

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

«Вебплатформа для організації групових проєктів»

Виконав студент групи ПЗ-22-4
ПАСТУХОВ Олександр Костянтинович

Керівник роботи:
САВІЦЬКИЙ Роман Святославович

Рецензент:
ЄФРЕМОВ Юрій Миколайович

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ПАСТУХОВ Олександр Костянтинович**

Керівник роботи: **САВІЦЬКИЙ Роман Святославович**

Тема роботи: **«Вебплатформа для організації групових проєктів»**,
затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., №92/с**

Вихідні дані для роботи: **розробка веборієнтованої платформи для організації групових проєктів на основі клієнт-серверної архітектури з використанням фреймворку ASP.NET Core, бібліотеки React та системи керування базами даних PostgreSQL.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Савіцький Р. С.	18.03.2026р.	22.03.2026р.
2	Савіцький Р. С.	20.04.2026р.	28.04.2026р.
3	Савіцький Р. С.	15.05.2026р.	18.05.2026р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з вебдодатку платформи для організації групових проєктів та пояснювальної записки. Пояснювальна записка до випускної роботи містить 99 сторінок, 54 ілюстрації та 14 таблиць.

Метою роботи є створення вебплатформи для організації групових проєктів із використанням сучасних вебтехнологій, яка об'єднує в єдиному цифровому середовищі інструменти керування проєктами, постановку та контроль виконання завдань, перевірку результатів роботи менторами, в режимі реального часу комунікацію між учасниками та формування звітності.

У першому розділі виконано аналіз предметної області організації групових проєктів та досліджено провідні програмні аналоги. На основі проведеного аналізу сформовано функціональні і нефункціональні вимоги до власної платформи та обґрунтовано вибір технологічного стеку.

У другому розділі виконано проєктування платформи. Визначено варіанти використання системи, спроектовано об'єктно-орієнтовану модель та реляційну базу даних, розроблено алгоритми ключових бізнес-процесів.

У третьому розділі описано практичну реалізацію та тестування платформи. Серверна частина побудована на ASP.NET Core 8 з PostgreSQL, клієнтська на React 18 з TypeScript.

У результаті виконання роботи розроблено готову до впровадження вебплатформу, яка дозволяє створювати проєкти, розподіляти завдання, відстежувати їх виконання та взаємодіяти в межах єдиного цифрового середовища з настільних і мобільних пристроїв.

КЛЮЧОВІ СЛОВА: ВЕБПЛАТФОРМА, ГРУПОВІ ПРОЄКТИ, КЕРУВАННЯ ЗАВДАННЯМИ, КОМАНДНА РОБОТА, ЧИСТА АРХІТЕКТУРА, ASP.NET CORE, REACT, POSTGRESQL, SIGNALR, DOCKER.

					ІПЗ.КР.Б – 121 – 26 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	«Вебплатформа для організації групових проєктів»	Лім.	Арк.	Аркуші
Розроб.		Пастухов О.К						
Керівник		Савіцький Р. С					3	99
						Житомирська політехніка, група ІПЗ-22-4		
Н. контр.								
Зав. каф.		Вакалюк Т.А.			Пояснювальна записка			

ABSTRACT

The bachelor's qualification work consists of a web application for a group project management platform and an explanatory note. The explanatory note contains 99 pages, 54 figures, and 14 tables.

The purpose of this work is to develop a web-based platform for organizing group projects using modern web technologies, which integrates project management tools, task assignment and monitoring, mentor review of completed work, real-time communication among participants, and reporting capabilities within a single digital environment.

The first chapter presents an analysis of the domain of group project organization and a study of leading software analogues. Based on the conducted analysis, functional and non-functional requirements for the platform were defined, and the choice of the technological stack was justified.

The second chapter describes the platform design. System use cases were identified, an object-oriented model and a relational database were designed, and algorithms for key business processes were developed.

The third chapter covers the practical implementation and testing of the platform. The server-side application was developed using ASP.NET Core 8 and PostgreSQL, while the client-side application was implemented using React 18 and TypeScript.

As a result of this work, a web platform ready for deployment was developed. The platform enables users to create projects, assign tasks, monitor their execution, and collaborate within a unified digital environment from both desktop and mobile devices.

