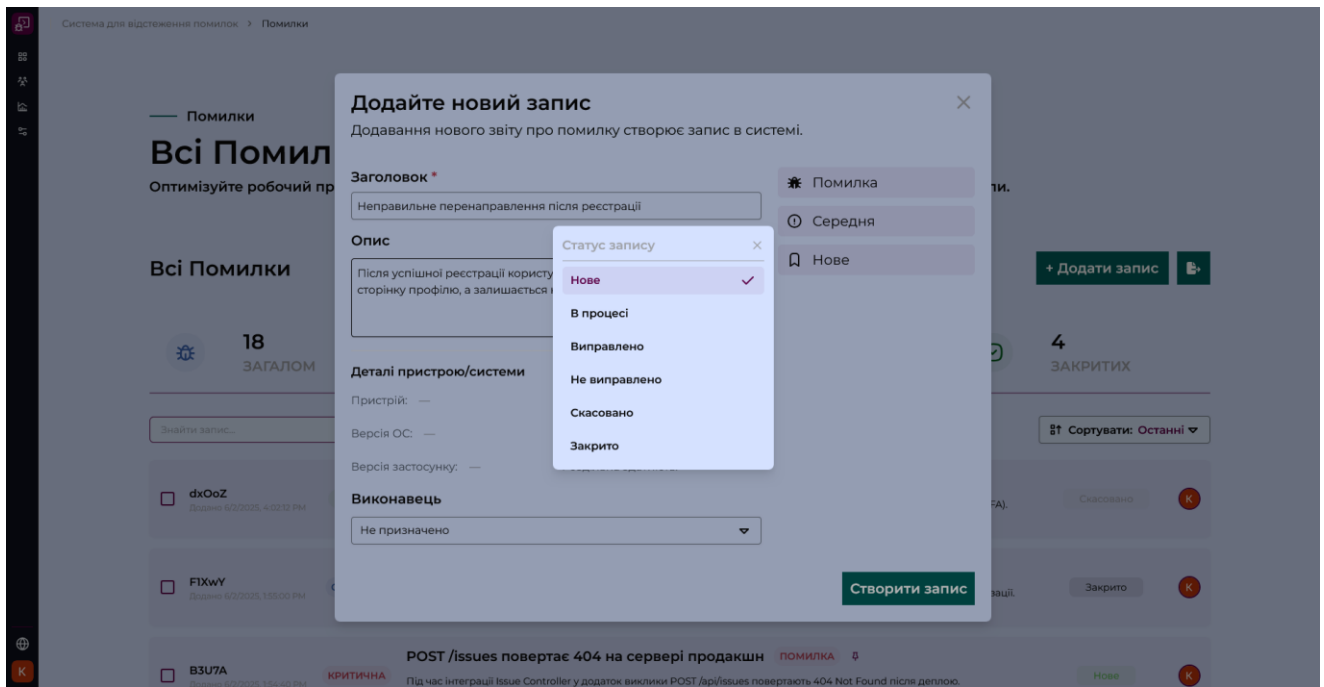


Рисунок 3.19 – Зміна статусу запису

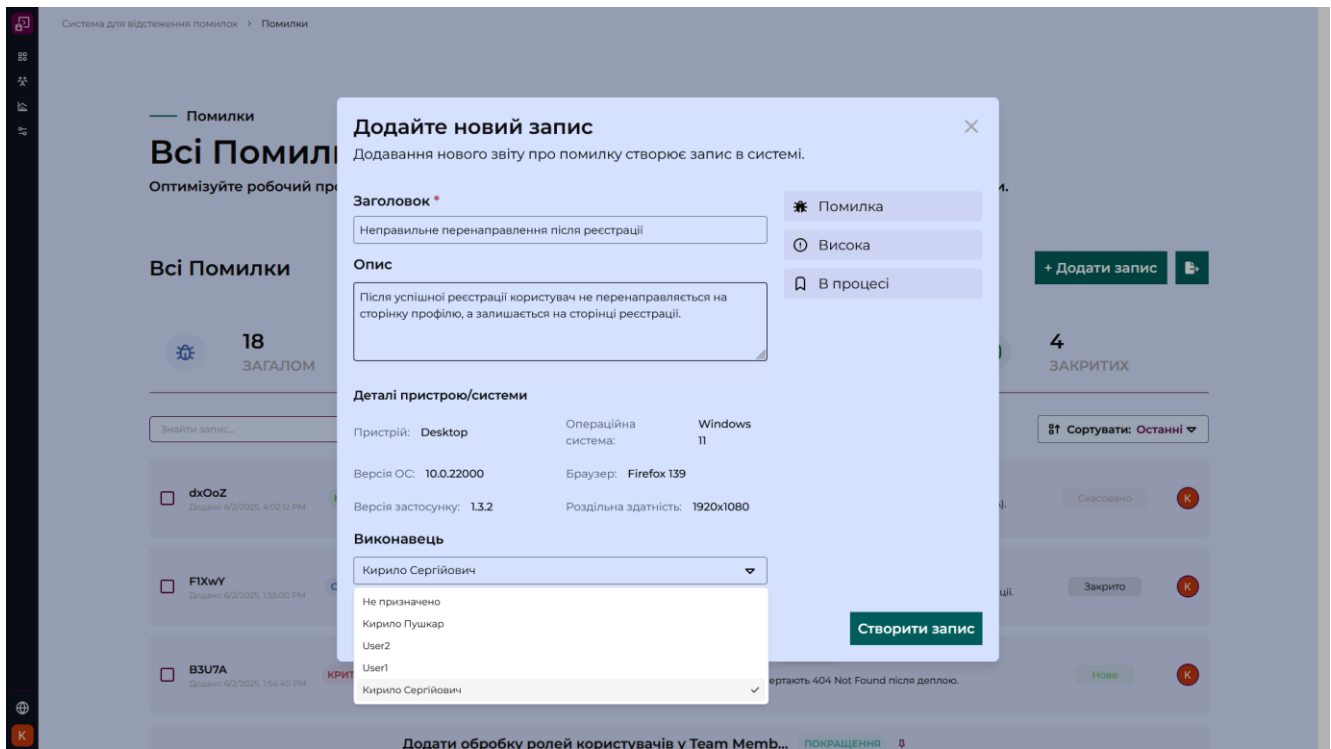


Рисунок 3.20 – Зміна деталей пристрою та виконавця

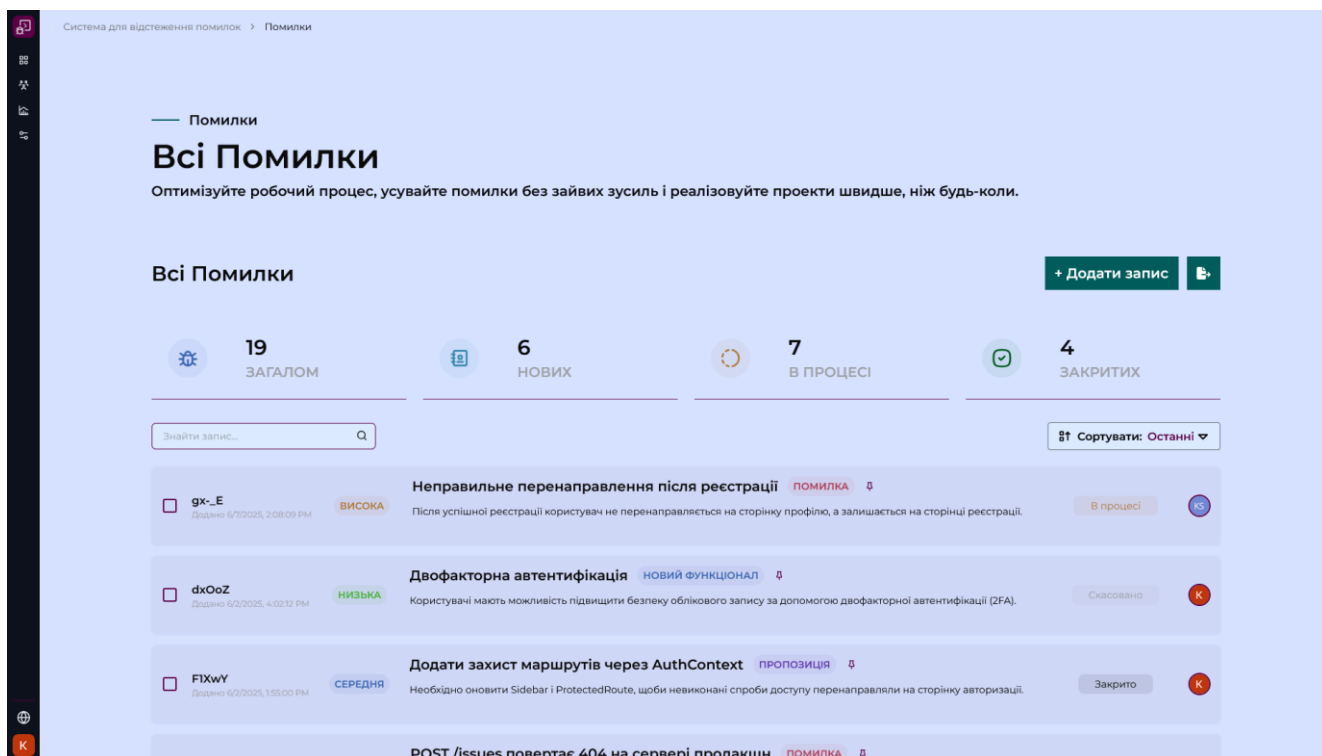


Рисунок 3.21 – Створення нового запису про помилку

Після створення або вибору існуючої помилки відкривається модальне вікно, яке дозволяє переглянути та редагувати детальну інформацію. У верхній частині відображається унікальний ідентифікатор помилки та її поточний статус. Користувач може змінити тип помилки, відкривши додаткове вікно для вибору значення. Нижче розташоване поле для детального опису проблеми, яке можна заповнювати або редагувати (рис. 3.22).

У секції вкладень показано список вкладень. Кожне вкладення — це посилання на файл (скріншот), яке відкривається в новій вкладці браузера при натисканні, дозволяючи переглянути візуальний доказ помилки. Користувач може додавати нові вкладення через кнопку вибору файлів, після чого файли автоматично завантажуються та з'являються у списку. Видалити непотрібні вкладення можна натиснувши кнопку «Видалити» поруч із ними (рис. 3.23).

Окрема секція відповідає за метадані: тут можна встановити статус, критичність, тип помилки та ввести технічні характеристики пристрою та виконавця.

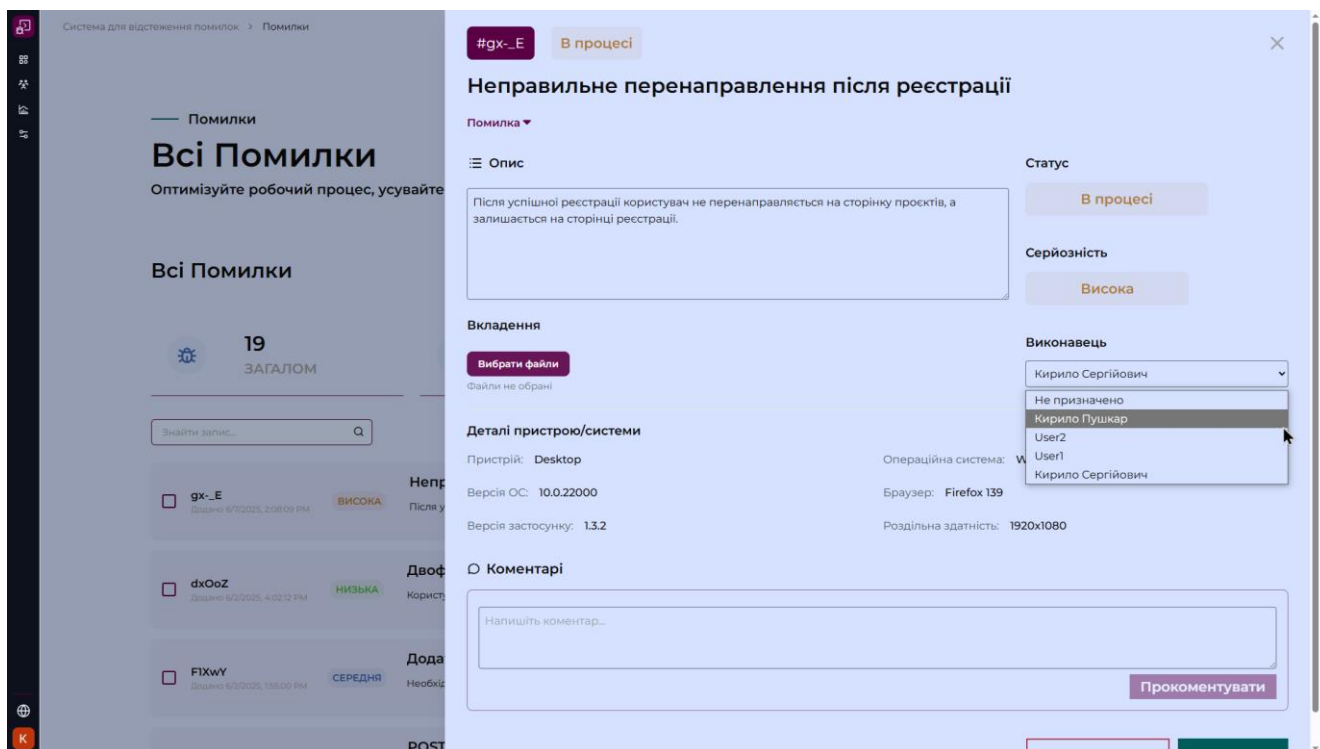


Рисунок 3.22 – Оновлення інформації про помилку

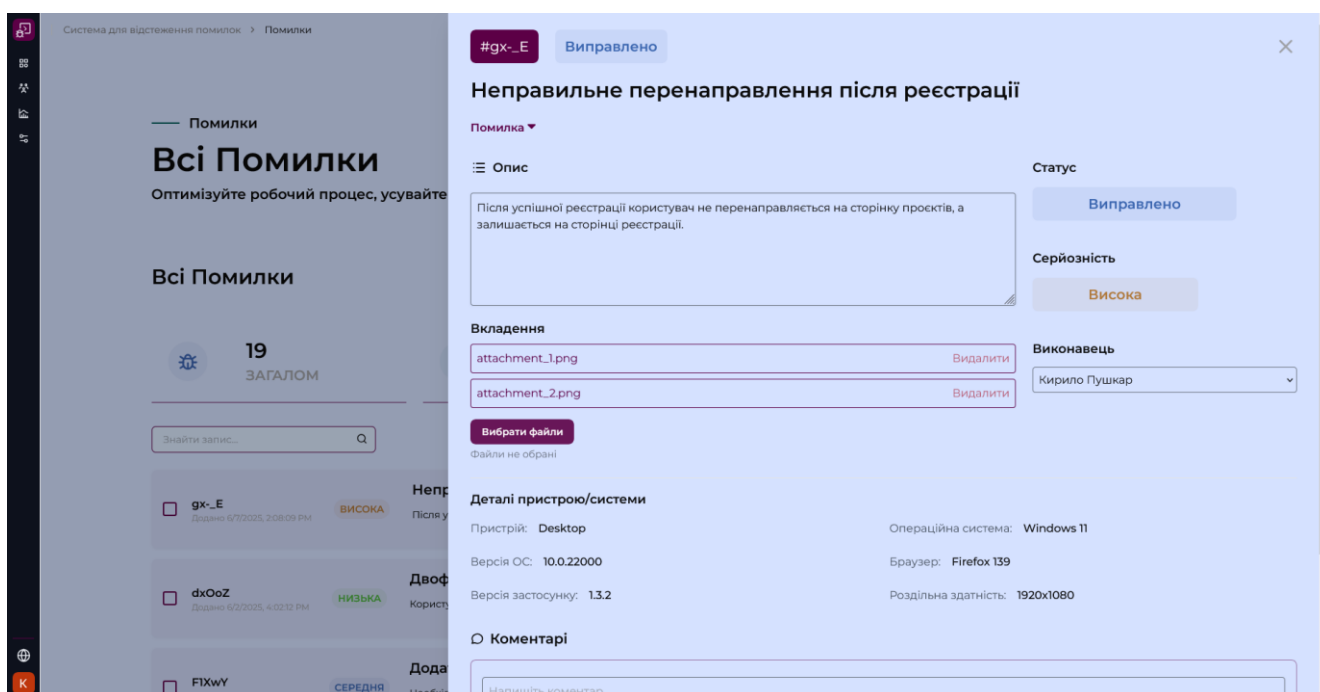


Рисунок 3.23 – Додавання нових вкладень до запису

У нижній частині модального вікна розміщено розділ коментарів. Користувач може залишати нові коментарі, редагувати або видаляти власні, а також відповідати на вже існуючі, формуючи ієрархічну структуру обговорень. Кожен коментар супроводжується ім'ям автора та міткою часу (рис. 3.24, 3.25).

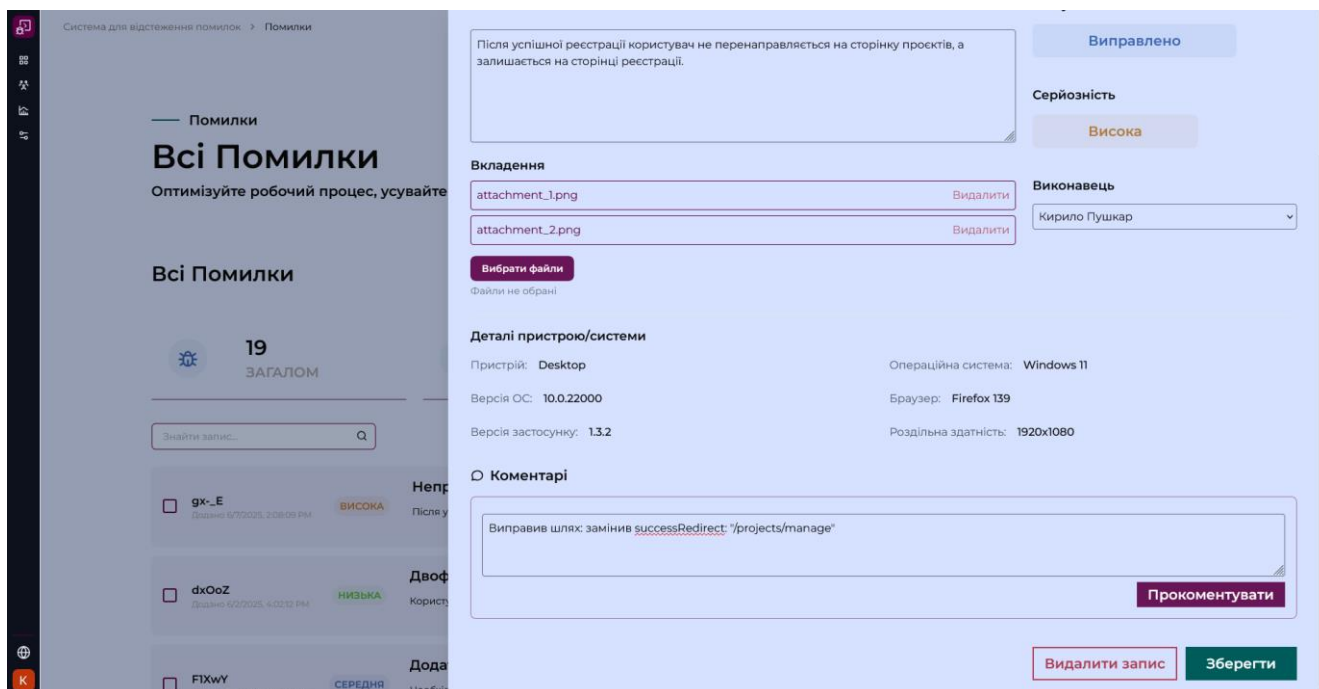


Рисунок 3.24 – Створення нового коментаря

Після будь-яких змін у записі користувач має можливість зберегти дані натисканням кнопки «Зберегти».

Якщо помилку потрібно видалити — доступна кнопка «Видалити запис», яка після підтвердження ініціює видалення з подальшим повідомленням про успішну дію або помилку.

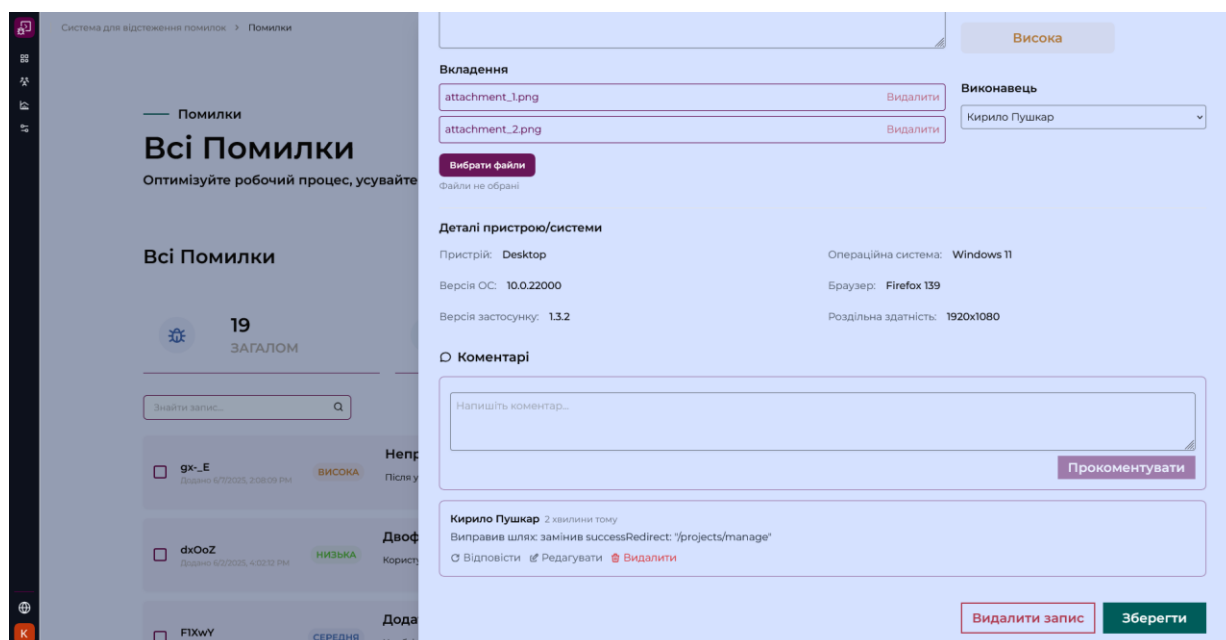


Рисунок 3.25 – Створення нового коментаря

Система відстеження помилок також надає розділ аналітики та статистики(рис. 3.26, 3.27), який дає змогу швидко оцінити стан проєктів і ефективність роботи команд. У верхній частині сторінки за допомогою випадаючого списку вибирається команда; якщо ж користувач не належить до жодної з них або команда не обрана, система відображає відповідне повідомлення.

Після вибору на екран виводяться три кругові діаграми: перша демонструє відсотковий розподіл помилок за рівнем критичності, друга ілюструє типи завдань, а третя показує їхні статуси. Далі розміщена теплова карта, яка відображає щоденну активність створення та оновлення завдань: інтенсивніші кольори вказують на пікові дні, а при наведенні курсору можна побачити точну кількість записів за кожен день. Внизу розташований гістограмний графік, де стовпці представляють проєкти й розбиті за статусами завдань.

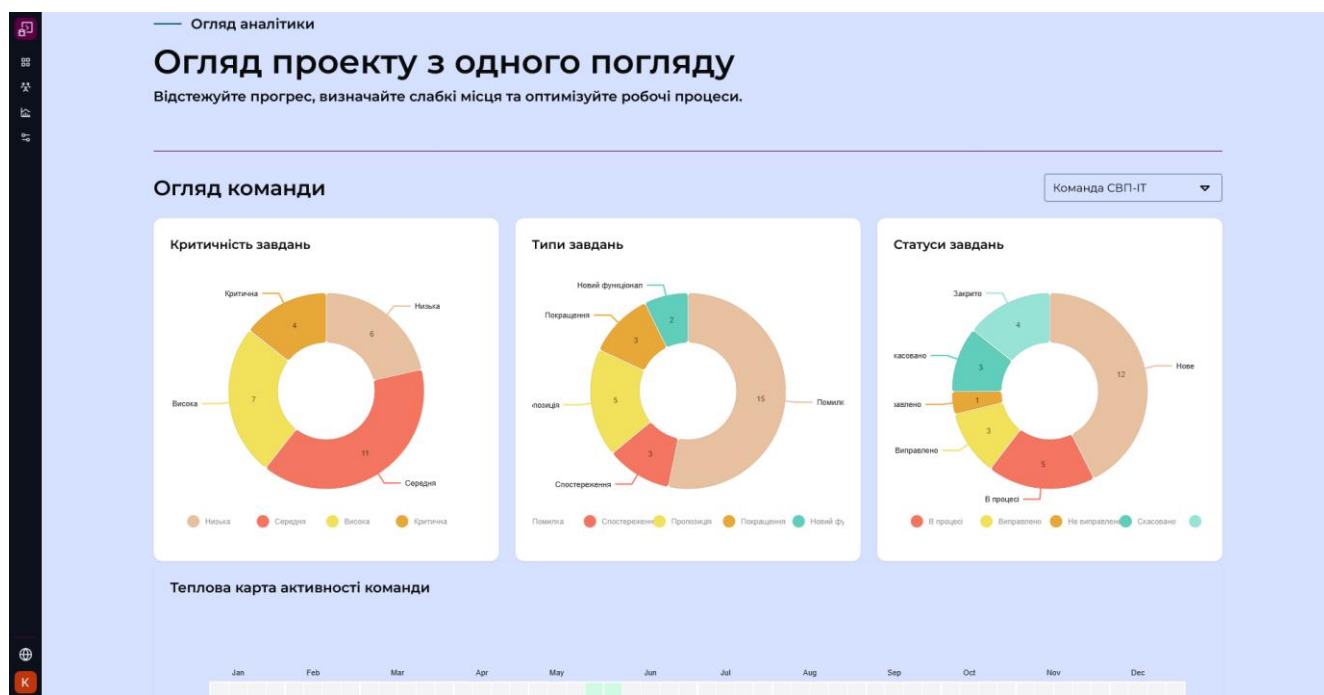


Рисунок 3.26 – Сторінка «Огляд аналітики»