KEYWORDS: WEB PLATFORM, GROUP PROJECTS, TASK MANAGEMENT, TEAMWORK, CLEAN ARCHITECTURE, ASP.NET CORE, REACT, POSTGRESQL, SIGNALR, DOCKER.

		Пастухов О. К.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Савицький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ОРГАНІЗАЦІЇ ГРУПОВИХ ПРОЄКТІВ	9
1.1 Постановка задачі	9
1.2 Аналіз аналогів програмного продукту	11
1.3 Вибір архітектури системи	16
1.4. Обґрунтування вибору інструментальних засобів	19
Висновки до першого розділу.....	22
РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБОРІЄНТОВАНОЇ ПЛАТФОРМИ.....	24
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	24
2.2. Розробка бази даних системи.....	28
2.3 Проєктування та реалізація алгоритмів роботи системи	39
2.4 Реалізація функціоналу платформи для організації групових проєктів	44
Висновки до другого розділу.....	48
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ ДЛЯ ОРГАНІЗАЦІЇ ГРУПОВИХ ПРОЄКТІВ	49
3.1 Архітектура розгортання та налаштування параметрів платформи.....	49
3.2. Структура інтерфейсу вебплатформи	53
3.3 Тестування платформи	83
Висновки до третього розділу	87
ВИСНОВКИ.....	88
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	90
ДОДАТКИ.....	93
Додаток А.....	94

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савицький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface, прикладний програмний інтерфейс

ACID – Atomicity, Consistency, Isolation, Durability, набір властивостей транзакцій бази даних

CORS – Cross-Origin Resource Sharing, механізм спільного використання ресурсів між різними джерелами

CRUD – Create, Read, Update, Delete, базові операції роботи з даними

CSS – Cascading Style Sheets, каскадні таблиці стилів

DTO – Data Transfer Object, об'єкт передачі даних

HTML – HyperText Markup Language, мова розмітки гіпертексту

HTTP – HyperText Transfer Protocol, протокол передачі гіпертексту

HTTPS – HyperText Transfer Protocol Secure, захищений протокол передачі гіпертексту

JSON – JavaScript Object Notation, текстовий формат обміну даними

JWT – JSON Web Token, маркер автентифікації та авторизації користувача

SPA – Single Page Application, односторінковий застосунок

SQL – Structured Query Language, мова структурованих запитів

UI – User Interface, інтерфейс користувача

UX – User Experience, досвід користувача

БД – база даних

ПЗ – програмне забезпечення

СУБД – система керування базами даних

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми зумовлена необхідністю використання сучасних інформаційних технологій для організації групової співпраці над спільними проєктами. Зростання складності розробки програмного забезпечення вимагає залучення різних спеціалістів, що значно ускладнює командну взаємодію, розподіл завдань та контроль за станом проєкту. Застосування розрізнених програмних засобів, таких як окремі месенджери для спілкування, зовнішні дошки планування та ізольовані сховища даних, часто призводить до втрати контексту. Особливо гостро постає проблема інтеграції процесу рецензування безпосередньо в життєвий цикл виконання задачі. Популярні платформи управління проєктами зазвичай занадто складні для невеликих команд, але при цьому не мають зручних інструментів для перевірки завдань, надання зворотного зв'язку та ведення бази знань. Відсутність зручного середовища змушує команди витрачати багато часу на перемикання між різними додатками замість виконання поставлених цілей. Вирішенням цих проблем є розробка вебплатформи для прозорого розподілу завдань, налаштування прав доступу, ведення комунікації та створення звітних документів. Застосування такого підходу дозволяє об'єднати всі інструменти в одному місці. Зберігання документації та завдань разом суттєво полегшує залучення нових учасників до роботи. Зазначені фактори підтверджують актуальність та доцільність обраної теми дослідження.

Метою випускної кваліфікаційної роботи є створення єдиної вебплатформи для організації спільної роботи над груповими проєктами, яка інтегрує засоби керування завданнями, налаштування прав доступу та рецензування результатів менторами в одному цифровому середовищі.