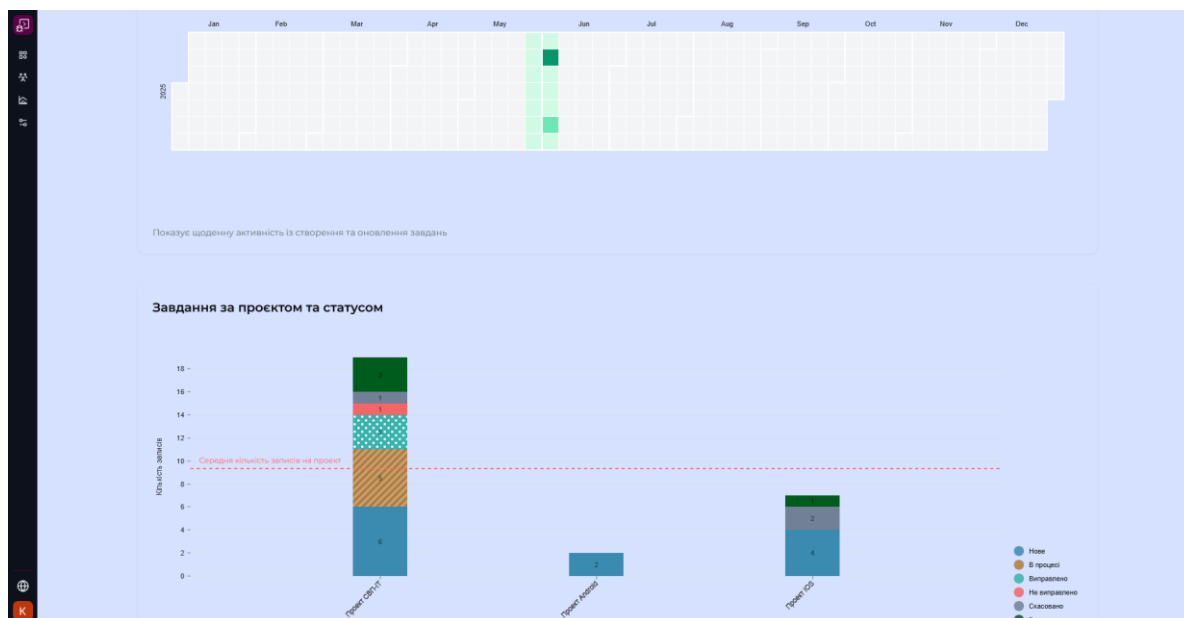


Рисунок 3.27 – Сторінка «Огляд аналітики»

Система відстеження помилок, окрім функцій аналітики, надає можливість зручного експорту даних. Це дозволяє користувачам зберігати інформацію про помилки у популярних форматах для подальшого аналізу або перенесення в Jira. Кнопка «Експорт» доступна на сторінці огляду помилок (рис. 3.28). Після кліку з’являється меню з трьома варіантами: завантажити дані у вигляді CSV, у вигляді JSON або безпосередньо експортувати до Jira.

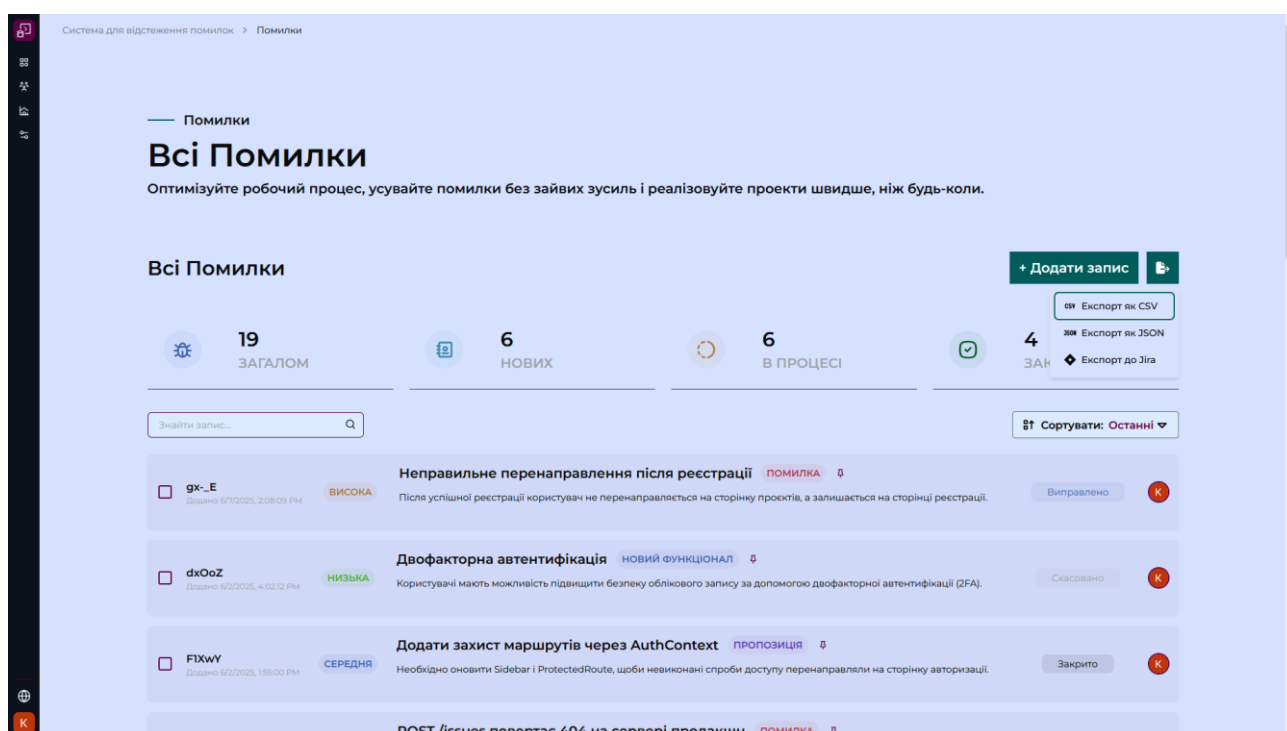


Рисунок 3.28 – Меню експорту

Під опцією CSV формується файл з роздільниками-комами — ідеальний для імпорту в табличні редактори (Excel, Google Sheets) та подальшого аналізу (рис. 3.29). Вибір JSON дає змогу отримати структурований документ у форматі, зручному для обміну між вебсервісами та автоматизації обробки (рис. 3.30). Опція «Експорт до Jira» надсилає звіти прямо в обрану систему керування задачами. ,

Issue ID	Reported Date	Bug Summary	Description	Build Ver	Severity	Issue Status	Device	OS Ver	Type	Tester	Assignee
2	2025-06-02T10:19:50.631Z	Помилка завантаження списку завдань	При спробі відобразити список завдань на сторінці Issue виникає	1.2.2003	Critical	Fixed	Desktop	10.0.19042	Bug	Кирило Пушкар	Кирило Пушкар
3	2025-06-02T10:22:43.500Z	Пропозиція покращення інтерфейсу	Було б зручно додати можливість згорнути/розгорнути секції з д	1.2.2003	Low	New	Desktop	11.4	Suggestion	Кирило Пушкар	Кирило Пушкар
4	2025-06-02T10:53:26.597Z	Відсутність статичних сторінок для шаблону Figma	Не всі статичні сторінки збігаються з макетами Figma (Dashboard, Us	1.3.2000	Medium	New	Desktop	10.0.19042	Bug	Кирило Пушкар	Кирило Пушкар
5	2025-06-02T10:53:47.718Z	Помилка при встановленні шрифтів за замовчуванням	Після додавання Montserrat у конфігурацію проекту шрифти не за	1.3.2001	High	InProgress	Laptop	11.4	Bug	Кирило Пушкар	Кирило Пушкар
6	2025-06-02T10:54:40.109Z	POST /issues повертає 404 на сервері продакшн	Під час інтеракції Issue Controller у додаток викликає POST /api/le	1.3.2000	Critical	New	Server	20.04	Bug	Кирило Пушкар	Кирило Пушкар
7	2025-06-02T10:48:35.846Z	Додати можливість сортування завдань	Корисно мати сортування за датою створення, пріоритетом та с	1.2.2003	Low	New	Smartphone	10	Suggestion	Кирило Пушкар	Кирило Пушкар
8	2025-06-02T10:50:19.382Z	Оновити залежності до останніх версій	Бібліотека Tailwind застаріла, необхідно оновити до версії 4.x і п	1.2.2000	Medium	Closed	Laptop	10.15.7	Improvement	Кирило Пушкар	Кирило Пушкар
9	2025-06-07T11:08:09.361Z	Неправильне перенаправлення після реєстрації	Після успішної реєстрації користувач не перенаправляється на	1.3.2002	High	Fixed	Desktop	10.0.22000	Bug	Кирило Пушкар	Кирило Пушкар
10	2025-06-02T10:22:53.641Z	Повільний відгук сервера	Після створення нового Issue відповідь API займає понад 5 сек	1.2.2003	High	InProgress	Desktop	20.04	Observation	Кирило Пушкар	Кирило Пушкар
11	2025-06-02T10:23:16.576Z	Втрата даних при оновленні запису	При оновленні Issue через PUT /issues/id деякі поля обнулюють	1.2.2003	Critical	InProgress	iPhone 12	14.5	Bug	Кирило Пушкар	Кирило Пушкар
12	2025-06-02T10:23:09.492Z	Оптимізувати запити до бази даних	Запити GET /issues на індексованому полі виконуються занадто	1.2.2003	Medium	Closed	Desktop	10.0.19042	Improvement	Кирило Пушкар	Кирило Пушкар
13	2025-06-02T10:23:25.248Z	Неправильний формат дати	При створенні Issue з датою у форматі DD-MM-YYYY сервер по	1.2.2002	Medium	Fixed	Desktop	20.04	Bug	Кирило Пушкар	Кирило Пушкар
14	2025-06-02T10:53:56.871Z	Автоматизувати створення Postman запитів для	Потрібно спростити процес документування API шляхом генера	1.3.2000	Low	New	Desktop	20.04	Suggestion	Кирило Пушкар	Кирило Пушкар
15	2025-06-02T10:54:08.372Z	Немає обробки помилок при генерації CSV експ	Під час виклику експорту /export/csv відсутня перевірка на пом	1.3.2001	High	InProgress	Desktop	6.3	Observation	Кирило Пушкар	Кирило Пушкар
16	2025-06-02T10:54:21.389Z	Додавання компоненту Sidebar Navigation	Потрібно реалізувати Sidebar Navigation за макетом Figma зі зго	1.3.2001	Medium	InProgress	iPad Pro	14.6	NewFeature	Кирило Пушкар	Кирило Пушкар
17	2025-06-02T10:54:31.477Z	Додати обробку ролей користувачів у Team Mem	Потрібно оновити ролю Team Members, щоб відображати ролю (	1.3.2000	Low	New	Smartphone	11	Improvement	Кирило Пушкар	Кирило Пушкар
18	2025-06-02T10:54:38.712Z	Двофакторна аутентифікація	Користувачі мають можливість підвищити безпеку облікового за	1.2.2004	Low	Cancelled	Smartphone	11	NewFeature	Кирило Пушкар	Кирило Пушкар
19	2025-06-02T10:55:00.743Z	Додати захист маршрутів через AuthContext	Необхідно оновити Sidebar / ProtectedRoute, щоб невиконані ст	1.3.2000	Medium	Closed	Smartphone	10	Suggestion	Кирило Пушкар	Кирило Пушкар
20	2025-06-02T10:48:24.835Z	Незрозумілі повідомлення про помилки	Коди API повертає помилку, повідомлення не містять деталей (i	1.2.2001	Low	NotFixed	Desktop	6.3	Observation	Кирило Пушкар	Кирило Пушкар

Рисунок 3.29 – Приклад експортування даних в форматі CSV

```

{
  "username": "Кирило Пушкар",
  "email": "Кирило Пушкар",
  "attachment": [
    {
      "id": 46,
      "visibleId": "gk_E",
      "project": 1,
      "reporterId": 1,
      "assignedTo": 1,
      "title": "Неправильне перенаправлення після реєстрації",
      "description": "Після успішної реєстрації користувач не перенаправляється на сторінку проекту, а залишається на сторінці реєстрації.",
      "severity": "High",
      "type": "Bug",
      "status": "Fixed",
      "isPinned": false,
      "createdAt": "2025-06-07T11:08:09.361Z",
      "updatedAt": "2025-06-07T11:04:04.489Z",
      "operatingSystem": "Windows 11",
      "version": "10.0.22000",
      "deviceType": "Desktop",
      "browser": "Firefox 119",
      "screenResolution": "1920x1080",
      "appVersion": "1.1.2",
      "project": {
        "name": "Проект GBN-IT"
      },
      "reporter": {
        "username": "Кирило Пушкар"
      },
      "assigned": {
        "username": "Кирило Пушкар"
      },
      "attachment": [
        {
          "file": {
            "filename": "attachment_1.png",
            "url": "https://storage.cloud.google.com/bugtracker-attachments/76038672001-vj555w-attachment_1.png?authuser=1"
          },
          "id": 1,
          "filename": "attachment_1.png",
          "url": "https://storage.cloud.google.com/bugtracker-attachments/76038672001-vj555w-attachment_1.png?authuser=1"
        }
      ]
    },
    {
      "id": 48,
      "visibleId": "RTmp",
      "project": 1,
      "reporterId": 1,
      "assignedTo": 1,
      "title": "Повільний відгук сервера",
      "description": "Після створення нового Issue відповідь API займає понад 5 секунд.",
      "severity": "High"
    }
  ]
}

```

Рисунок 3.30 – Приклад експортування даних в форматі JSON

Розглянемо приклад експорту даних проекту до Jira. Для цього спочатку створимо новий проект у Jira, обравши шаблон «Bug Tracking».

Послідовність створення нового проєкту демонструється на рисунках 3.31 та 3.32.

Рисунок 3.31 – Створення нового проєкту Jira

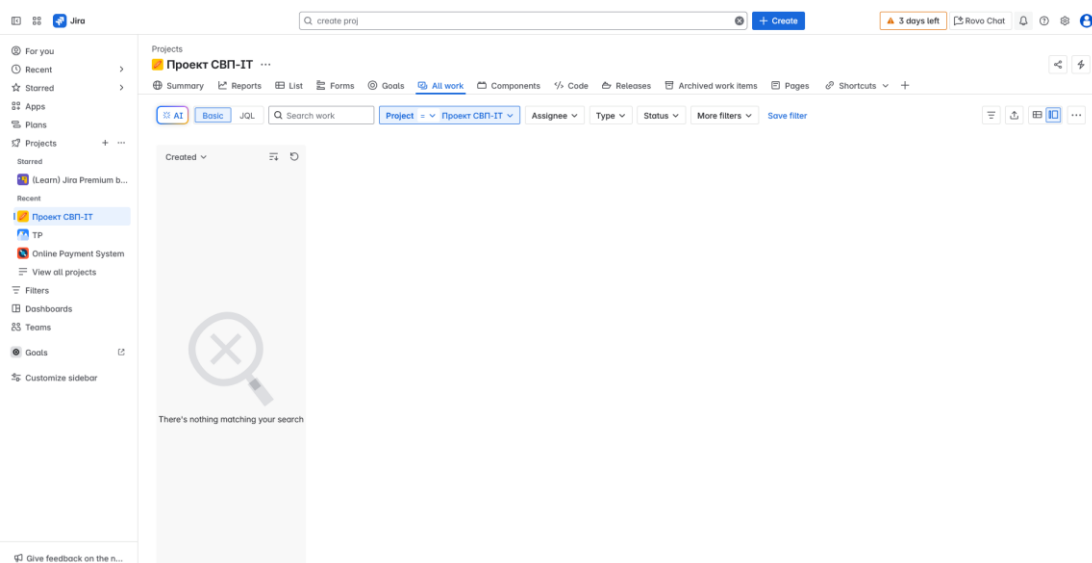


Рисунок 3.32 – Створення нового проєкту Jira

Тепер перейдемо до демонстрації процесу експорту даних. Перед тим як розпочати, користувач повинен під'єднати систему до свого акаунта в Jira. Для цього необхідно ввести кілька ключових параметрів: адресу сервера Jira, електронну пошту, що використовується для входу, та токен доступу. Після

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				56
Змн.	Арк.	№ докум.	Підпис	Дата		

заповнення цієї інформації достатньо натиснути кнопку «Далі» — система автоматично ініціює з'єднання з обліковим записом користувача та завантажить перелік доступних проєктів у Jira (рис. 3.33).

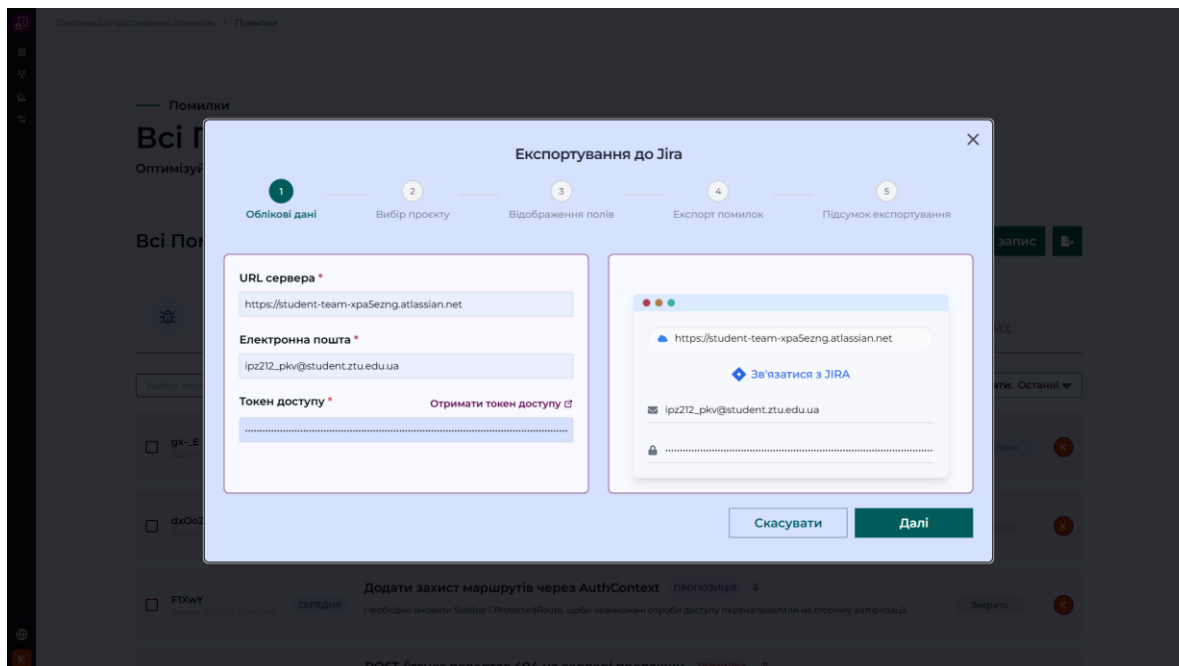


Рисунок 3.33 – Введення параметрів підключення до Jira та завантаження списку проєктів

Після успішного з'єднання з обліковим записом Jira користувачеві відображається перелік доступних проєктів. На цьому етапі потрібно обрати той, до якого буде здійснено експорт даних (рис. 3.34).

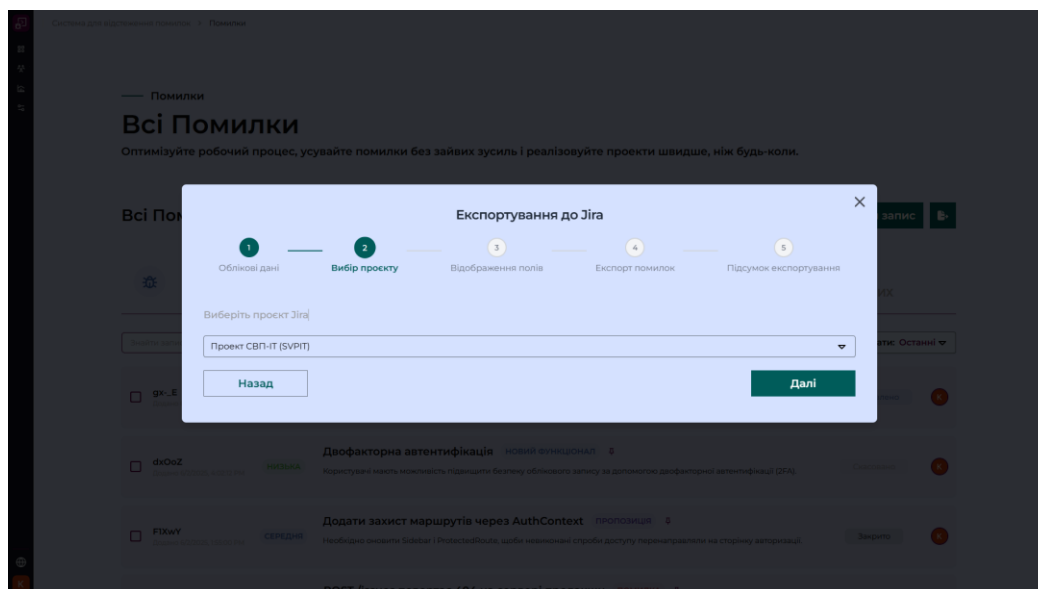


Рисунок 3.34 – Вибір проєкту Jira для експорту даних

На наступному етапі користувач має змогу ознайомитися з тим, як дані з його проекту зіставляються з відповідними полями в системі Jira. Це дозволяє впевнитися, що передача інформації відбудеться коректно і без втрат важливих даних (рис. 3.35). На рисунку 3.36 показано етап підтвердження експорту.

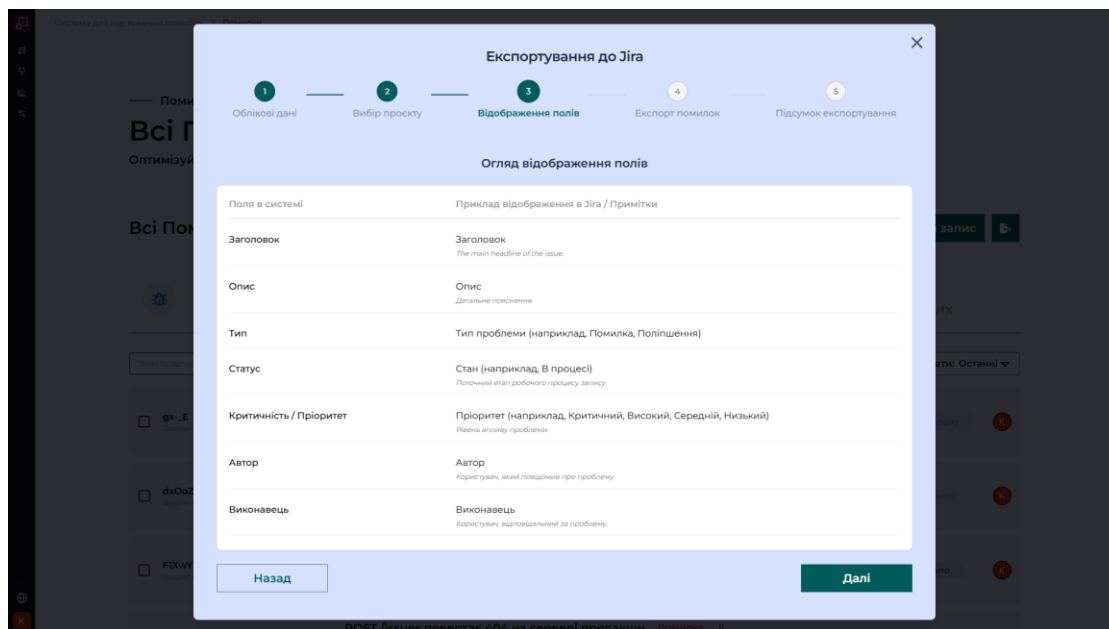


Рисунок 3.35 – Попередній перегляд відповідності полів між системою та Jira

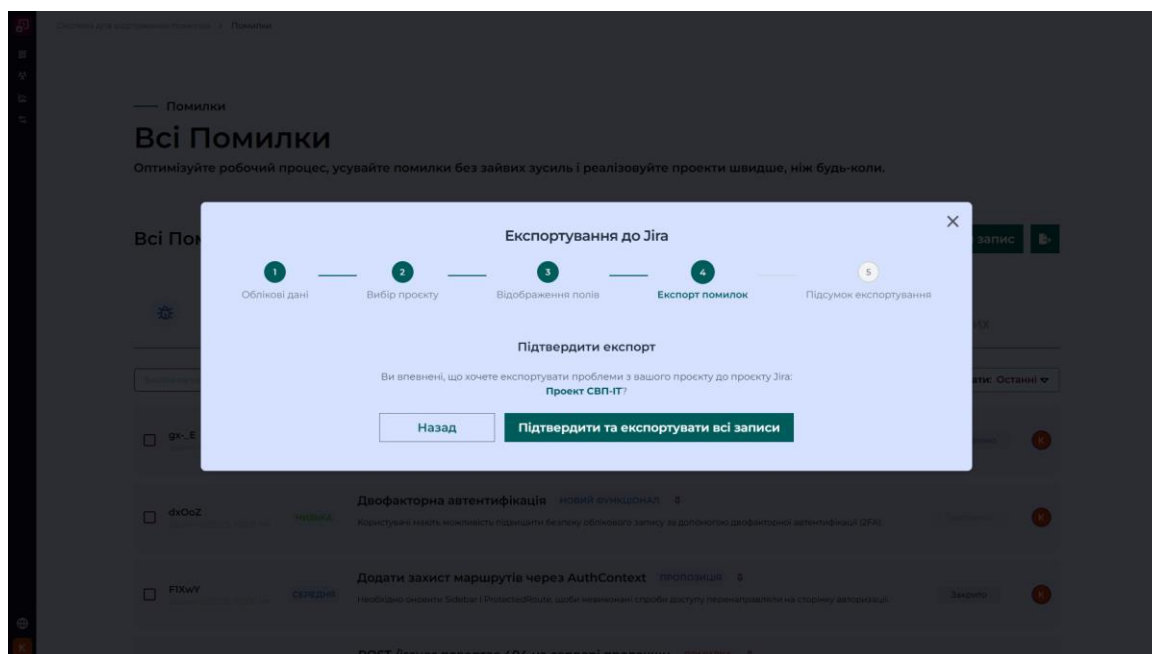


Рисунок 3.36 – Підтвердження експорту даних до Jira

Після завершення експорту система формує підсумковий звіт, який відображає кількість успішно перенесених задач, а також ті, що не вдалося експортувати. Для кожної успішно створеної задачі вказується її ключ, унікальний

ідентифікатор та активне посилання на відповідний запис у Jira. Якщо під час експорту виникли помилки, система зазначить їх кількість і надасть опис кожної з них. Крім того, у звіті можуть міститися попередження щодо задач, що були перенесені з окремими неточностями, як-от некоректний статус або відсутність виконавця. Результат експортування показано на рисунках 3.37 – 3.39.