Для реалізації поставленої мети передбачено послідовне вирішення взаємопов'язаних завдань. На початковому етапі необхідно провести комплексний аналіз предметної області організації групових проєктів, детально дослідити існуючі програмні аналоги та на основі отриманих даних визначити ключові функціональні й нефункціональні вимоги до майбутньої системи. Наступний крок передбачає теоретичне обґрунтування вибору архітектури

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

платформи та відповідних інструментальних засобів її реалізації, а також безпосереднє проєктування логічної та фізичної моделей інформаційної системи. Фінальним етапом роботи є практична реалізація та розгортання спроектованої вебплатформи для ефективної організації групових проєктів у реальних умовах. Об'єктом дослідження є процес організації групової роботи над проєктами в межах веборієнтованої інформаційної системи.

Об'єктом дослідження є процес організації групової роботи та координації спільної діяльності учасників над проєктами в межах веборієнтованих інформаційних систем. У межах цього об'єкта особлива увага приділяється взаємодії між членами команди, етапам життєвого циклу управління завданнями, а також механізмам контролю строків виконання та моніторингу загального прогресу проєктної діяльності.

Предметом дослідження є методи, засоби та технології розробки вебплатформи для організації групових проєктів, її функціональні можливості та переваги у порівнянні з використанням розрізнених цифрових сервісів для командної роботи.

Практичне значення одержаних результатів полягає у створенні готової до впровадження вебплатформи, яка забезпечує повний цикл організації та ведення групових проєктів у межах єдиного цифрового простору. Розроблений застосунок дозволяє повністю відмовитися від використання кількох розрізнених сервісів, поєднуючи в собі інструменти для формування різноманітних проєктів, гнучкого менеджменту завдань, збереження робочих файлів та ведення внутрішніх обговорень.

За темою випускної кваліфікаційної роботи було опубліковано тезу [30].

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ОРГАНІЗАЦІЇ ГРУПОВИХ ПРОЄКТІВ

1.1 Постановка задачі

У межах дослідження, що проводиться у випускній кваліфікаційній роботі бакалавра, першочерговою задачею є всебічний аналіз процесів спільної діяльності та координації робочих груп під час ведення колективних проєктів. Постановка задачі передбачає детальне визначення вимог до майбутнього програмного продукту, виявлення існуючих технологічних обмежень, а також чітке окреслення цільової аудиторії системи та її ключових експлуатаційних потреб.

Цільова аудиторія розроблюваного програмного простору охоплює два ключові сектори: академічне середовище, представлене студентами, викладачами та науковими керівниками, а також сферу малого бізнесу, де система буде корисною для гнучких ІТ-стартапів та невеликих проєктних груп. Важливою вимогою до логіки побудови вебплатформи є її адаптивність до постійної зміни функціональних обов'язків користувачів. Оскільки в межах паралельних процесів один і той самий учасник інформаційної взаємодії може одночасно поєднувати функції організатора, який здійснює менеджмент дедлайнів і делегування задач, та звичайного виконавця, відповідального за реалізацію окремих етапів суміжних завдань. Окрему категорію користувачів становлять ментори – це довірені учасники, до завдань яких належить перевірка результатів виконання задач, надання змістовного зворотного зв'язку та супровід менш досвідчених членів команди.

Сучасний стан організації командної роботи демонструє значні труднощі оскільки всі потрібні інструменти розкидані по різних сервісах. Щоб вирішувати щоденні завдання, учасники проєктів змушені одночасно використовувати багато непов'язаних між собою програм: окремо чати для спілкування, окремо хмарні диски для зберігання файлів, сторонні системи керування завданнями для контролю задач та електронні таблиці для ведення звітності. Через використання такої великої кількості різних сервісів часто губиться важлива інформація,

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савицький Р. С.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

зникає історія обговорень, стає важко стежити за дедлайнами, а сама команда витрачає на координацію багато зайвого часу. Додаткові труднощі виникають у разі підключення до проєкту нового учасника, який змушений отримати доступ одразу до декількох систем, перейти базовий процес адаптації у кожній з них, відновити контекст попередніх обговорень із різних каналів комунікації, що суттєво уповільнює приєднання нових членів команди до продуктивної роботи.

Окремою проблемою є нерівномірність контролю якості виконання задач у наявних аналогічних рішеннях. У більшості систем, які використовуються для управління проєктами, відсутні вбудовані механізми поетапного приймання роботи, фіксації причин повернення задачі на доопрацювання та обліку кількості ітерацій, що знадобились для досягнення прийнятного результату. Це призводить до того, що менеджери чи ментори не мають об'єктивних даних для оцінювання завдань, ефективності виконавців та якості виконаної роботи. Внаслідок цього, ускладнюється формування справедливої системи мотивації, планування подальших етапів проєкту та збір метрик для аналізу результативності команди.