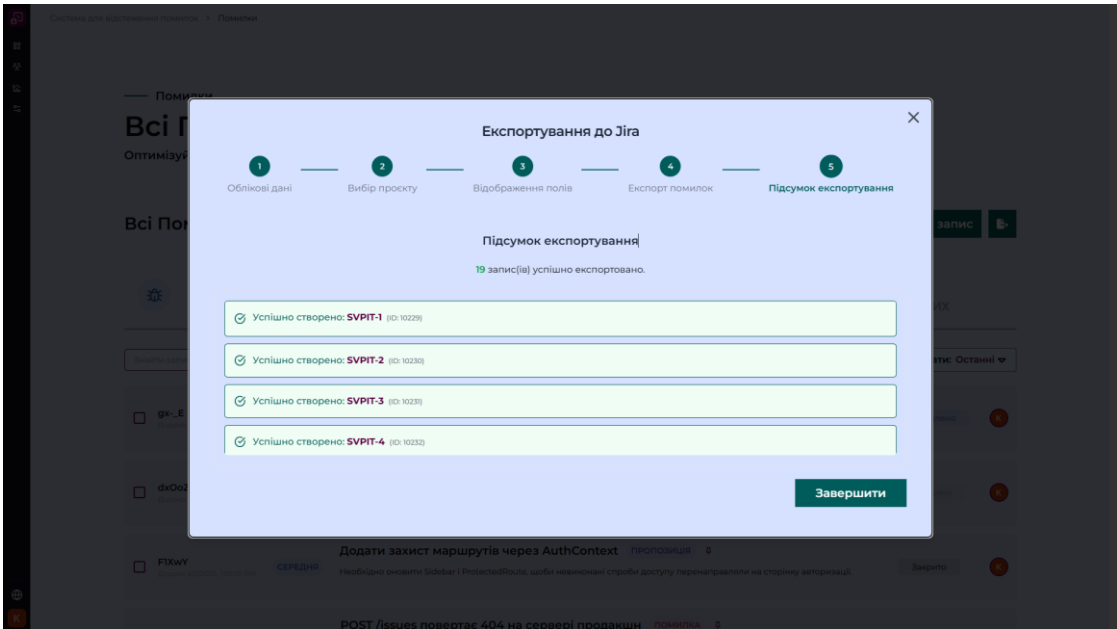


Рисунок 3.37 – Підсумок про успішне завершення експорту

Q Search list  Filter  Group  										
☐	Type	Key	Summary	Status	Comments	Priority	Created	Updated	Reporter	
☐	❌	SVPIT-1	Помилка завантаження списку завдань	TO DO	 Add comment	⬆	 Jun 9, 2025	 Jun 9, 2025	 Кирило Пушкар	
☐	✅	SVPIT-2	Пропозиція покращення інтерфейсу	TO DO	 Add comment	⬇	 Jun 9, 2025	 Jun 9, 2025	 Кирило Пушкар	
☐	❌	SVPIT-3	Відсутність статичних сторінок для шаблонів Figma	TO DO	 Add comment	=	 Jun 9, 2025	 Jun 9, 2025	 Кирило Пушкар	
☐	❌	SVPIT-4	Помилка при встановленні шрифту за замовчуванням (M...	IN PROGRESS	 Add comment	⬆	 Jun 9, 2025	 Jun 9, 2025	 Кирило Пушкар	
☐	❌	SVPIT-5	POST /issues повертає 404 на сервері продакшн	TO DO	 Add comment	⬆	 Jun 9, 2025	 Jun 9, 2025	 Кирило Пушкар	
☐	✅	SVPIT-6	Додати можливість сортування завдань	TO DO	 Add comment	⬇	 Jun 9, 2025	 Jun 9, 2025	 Кирило Пушкар	
☐	⬆	SVPIT-7	Оновити залежності до останніх версій	TO DO	 Add comment	=	 Jun 9, 2025	 Jun 9, 2025	 Кирило Пушкар	
☐	❌	SVPIT-8	Неправильне перенаправлення після реєстрації	TO DO	 Add comment	⬆	 Jun 9, 2025	 Jun 9, 2025	 Кирило Пушкар	
☐	✅	SVPIT-9	Повільний відгук сервера	IN PROGRESS	 Add comment	⬆	 Jun 9, 2025	 Jun 9, 2025	 Кирило Пушкар	
☐	❌	SVPIT-10	Втрата даних при оновленні запису	IN PROGRESS	 Add comment	⬆	 Jun 9, 2025	 Jun 9, 2025	 Кирило Пушкар	
☐	⬆	SVPIT-11	Оптимізувати запити до бази даних	TO DO	 Add comment	=	 Jun 9, 2025	 Jun 9, 2025	 Кирило Пушкар	
☐	❌	SVPIT-12	Неправильний формат дати	TO DO	 Add comment	=	 Jun 9, 2025	 Jun 9, 2025	 Кирило Пушкар	
☐	✅	SVPIT-13	Автоматизувати створення Postman запитів для API	TO DO	 Add comment	⬇	 Jun 9, 2025	 Jun 9, 2025	 Кирило Пушкар	
☐	✅	SVPIT-14	Немає обробки помилок при генерації CSV експорту	IN PROGRESS	 Add comment	⬆	 Jun 9, 2025	 Jun 9, 2025	 Кирило Пушкар	
☐	✅	SVPIT-15	Додавання компоненту Sidebar Navigation	IN PROGRESS	 Add comment	=	 Jun 9, 2025	 Jun 9, 2025	 Кирило Пушкар	
☐	⬆	SVPIT-16	Додати обробку ролей користувачів у Team Members	TO DO	 Add comment	⬇	 Jun 9, 2025	 Jun 9, 2025	 Кирило Пушкар	
+ Create										

Рисунок 3.38 – Результат експорту в Jira

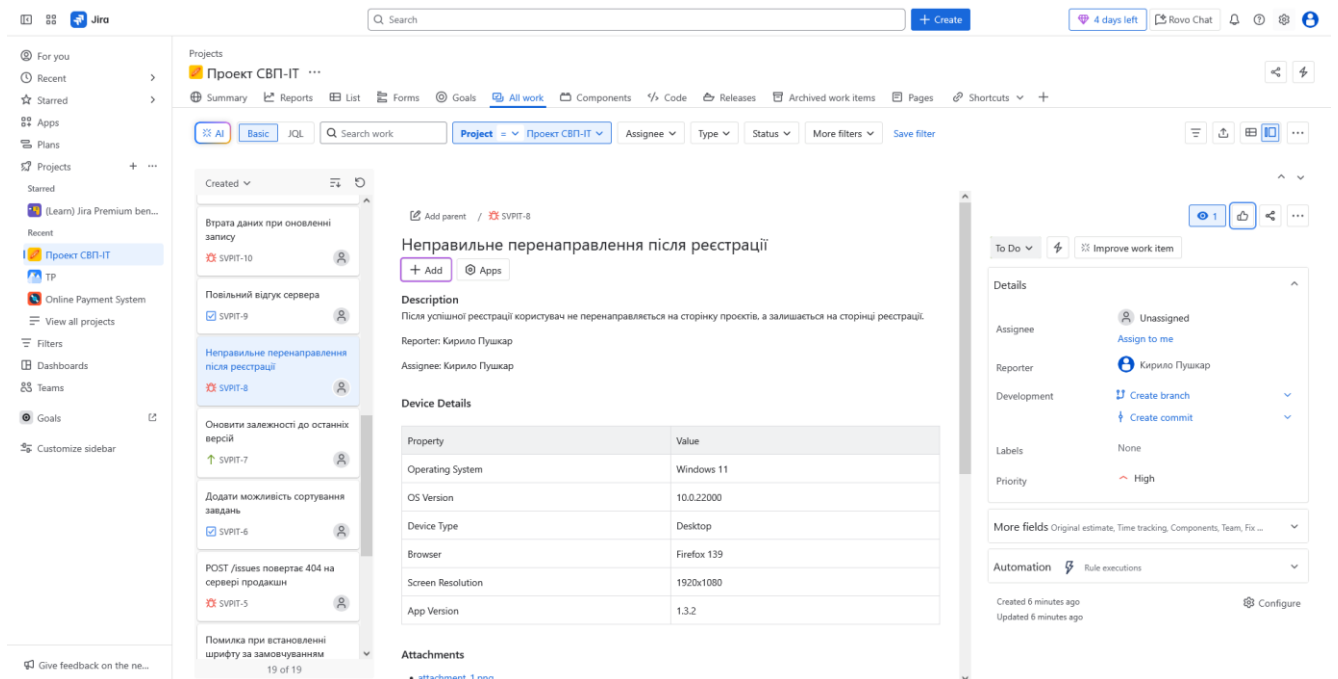


Рисунок 3.39 – Результат експорту в Jira

Розглянемо ще одну важливу функцію системи — підтримку кількох мов інтерфейсу. Перемикання мови здійснюється через бічне меню. На рисунку 3.40 та 3.41 наведено інтерфейс сторінки огляду помилок англійською мовою.