Для подолання негативних чинників, було поставлено завдання створити єдину вебплатформу, яка об'єднає всі етапи роботи над проєктами в одному місці. Програмний продукт повинен мати простий інтерфейс, щоб користувачі могли легко створювати нові проєкти, швидко збирати команди та зручно розподіляти завдання, виставляючи для них пріоритети. Крім того, платформа має дозволяти учасникам вести обговорення та прикріплювати потрібні документи безпосередньо всередині карток із завданнями. Передбачено також впровадження повного циклу перевірки виконаних робіт, у межах якого виконавець здає результат через спеціальну форму, ментор бере задачу на перевірку, після чого приймає рішення про прийняття задачі або повернення її на доопрацювання з обов'язковим коментарем причини. Такий підхід забезпечує прозорість процесу оцінювання та накопичення детальної історії роботи над кожним завданням.

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

До функціональних вимог системи належать також можливість обміну повідомленнями в межах окремих проєктів, ведення внутрішньої бази знань команди у форматі вікі-сторінок, перегляд календарного представлення задач з прив'язкою до дедлайнів та автоматичне формування звітності у форматах PDF та електронних таблиць Excel. Серед нефункціональних вимог визначено забезпечення безпеки даних користувачів через хешування паролів та використання маркерів доступу JWT, високу швидкодію інтерфейсу при роботі з великою кількістю задач, можливість горизонтального масштабування серверної частини, а також підтримку контейнеризації для спрощення процесу розгортання у середовищах різного типу [13].

Окремим важливим аспектом автоматизації є впровадження інструментів для безперервного моніторингу ефективності роботи команди. Платформа повинна збирати статистичні відомості про виконані, поточні та прострочені завдання, надаючи менторам наочні аналітичні панелі та розраховуючи внутрішні показники активності кожного учасника. Сформоване функціональне наповнення є логічним підґрунтям для подальшого проєктування логічної й фізичної моделей бази даних, вибору архітектурних шаблонів та написання вихідного коду системи.

1.2 Аналіз аналогів програмного продукту

Розробка платформи для організації групових проєктів потребує аналізу наявних рішень у цій сфері. Оскільки інструменти для управління командами та планування задач сьогодні дуже популярні як у навчанні, так і в бізнесі, корисно розібратися в роботі найвідоміших сервісів. Найбільш поширеними аналогами на ринку є Jira, Trello та Notion, кожен з яких пропонує свій підхід до організації командної роботи.

Першим відомим аналогом є сервіс Jira, який вважається професійним і дуже потужним інструментом для контролю завдань, особливо у сфері розробки програмного забезпечення та IT-індустрії загалом. Ця програма створена спеціально для управління великими, масштабними і складними проєктами, де робота ведеться за чіткими правилами. Вона дозволяє розбивати великі завдання

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

на менші підзадачі, організовувати роботу за спринтами та фіксувати кожен етап життєвого циклу розробки програмного продукту.

До головних переваг цієї системи можна віднести можливість дуже гнучко налаштовувати етапи виконання задач під конкретні потреби команди та чітко розподіляти ролі й права доступу між усіма користувачами. В платформі наявна функціональна аналітика. Програма вміє автоматично будувати різноманітні графіки, діаграми швидкості роботи, звіти про завантаженість учасників та точно оцінювати загальний прогрес проєкту, що дозволяє менторам чи керівникам легко контролювати стан справ.

Проте Jira має і мінуси, серед яких найголовнішим є дуже висока складність освоєння. Через величезну кількість налаштувань інтерфейс програми виглядає заплутаним і перевантаженим, тому звичайним користувачам доводиться довго вчитися, щоб просто почати роботу. Крім того, правильне налаштування сервісу під потреби конкретного проєкту займає багато часу та зусиль, що робить його занадто важким, надлишковим і незручним для звичайних студентських груп чи маленьких стартап-команд (рис. 1.1).

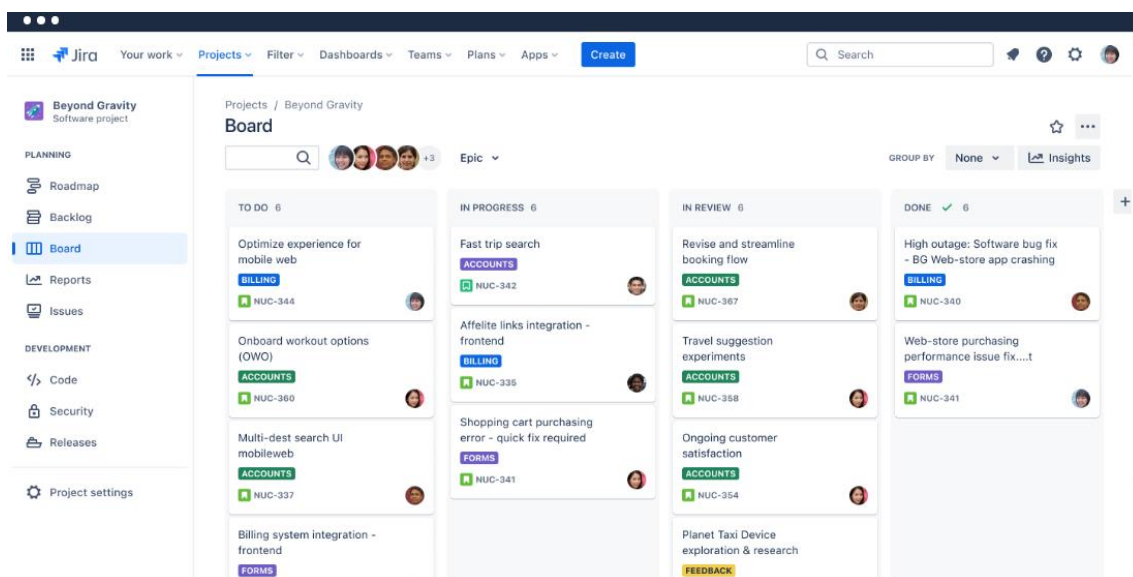


Рис. 1.1. Інтерфейс платформи Jira

Наступним популярним рішенням, яке пропонує зовсім інший, значно простіший підхід до планування, є платформа Trello. Вся робота в цьому сервісі побудована на основі візуальних канбан-дошок, що робить процес управління завданнями дуже наочним. Користувач може створювати тематичні дошки для

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савицький Р. С.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

різних проєктів, додавати туди списки, які відображають етапи роботи, та наповнювати їх картками завдань, які можна легко перетягувати мишкою в міру виконання.

Головною перевагою Trello є його максимально простий, чистий і зрозумілий інтерфейс, завдяки чому освоїти платформу та почати роботу можна без підготовки. Всередині кожної картки користувачі можуть зручно писати коментарі, ставити дедлайни, призначати відповідальних виконавців та прикріплювати потрібні документи. Це робить сервіс дуже зручним для швидкої організації невеликих команд чи координації простих навчальних завдань.

З іншого боку, надмірна простота платформи є одночасно і її головним недоліком. Trello абсолютно не підходить для ведення складних проєктів, оскільки в ньому не можна налаштувати залежності між завданнями або автоматично контролювати велику кількість процесів. Також у системі практично відсутні вбудовані інструменти для збору глибокої аналітики чи статистики, а для розширення можливостей користувачам доводиться підключати сторонні плагіни та платити за додаткові інтеграції (рис. 1.2).



Рис. 1.2. Інтерфейс платформи Trello

Третім поширеним аналогом є сервіс Notion, який працює не просто як менеджер завдань, а як універсальний цифровий простір для всієї команди. Він

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Савицький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

успішно поєднує в собі функції текстового редактора, великої бази знань, сховища для робочої документації та інструментів планування справ. Це дозволяє командам не лише фіксувати завдання, а й зберігати в одному місці всі супровідні матеріали, інструкції, конспекти та звіти, що дуже зручно для тривалих проєктів.

Головна перевага Notion полягає в його неймовірній гнучкості, адже користувачі можуть самостійно, як із конструктора, збирати сторінки будь-якої структури. Сервіс дозволяє створювати бази даних і відображати їх у різних виглядах: як звичайні таблиці, календарі, списки або ті ж самі канбан-дошки, легко пов'язуючи інформацію між собою. Це робить його одним із найкращих рішень для накопичення бази знань та ведення детальної проєктної документації (рис. 1.3).