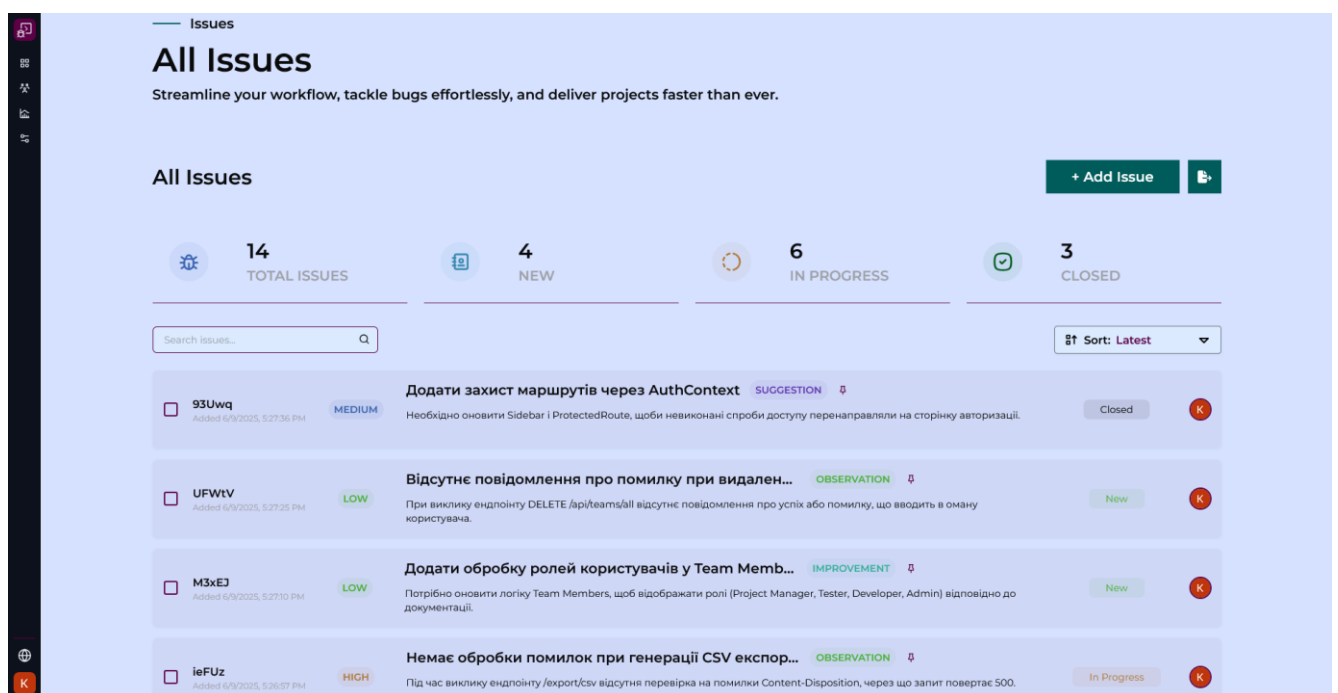


Рисунок 3.40 – Приклад інтерфейсу англійською мовою

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

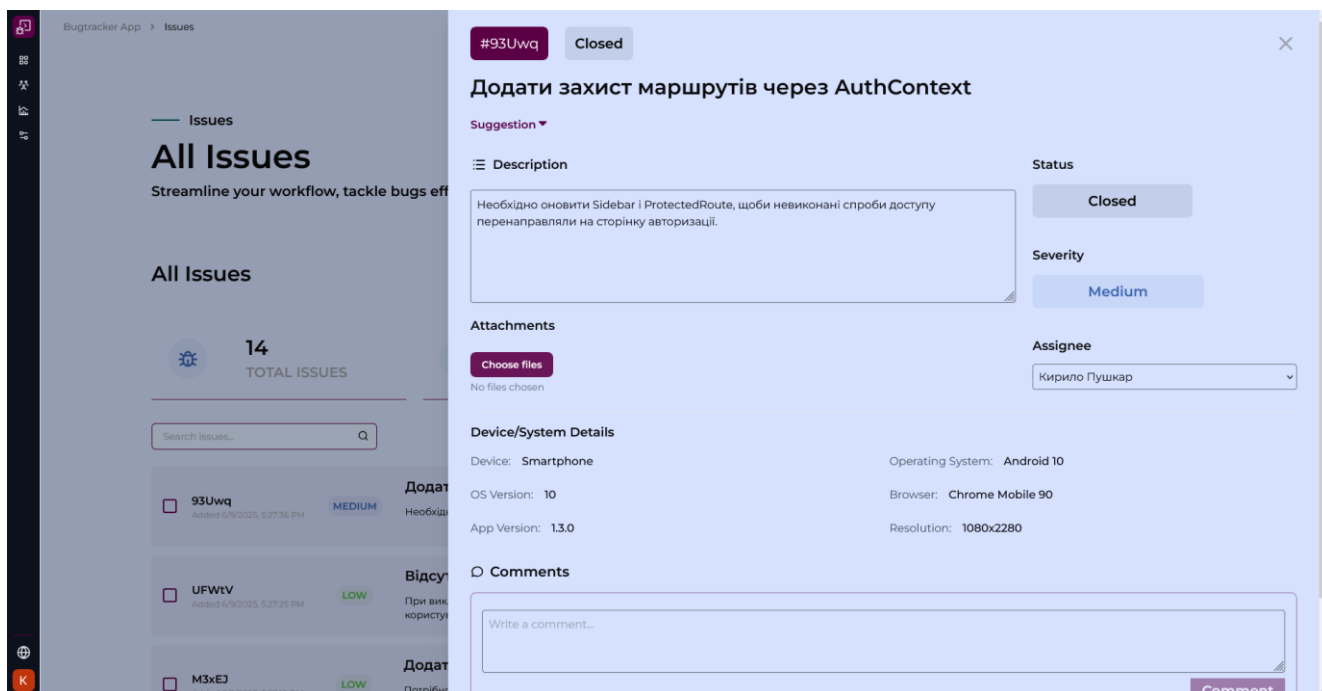


Рисунок 3.41 – Приклад інтерфейсу англійською мовою

3.3. Тестування системи відстеження помилок

Для забезпечення високої якості розробленого програмного забезпечення було проведено тестування API-інтерфейсів за допомогою інструменту Postman[22]. Цей підхід дозволяє перевірити взаємодію між клієнтською та серверною частиною, а також коректність обробки даних та відповідей сервера (рис. 3.42, 3.43). Використання Postman для тестування API дає можливість відправляти запити до різних ендпоінтів системи та аналізувати отримані відповіді. Це дозволяє переконатися, що всі функціональні можливості, такі як реєстрація користувачів, управління проблемами, командна співпраця, аналітика та інтеграції, працюють належним чином. Наприклад, було виконано запити для створення нових користувачів, отримання інформації про існуючих, редагування даних проблем, а також для перевірки механізмів авторизації.

Зокрема, тестування експорту даних до Jira включало перевірку API-запитів, які відповідають за підключення до облікового запису Jira та передачу звітів про помилки. Це передбачало відправлення POST-запитів на ендпоінт «/api/project/{projectId}/jira/export-issues» з відповідними параметрами автентифікації та даними для експорту. Результати цих тестів дозволили

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

підтвердити, що система правильно взаємодіє з Jira, успішно створює та оновлює задачі в зовнішній системі управління проєктами, а також обробляє можливі помилки під час експорту.

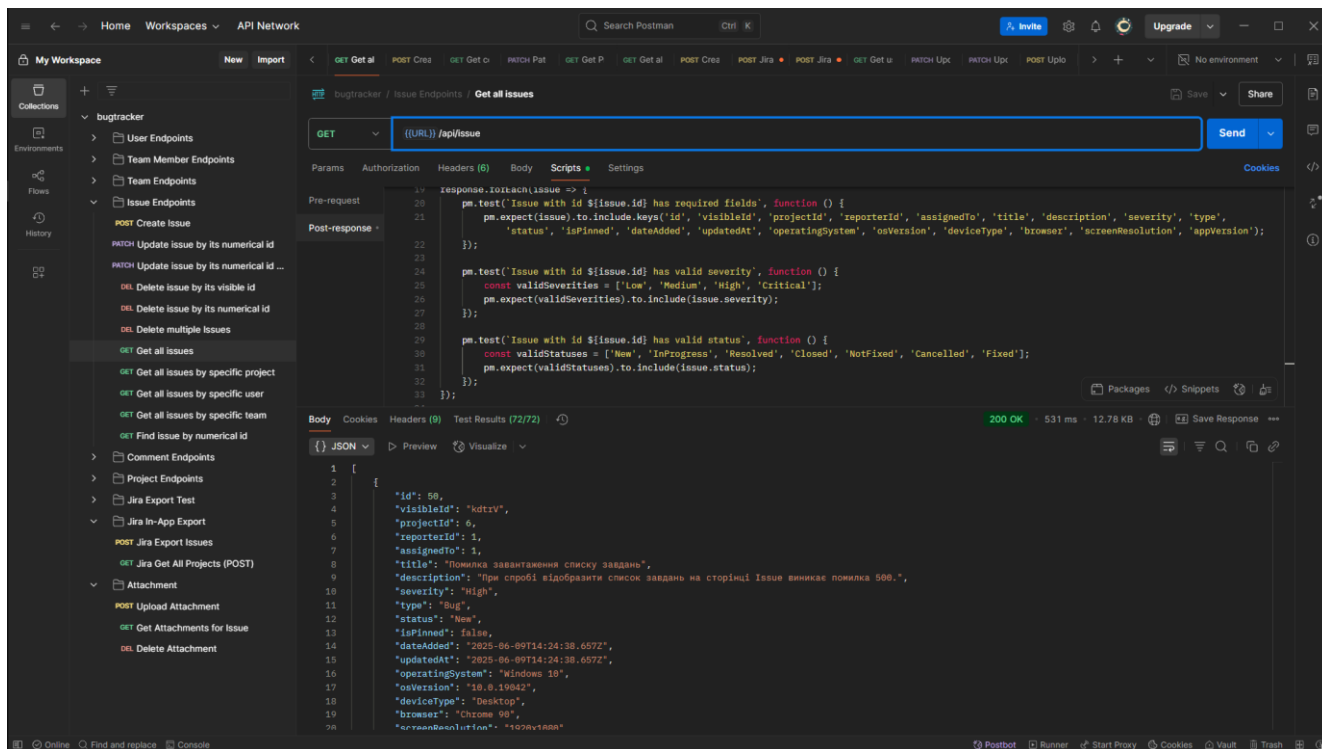


Рисунок 3.42 – Тестування вебдодатку за допомогою Postman

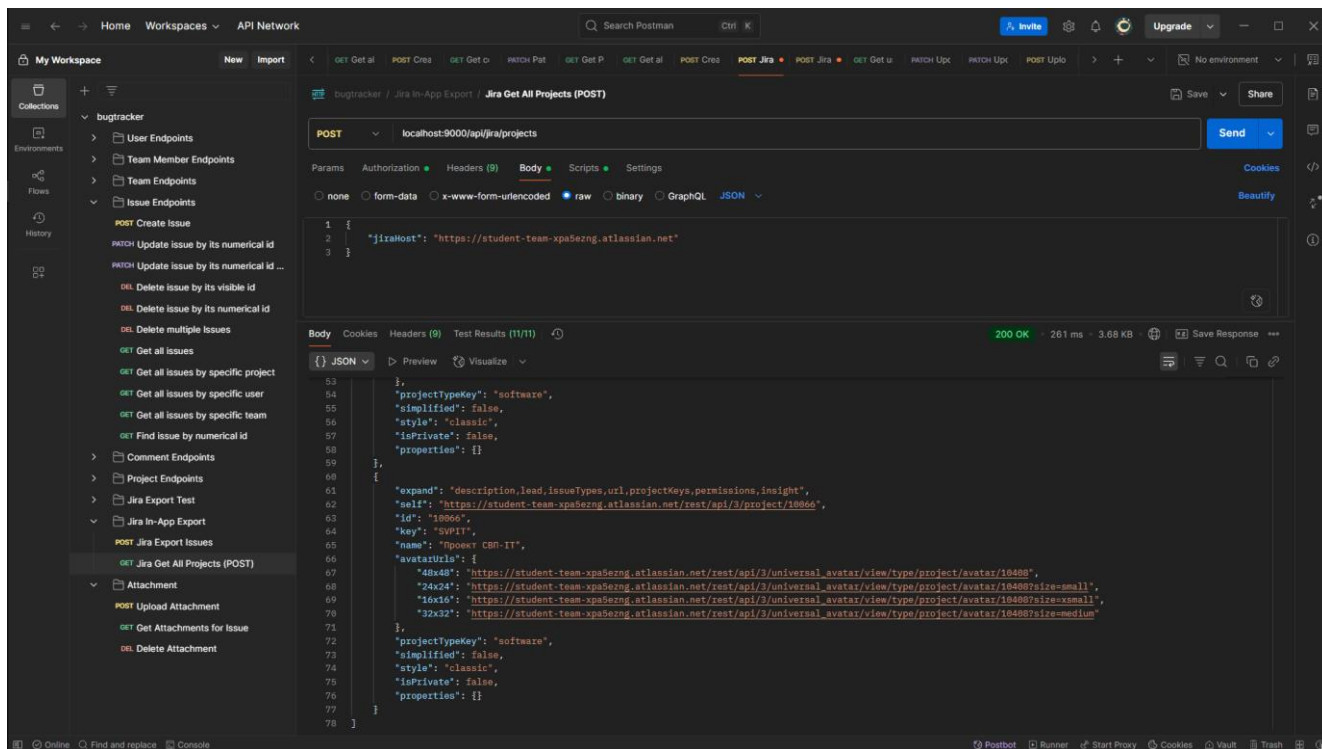


Рисунок 3.43 – Тестування вебдодатку за допомогою Postman

		Пушкар К.В.			Арк.
		Фант М.О.			
Змн.	Арк.	№ докум.	Підпис	Дата	62

Postman також дозволяє автоматизувати тестові сценарії, створюючи колекції запитів та додаючи скрипти для перевірки статус-кодів відповідей, структури даних та відповідності очікуваним результатам. Такий підхід забезпечує комплексне покриття API тестами, дозволяючи швидко виявляти регресії та гарантувати, що зміни в коді не призводять до виникнення нових дефектів у вже реалізованих функціях. Це допомагає підтримувати високу якість API та впевненість у стабільній роботі всіх інтегрованих частин системи відстеження помилок.

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

Висновки до третього розділу

У третьому розділі було детально розглянуто архітектуру, інтерфейс та порядок роботи з розробленою системою відстеження помилок. На основі діаграми розгортання було описано клієнт-серверну архітектуру, базу даних та інтеграції зі сторонніми сервісами, такими як Google Cloud та Jira API.

Було продемонстровано ключові функціональні можливості та візуальні компоненти користувацького інтерфейсу. Детально описано процес взаємодії користувача з системою: від авторизації через Google OAuth до управління командами, проєктами та безпосередньо записами про помилки. Особливу увагу приділено практичним аспектам, таким як створення, редагування, фільтрація та сортування помилок, візуалізація аналітичних даних за допомогою діаграм, а також реалізація багатомовності інтерфейсу.

Важливим результатом роботи, представленої в розділі, є реалізація механізмів експорту даних у форматах CSV, JSON та безпосередньо до системи Jira, що значно підвищує практичну цінність додатку. Надійність та коректність функціонування системи було підтверджено шляхом тестування API за допомогою інструменту Postman, що гарантувало стабільну взаємодію всіх її компонентів.

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра було спроектовано та реалізовано систему відстеження помилок для ІТ-проектів. Проведено всебічний аналіз предметної області, включаючи вивчення існуючих рішень та їхніх функціональних можливостей, що дозволило визначити ключові вимоги до розроблюваної платформи.

На основі цього аналізу була розроблена архітектура системи, визначені підходи та основні технології для реалізації вебдодатка. Проектування включало детальний аналіз варіантів використання програмного продукту та моделювання його структури, зокрема взаємодії між усіма компонентами. Важливою частиною стало проектування реляційної бази даних для ефективного зберігання та управління інформацією про помилки, користувачів, проекти та команди.

У роботі детально описано алгоритм роботи з додатком та послідовні кроки його використання для різних ролей користувачів. Розглянуто основний функціонал, такий як створення звітів про помилки, їхнє оновлення, призначення відповідальних, коментування, а також можливості пошуку, фільтрації та експорту даних.

Крім того, була надана детальна схема розгортання системи, описано взаємодію її структурних елементів та представлено опис користувацького інтерфейсу, адаптованого для різних типів користувачів. Показано роботу аналітичних інструментів та статистики, що надає інформацію про статус помилок, активність команди та інші ключові показники.

Система розроблена з використанням передових технологічних рішень, які гарантують високу ефективність та зручність застосунку. Користувацький інтерфейс спроектований з акцентом на простоту та функціональність, що дає змогу команді швидко та зручно керувати процесами реєстрації, моніторингу та комунікації щодо виявлених помилок.

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Atlassian Jira [Електронний ресурс] – Режим доступу до ресурсу: <https://www.atlassian.com/software/jira>.
2. Bugzilla [Електронний ресурс] – Режим доступу до ресурсу: <https://www.bugzilla.org/>.
3. MantisBT [Електронний ресурс] – Режим доступу до ресурсу: <https://mantisbt.org/>.
4. YouTrack [Електронний ресурс] – Режим доступу до ресурсу: <https://youtrack.jetbrains.com/>.
5. Redmine [Електронний ресурс] – Режим доступу до ресурсу: <https://www.redmine.org/>.
6. Node.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nodejs.org/>.
7. Express [Електронний ресурс] – Режим доступу до ресурсу: <https://expressjs.com/>.
8. TypeScript [Електронний ресурс] – Режим доступу до ресурсу: <https://www.typescriptlang.org/>.
9. React [Електронний ресурс] – Режим доступу до ресурсу: <https://react.dev/>.
10. Vite [Електронний ресурс] – Режим доступу до ресурсу: <https://vite.dev/>.
11. TailwindCSS [Електронний ресурс] – Режим доступу до ресурсу: <https://tailwindcss.com/>.
12. Shadcn/ui [Електронний ресурс] – Режим доступу до ресурсу: <https://ui.shadcn.com/>.
13. LinguiJS [Електронний ресурс] – Режим доступу до ресурсу: <https://lingui.dev/>.
14. Passport.js [Електронний ресурс] – Режим доступу до ресурсу: <https://www.passportjs.org/>.

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

15. OAuth 2.0 [Електронний ресурс] – Режим доступу до ресурсу:
<https://oauth.net/2/>.

16. Prisma [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.prisma.io/>.

17. PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.postgresql.org/>.

18. PgAdmin [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.pgadmin.org/>.

19. Google Cloud Storage [Електронний ресурс] – Режим доступу до ресурсу:
<https://cloud.google.com/storage>.

20. Multer [Електронний ресурс] – Режим доступу до ресурсу:
<https://github.com/expressjs/multer>.

21. Jira-client [Електронний ресурс] – Режим доступу до ресурсу:
<https://github.com/jira-node/node-jira-client>.

22. Postman [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.postman.com/>.

23. Gitlab [Електронний ресурс] – Режим доступу до ресурсу:
<https://gitlab.com/>.

24. Plantuml [Електронний ресурс] – Режим доступу до ресурсу:
<https://plantuml.com/>.

25. Pnpm [Електронний ресурс] – Режим доступу до ресурсу:
<https://pnpm.io/>.

26. react-icons [Електронний ресурс] – Режим доступу до ресурсу:
<https://github.com/react-icons/react-icons>.

27. Nivo [Електронний ресурс] – Режим доступу до ресурсу:
<https://nivo.rocks/>.

28. React-router-dom [Електронний ресурс] – Режим доступу до ресурсу:
<https://github.com/remix-run/react-router>.

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Фант М.О.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

29. Cors [Электронный ресурс] – Режим доступа до ресурсу:
<https://github.com/expressjs/cors>.

30. Nanoid [Электронный ресурс] – Режим доступа до ресурсу:
<https://github.com/ai/nanoid>.

		Пушкар К.В.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

ДОДАТКИ

		Пушкар К.В.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

Опис прецеденту «Вхід/Реєстрація»

Назва варіанту використання:	Вхід/Реєстрація
Основні актори:	Адміністратор системи, розробник, тестувальник, менеджер проєкту, переглядач
Короткий опис:	Користувачі можуть створювати новий обліковий запис або входити в систему для подальшої роботи з помилками. Для додаткової безпеки реалізована автентифікація через зовнішню систему авторизації OAuth 2.0.

Опис прецеденту «Створення звіту про помилку»

Назва варіанту використання:	Створення звіту про помилку
Основні актори:	Розробник, тестувальник, менеджер проєкту
Короткий опис:	Користувачі можуть створювати записи з описом проблем, що виникли. Запис включає деталі: унікальний ідентифікатор, дату створення, пріоритет, тип, а також статус. Ця інформація полегшує подальше розподілення та моніторинг завдань.

Таблиця А.3

Опис прецеденту «Пошук та перегляд помилок»

Назва варіанту використання:	Пошук та перегляд помилок
Основні актори:	Розробник, тестувальник, менеджер проєкту, переглядач, адміністратор системи
Короткий опис:	Користувачі можуть шукати існуючі записи про помилки за різними параметрами, такими як статус, тип, дата створення чи пріоритет, що дозволяє оперативно отримувати необхідну інформацію для аналізу проблем у проєкті.

Таблиця А.4

Опис прецеденту «Оновлення статусу помилки»

Назва варіанту використання:	Оновлення статусу помилки
Основні актори:	Розробник, тестувальник, менеджер проєкту
Короткий опис:	Учасники проєкту можуть змінювати статус записів. Це дозволяє відстежувати етапи усунення помилок та оперативно інформувати команду про поточний стан вирішення проблем.

Таблиця А.5.

Опис прецеденту «Призначення задачі»

Назва варіанту використання:	Призначення задачі
Основні актори:	Менеджер проєкту, адміністратор
Короткий опис:	Менеджери проєкту або адміністратори мають можливість призначати окремі записи помилок конкретним розробникам або тестувальникам для виконання. Це сприяє чіткому розподілу завдань і зменшує час на комунікацію між учасниками процесу.

Таблиця А.6

Опис прецеденту «Коментування в звітах про помилки»

Назва варіанту використання:	Коментування в звітах про помилки
Основні актори:	Розробник, тестувальник, менеджер проєкту
Короткий опис:	Користувачі можуть додавати коментарі до записів про помилки для уточнення деталей, обговорення варіантів рішень або позначення поступу у виправленні. Функціонал коментування забезпечує багаторівневий зворотній зв'язок і сприяє підвищенню ефективності командної роботи.

Опис прецеденту «Аналітика та звітність»

Назва варіанту використання:	Аналітика та звітність
Основні актори:	Менеджер проєкту, адміністратор
Короткий опис:	Система формує аналітичні звіти щодо показників роботи (кількість нових, в роботі, вирішених та закритих помилок, розподіл за пріоритетами та типами). Ці дані допомагають приймати керівничі рішення для підвищення якості розробки та оптимізації процесів.

Опис прецеденту «Експорт даних про помилки в форматі CSV»

Назва варіанту використання:	Експорт даних про помилки в форматі CSV
Основні актори:	Менеджер проєкту, адміністратор
Короткий опис:	Система надає можливість експортувати інформацію про помилки у вигляді CSV-файлу, що містить деталі записів (ідентифікатор, пріоритет, статус, дату створення). Це дозволяє аналізувати дані за допомогою зовнішніх інструментів, створювати звіти та інтегрувати інформацію з іншими системами аналітики.

Опис прецеденту «Експорт даних про помилки до Jira (одноразова синхронізація)»

Назва варіанту використання:	Експорт даних про помилки в зовнішні системи менеджменту проєктів
Основні актори:	Менеджер проєкту, адміністратор
Короткий опис:	Цей прецедент забезпечує можливість одноразового експорту даних про помилки з локальної системи відстеження до конкретного проєкту в Jira. Після виконання операції всі обрані записи про помилки створюються в зазначеному Jira-проєкті як окремі завдання. Одноразова синхронізація означає, що після завершення експорту подальші зміни локальних помилок не віддзеркалюватимуться автоматично в Jira. Це дозволяє уникнути надлишкового дублювання даних і дає змогу чітко контролювати момент передачі інформації.

Таблиця А.10

Опис прецеденту «Створення та менеджмент команд»

Назва варіанту використання:	Створення та менеджмент команд
Основні актори:	Адміністратор, менеджер проєкту
Короткий опис:	Користувачі з відповідними повноваженнями можуть створювати нові команди, додавати до них користувачів, визначати ролі (переглядач, розробник, тестувальник) та редагувати інформацію про команду. Це дозволяє організувати роботу в межах проєкту та забезпечити ефективну комунікацію між членами команди.

Таблиця А.11

Опис прецеденту «Створення та менеджмент проєктів»

Назва варіанту використання:	Створення та менеджмент команд
Основні актори:	Адміністратор, менеджер проєкту
Короткий опис:	Користувачі можуть створювати нові проєкти, редагувати існуючі та додавати проєкти до команд. Цей модуль дозволяє управляти життєвим циклом проєкту від ініціації до випуску, забезпечуючи контроль над процесами розробки.

Таблиця А.12

Опис прецеденту «Зміна даних користувача»

Назва варіанту використання:	Створення та менеджмент команд
Основні актори:	Адміністратор системи, розробник, тестувальник, менеджер проєкту, переглядач
Короткий опис:	Зареєстрований користувач переходить на сторінку свого профілю та змінює особисті. Система порівнює нові значення з початковими, відправляє запит на сервер для оновлення інформації та показує повідомлення про результат операції.

Таблиця А.13

Опис прецеденту «Зміна даних користувача»

Назва варіанту використання:	Сортування та фільтрування помилок
Основні актори:	Адміністратор системи, розробник, тестувальник, менеджер проєкту, переглядач
Короткий опис:	Користувач переглядає список помилок у межах певного проєкту. За допомогою інтерфейсу він може задати критерій сортування (дата створення тощо) та обрати фільтр за статусом. Система динамічно застосовує сортування і фільтрацію, відображаючи відфільтрований і впорядкований перелік записів без перезавантаження сторінки.