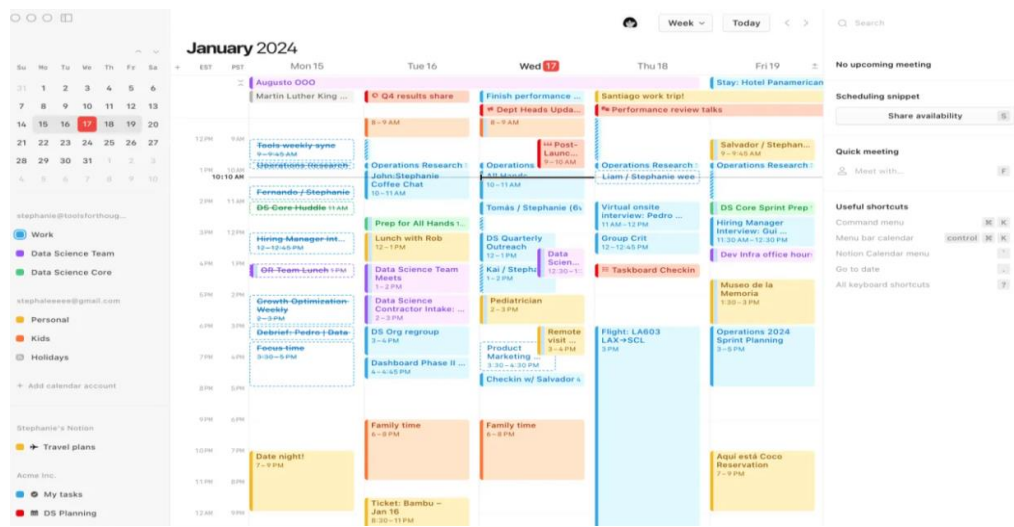


Рис. 1.3. Інтерфейс платформи Notion

Але як інструмент для щоденного та оперативного контролю за виконанням завдань Notion помітно програє спеціалізованим програмам. У ньому досить важко налаштувати автоматичні процеси, важко контролювати чіткі дедлайни або збирати точну статистику ефективності. Крім того, через відсутність жорстких правил оформлення сторінок, коли в Notion працює багато людей одночасно, робочий простір часто перетворюється на хаотичне накопичення інформації, в якому користувачам стає складно орієнтуватися.

Для узагальнення основних характеристик розглянутих платформ доцільно подати їх порівняння у вигляді таблиці (табл. 1.1). Це дає змогу більш

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савицький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

наочно оцінити можливості Trello, Jira та Notion у контексті управління проєктами.

Таблиця 1.1

Порівняння інструментів для управління проєктами

Характеристика	Trello	Jira	Notion	Squadly
Простота використання	висока	низька	середня	висока
Гнучкість налаштування	середня	висока	висока	середня
Управління знаннями та документацією	ні	потребує інтеграції	розширена	вбудована
Контроль зв'язків та залежностей між завданнями	обмежена	так	частково	так
Спеціалізовані ролі для навчання	ні	ні	ні	так
Комунікація та обговорення	обмежена	обмежена	обмежена	вбудована

Отже, проведене порівняння показує, що вебплатформа Squadly покликана поєднати ключові переваги розглянутих аналогів у межах єдиного, збалансованого середовища. Вона запозичує простоту та наочність управління завданнями у Trello, але водночас пропонує інструменти для збору аналітики, подібні до Jira, зберігаючи інтерфейс легким і зрозумілим для невеликих студентських команд. Крім того, на відміну від Notion, платформа дозволяє не просто зберігати документацію, а й здійснювати чіткий контроль за дедлайнами та щоденним прогресом виконання робіт. Таким чином, розроблюваний програмний продукт усуває потребу в комбінуванні кількох різних сервісів, забезпечуючи повний цикл організації групових проєктів в одному місці.

1.3 Вибір архітектури системи

Під час проєктування програмного продукту одним із ключових рішень є вибір архітектури системи, оскільки саме архітектурний підхід визначає структуру програмного продукту, спосіб взаємодії його компонентів, зручність супроводу та організацію обробки даних. Для платформи організації групових проєктів це має особливе значення, оскільки система повинна забезпечувати роботу з користувачами, проєктами, завданнями, засобами комунікації та аналітикою в межах єдиного узгодженого простору.