App.ts

```

import cors from "cors";
import type { Application } from "express";
import express from "express";
import session from "express-session";
import passport from "passport";
import "api/config/Passport";

import IssueRoute from "api/routes/IssueRoute";
import ProjectRoute from "api/routes/ProjectRoute";
import UserRoute from "api/routes/UserRoute";
import TeamRoute from "api/routes/TeamRoute";
import TeamMembersRoute from "api/routes/TeamMemberRoute";
import CommentRoute from "api/routes/CommentRoute";
import JiraRoute from "api/routes/JiraRoute";
import AttachmentRoute from "api/routes/AttachmentRoute";

import AuthRoute from "api/auth/AuthRoute";

import environment from "api/config/Environment";

const app: Application = express();

app.use(cors({ credentials: true }));

app.use(
  session({
    secret: environment.SESSION_SECRET,
    resave: true,
    saveUninitialized: false,
    cookie: { secure: false, maxAge: 8 * 60 * 60 * 1000 }
  })
);

app.use(passport.initialize());
app.use(passport.session());

app.use(express.json({ limit: "20mb" }));

app.use("/api", IssueRoute);

app.use("/api", ProjectRoute);

app.use("/api", UserRoute);

app.use("/api", TeamRoute);

app.use("/api", TeamMembersRoute);

```

		Пушкар К.В.			ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
		Фант М.О.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

app.use("/api", CommentRoute);

app.use("/api", JiraRoute);

app.use("/api", AttachmentRoute);

app.use("/api", AuthRoute);

export default app;

```

AuthRoute.ts

```

import { Router } from "express";
import passport from "passport";

const router = Router();

router.get(
  "/auth/google",
  passport.authenticate("google", {
    successRedirect: "/projects/manage",
    scope: ["email", "profile"]
  })
);

router.get(
  "/auth/google/callback",
  passport.authenticate("google", {
    successRedirect: "/projects/manage",
    failureRedirect: "/login"
  })
);

router.get("/currentUser", (req, res) => {
  if (req.isAuthenticated()) {
    res.json(req.user);
  } else {
    res.status(401).json({ error: "Not authenticated" });
  }
});

router.post("/logout", (req, res, next) => {
  req.logout((err) => {
    if (err) {
      return next(err);
    }
    res.redirect("/");
  });
});

export default router;

```

		Пушкар К.В.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Фант М.О.				78
Змн.	Арк.	№ докum.	Підпис	Дата		

gcs.ts

```
/* eslint-disable @typescript-eslint/no-explicit-any */
import { Storage } from "@google-cloud/storage";
import environment from "api/config/Environment";
// eslint-disable-next-line @typescript-eslint/no-unused-vars
import { Multer } from "multer";
import { nanoid } from "nanoid";

const storage = new Storage({
  projectId: environment.GOOGLE_PROJECT_ID,
  keyFilename: environment.GOOGLE_APPLICATION_CREDENTIALS
});

const bucketName = environment.GCS_BUCKET_NAME;
const bucket = storage.bucket(bucketName);

async function uploadFileToGCS(file: Express.Multer.File): Promise<string> {
  try {
    console.log(bucket);
    const filename = `${nanoid()}-${file.originalname}`;
    const fileBuffer = file.buffer;
    const fileMimeType = file.mimetype;

    await bucket.file(filename).save(fileBuffer, {
      metadata: {
        contentType: fileMimeType
      }
    });

    return
    `https://storage.cloud.google.com/${bucketName}/${filename}?authuser=1`;
  } catch (error: any) {
    console.error("Error uploading to GCS:", error);
    throw new Error("Failed to upload file to Google Cloud Storage.");
  }
}

async function deleteFileFromGCS(fileUrl: string): Promise<void> {
  try {
    const url = new URL(fileUrl);

    const pathSegments = url.pathname.split("/");
    const encodedFilename = pathSegments.pop();

    if (encodedFilename) {
      const filename = decodeURIComponent(encodedFilename);
      await bucket.file(filename).delete();
      console.log(`File ${filename} deleted from GCS.`);
    } else {
      console.warn(`Could not extract filename from URL: ${fileUrl}`);
    }
  }
}
```

		Пушкар К.В.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Фант М.О.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    } catch (error: any) {
        console.error("Error deleting from GCS:", error);
        throw new Error("Failed to delete file from Google Cloud Storage.");
    }
}

export { uploadFileToGCS, deleteFileFromGCS };

```

AttachmentService.ts

```

/* eslint-disable no-useless-catch */
import { PrismaClient, Attachment } from "@prisma/client";
import { uploadFileToGCS, deleteFileFromGCS } from "api/lib/gcs";

const prisma = new PrismaClient();

export async function uploadAttachment(
    issueId: number,
    userId: number | undefined,
    file: Express.Multer.File
): Promise<Attachment> {
    try {
        const gcsUrl = await uploadFileToGCS(file);

        const attachment = await prisma.attachment.create({
            data: {
                issueId,
                userId,
                filename: file.originalname,
                url: gcsUrl,
                mimeType: file.mimetype,
                size: file.size
            }
        });
        return attachment;
    } catch (error) {
        throw error;
    }
}

export async function deleteAttachment(
    attachmentId: number
): Promise<Attachment> {
    try {
        const attachmentToDelete = await prisma.attachment.findUnique({
            where: { id: attachmentId }
        });

        if (!attachmentToDelete) {
            throw new Error("Attachment not found.");
        }
    }
}

```

		Пушкар К.В.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Фант М.О.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        await deleteFileFromGCS(attachmentToDelete.url);

        const deletedAttachment = await prisma.attachment.delete({
            where: { id: attachmentId }
        });

        return deletedAttachment;
    } catch (error) {
        throw error;
    }
}

export async function getAttachmentsByIssueId(
    issueId: number
): Promise<Attachment[]> {
    try {
        const attachments = await prisma.attachment.findMany({
            where: { issueId }
        });
        return attachments;
    } catch (error) {
        throw error;
    }
}

```

JiraController.ts

```

/* eslint-disable @typescript-eslint/no-explicit-any */
import { Request, Response } from "express";
import JiraClient from "jira-client";
import {
    IssueTypeEnum,
    IssueSeverityEnum,
    IssueStatusEnum
} from "@prisma/client";
import getErrorMessage from "../utils/Error";
import { exportIssuesForProject } from "../services/IssueService";

export async function getJiraProjectsController(
    req: Request,
    res: Response
): Promise<void> {
    try {
        const jiraHost = req.body.jiraHost;
        if (!jiraHost) {
            res.status(400).send("Missing jiraHost in request body");
            return;
        }

        const authHeader = req.header("authorization");
        if (!authHeader?.startsWith("Basic ")) {
            res.status(400).send(

```

		Пушкар К.В.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Фант М.О.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        "Missing or invalid Authorization header (Basic expected)"
    );
    return;
}

const [jiraUserEmail, jiraApiToken] = Buffer.from(
    authHeader.slice("Basic ".length),
    "base64"
)
    .toString("utf-8")
    .split(":");
if (!jiraUserEmail || !jiraApiToken) {
    res.status(400).send("Invalid Basic auth credentials");
    return;
}

const jira = new JiraClient({
    protocol: "https",
    host: jiraHost.replace(/^https?:\/\//, ""),
    username: jiraUserEmail,
    password: jiraApiToken,
    apiVersion: "3",
    strictSSL: true
});

const projects = await jira.listProjects();
res.json(projects);
} catch (err: any) {
    res.status(500).send(getErrorMessage(err));
}
}

export async function exportToJiraController(
    req: Request,
    res: Response
): Promise<void> {
    try {
        const projectId = Number(req.params.id);
        const { jiraHost, jiraProjectKey } = req.body;

        if (!jiraHost || !jiraProjectKey) {
            res.status(400).send("Missing jiraHost or jiraProjectKey in body");
            return;
        }

        // Extract Basic auth header
        const authHeader = req.header("authorization");
        if (!authHeader?.startsWith("Basic ")) {
            res.status(400).send(
                "Missing or invalid Authorization header (Basic expected)"
            );
            return;
        }
    }
}

```

		Пушкар К.В.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Фант М.О.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}

// Decode credentials
const base64Creds = authHeader.slice("Basic ".length).trim();
let jiraUserEmail: string, jiraApiToken: string;
try {
    const decoded = Buffer.from(base64Creds, "base64").toString(
        "utf-8"
    );
    [jiraUserEmail, jiraApiToken] = decoded.split(":");
    if (!jiraUserEmail || !jiraApiToken) {
        throw new Error("Invalid Basic auth format");
    }
} catch {
    res.status(400).send("Unable to parse Basic auth credentials");
    return;
}

// Initialize Jira client
const jira = new JiraClient({
    protocol: "https",
    host: jiraHost.replace(/^https?:\/\//, ""),
    username: jiraUserEmail,
    password: jiraApiToken,
    apiVersion: "3",
    strictSSL: true
});

// Fetch local issues
const issues = await exportIssuesForProject(projectId);
const results: Array<{
    success: boolean;
    data?: any;
    transitionError?: any;
    error?: any;
}> = [];

// Main loop: create + optional transition
for (const issue of issues) {
    const payload = {
        fields: {
            project: { key: jiraProjectKey },
            summary: issue.title,
            description: formatJiraDescription(issue),
            issuetype: { name: mapIssueTypeToJira(issue.type) },
            priority: {
                name: mapSeverityToJiraPriority(issue.severity)
            }
        }
    };

    try {

```

		Пушкар К.В.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Фант М.О.				83
Змн.	Арк.	№ докum.	Підпис	Дата		

```

// create issue
const created = await jira.addNewIssue(payload);
const entry: any = { success: true, data: created };

// if not default status, try transition
if (issue.status !== IssueStatusEnum.New) {
    try {
        const { transitions } = await jira.listTransitions(
            created.key
        );
        const desired = mapStatusToJira(
            issue.status
        ).toLowerCase();
        const match = transitions.find(
            (t: any) => t.name.toLowerCase() === desired
        );

        if (match) {
            await jira.transitionIssue(created.key, {
                transition: { id: match.id },
                fields: getResolutionFields(issue.status)
            });
        } else {
            entry.transitionError = `No '${desired}' transition
found`;
        }
    } catch (te: any) {
        entry.transitionError = te.responseBody ?? te.message;
    }
}

results.push(entry);
} catch (err: any) {
    results.push({
        success: false,
        error: err.responseBody ?? err.message
    });
}
}

res.status(200).json(results);
} catch (error) {
    res.status(500).send(getErrorMessage(error));
}
}

// Returns any extra fields Jira needs when moving to "Resolved"/"Closed"
function getResolutionFields(status: IssueStatusEnum) {
    if (status === IssueStatusEnum.Fixed || status === IssueStatusEnum.Closed) {
        return { resolution: { name: "Done" } };
    }
    return {};
}

```

		Пушкар К.В.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		84

```

}

function formatJiraDescription(issue: any) {
  const content: any[] = [
    {
      type: "paragraph",
      content: [
        { text: issue.description || "No description", type: "text" }
      ]
    }
  ];

  // Reporter/assignee info
  const userDetails = [
    { label: "Reporter", value: issue.reporter?.username },
    { label: "Assignee", value: issue.assigned?.username }
  ];
  userDetails.forEach(({ label, value }) => {
    if (value) {
      content.push({
        type: "paragraph",
        content: [{ text: `${label}: ${value}`, type: "text" }]
      });
    }
  });

  // Device details table if any exists
  const deviceDetails = [
    { label: "Operating System", value: issue.operatingSystem },
    { label: "OS Version", value: issue.osVersion },
    { label: "Device Type", value: issue.deviceType },
    { label: "Browser", value: issue.browser },
    { label: "Screen Resolution", value: issue.screenResolution },
    { label: "App Version", value: issue.appVersion }
  ].filter((d) => d.value);

  if (deviceDetails.length > 0) {
    content.push({
      type: "heading",
      attrs: { level: 3 },
      content: [{ type: "text", text: "Device Details" }]
    });

    const tableRows = deviceDetails.map((detail) => ({
      type: "tableRow",
      content: [
        {
          type: "tableCell",
          content: [
            {
              type: "paragraph",
              content: [{ text: detail.label, type: "text" }]
            }
          ]
        }
      ]
    }));
  }
}

```

		Пушкар К.В.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Фант М.О.				85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        }
    ]
},
{
    type: "tableCell",
    content: [
        {
            type: "paragraph",
            content: [{ text: detail.value, type: "text" }]
        }
    ]
}
]
}
]
})))

content.push({
    type: "table",
    content: [
        // Header row
        {
            type: "tableRow",
            content: [
                {
                    type: "tableHeader",
                    content: [
                        {
                            type: "paragraph",
                            content: [
                                { text: "Property", type: "text" }
                            ]
                        }
                    ]
                },
                {
                    type: "tableHeader",
                    content: [
                        {
                            type: "paragraph",
                            content: [{ text: "Value", type: "text" }]
                        }
                    ]
                }
            ]
        }
    ],
    ...tableRows
}
]
})))

// Attachments section if any exist
if (issue.Attachment?.length > 0) {
    content.push({

```

		Пушкар К.В.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

```

        type: "heading",
        attrs: { level: 3 },
        content: [{ type: "text", text: "Attachments" }]
    });

    const attachmentItems = issue.Attachment.map((att: any) => ({
        type: "listItem",
        content: [
            {
                type: "paragraph",
                content: [
                    {
                        type: "text",
                        text: att.filename,
                        marks: [
                            {
                                type: "link",
                                attrs: { href: att.url }
                            }
                        ]
                    }
                ]
            }
        ]
    }));

    content.push({
        type: "bulletList",
        content: attachmentItems
    });
}

return { type: "doc", version: 1, content };
}

function mapIssueTypeToJira(type: IssueTypeEnum): string {
    switch (type) {
        case IssueTypeEnum.Bug:
            return "Bug";
        case IssueTypeEnum.Improvement:
            return "Improvement";
        case IssueTypeEnum.NewFeature:
            return "New Feature";
        default:
            return "Task";
    }
}

function mapSeverityToJiraPriority(sev: IssueSeverityEnum): string {
    switch (sev) {
        case IssueSeverityEnum.Critical:
            return "Highest";
    }
}

```

		Пушкар К.В.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Фант М.О.				87
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        case IssueSeverityEnum.High:
            return "High";
        case IssueSeverityEnum.Medium:
            return "Medium";
        case IssueSeverityEnum.Low:
            return "Low";
        default:
            return "Medium";
    }
}

function mapStatusToJira(status: IssueStatusEnum): string {
    switch (status) {
        case IssueStatusEnum.New:
            return "To Do";
        case IssueStatusEnum.InProgress:
            return "In Progress";
        case IssueStatusEnum.Fixed:
            return "Resolved";
        case IssueStatusEnum.NotFixed:
            return "Reopened";
        case IssueStatusEnum.Closed:
        case IssueStatusEnum.Cancelled:
            return "Closed";
        default:
            return "To Do";
    }
}

```

		Пушкар К.В.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		88