Під час проєктування системи було обрано клієнт-серверну архітектуру із чітким розподілом на серверний застосунок та клієнтський односторінковий застосунок SPA [9]. Цей підхід передбачає повне розмежування інтерфейсу користувача клієнтської частини на базі React та бізнес-логіки серверної частини на базі ASP.NET Core, які взаємодіють між собою за допомогою захищеного протоколу HTTPS шляхом обміну даними у форматі JSON. Для реалізації серверної частини було обрано монолітну структуру бекенду, де вся бізнес-логіка зосереджена в межах єдиного серверного застосунку, що значно спрощує розгортання, тестування і супровід системи, а також гарантує високу швидкість обміну даними між модулями [18].

Спрощену структурну схему обраної архітектури проєкту наведено на рисунку (рис. 1.4). На схемі показано, що користувач взаємодіє із системою через клієнтський вебінтерфейс React, який через API взаємодіє з монолітним бекендом [30]. Серверний застосунок Squadly містить у собі логічні модулі автентифікації, управління проєктами та завданнями, комунікації, сповіщень та аналітики. Зберігання структурованих даних здійснюється в реляційній базі даних PostgreSQL, а для збереження завантажених користувачами файлів та вкладень використовується локальне файлове сховище на сервері, представлене на схемі логічним блоком Blob Storage.

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савицький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

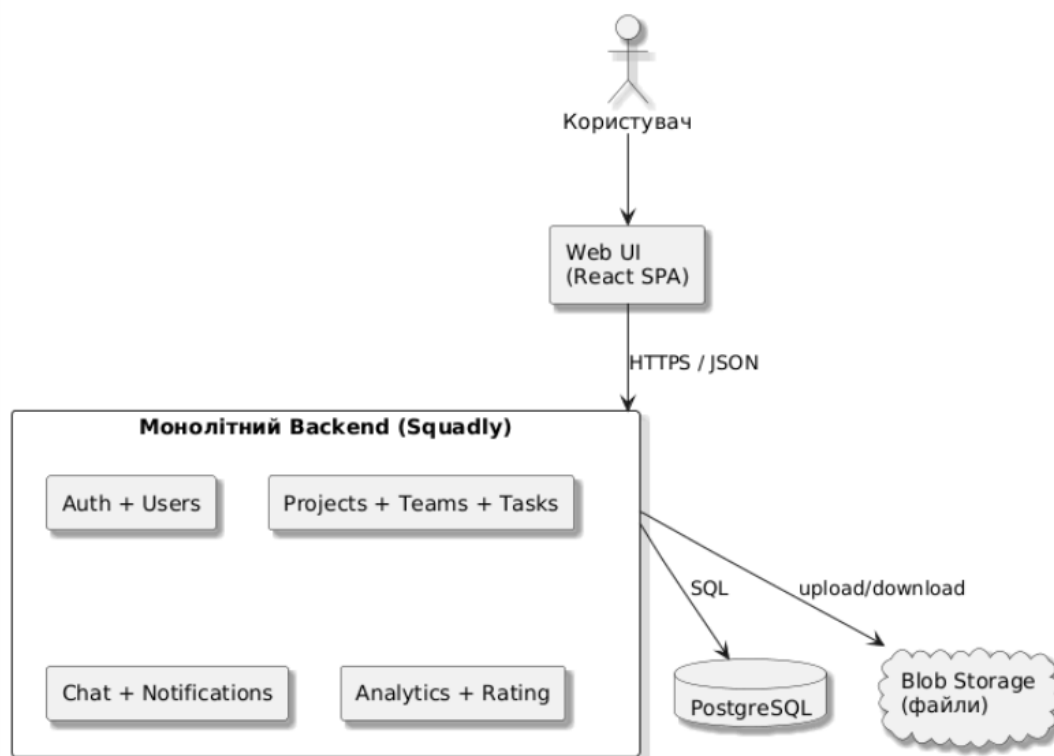


Рис. 1.4. Спрощена схема клієнт-серверної архітектури платформи

Для кращого розуміння принципу функціонування платформи доцільно розглянути схему взаємодії компонентів під час виконання користувацьких операцій (рис. 1.5). За такого сценарію користувач ініціює дію через вебінтерфейс (наприклад, створює проєкт або завдання). Клієнтська частина формує асинхронний HTTPS-запит, що містить тіло запиту у форматі JSON, і передає його до серверного API. Серверна частина Squadly обробляє запит, виконує необхідні правила бізнес-логіки та надсилає SQL-запити до бази даних PostgreSQL для збереження або зчитування інформації. Після успішного виконання операції сервер формує JSON-відповідь із результатом, яка повертається клієнту, де React динамічно оновлює стан сторінки та відображає зміни користувачеві без перезавантаження веббраузера. Також слід зазначити, що такий підхід мінімізує обсяг переданих мережею даних, суттєво підвищує швидкість реагування інтерфейсу та забезпечує безперервну взаємодію користувача з додатком. Окрім стандартних HTTP-запитів, для обміну даними в реальному часі використовується постійне двостороннє з'єднання на базі технології WebSockets. Це дозволяє серверній частині миттєво доставляти

оновлення іншим активним учасникам проєкту без необхідності ручного оновлення вебсторінки.

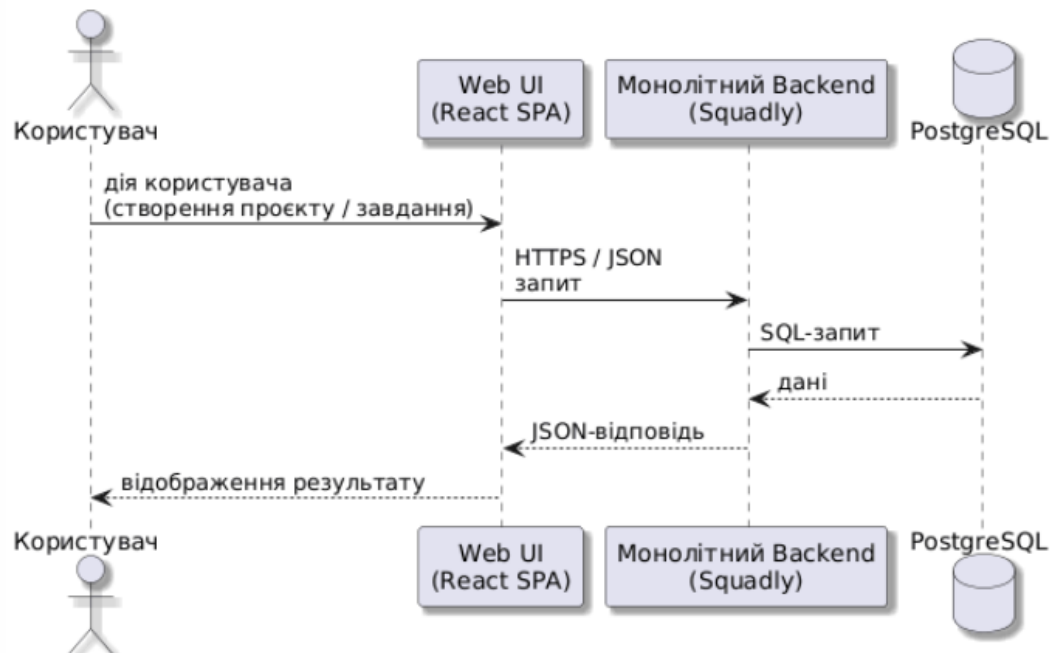


Рис. 1.5. Спрощена схема взаємодії користувача з програмним продуктом

З метою забезпечення гнучкості, масштабованості та зручності тестування коду, внутрішня логіка серверної частини побудована за принципами чистої архітектури. Програмний код бекенду розділений на чотири ізольовані шари, кожен з яких виконує строго визначену роль, а залежності між ними спрямовані виключно всередину.

Ядром системи виступає доменний шар Squadly.Domain, який визначає базові сутності предметної області, такі як користувачі, проєкти, завдання, коментарі та рейтингові показники. Доменний рівень є повністю автономним і не містить залежностей від баз даних, сторонніх бібліотек чи фреймворків. Навколо доменного шару будується аплікаційний рівень Squadly.Application, що відповідає за реалізацію безпосередніх бізнес-сценаріїв використання платформи, описує структури передачі даних DTO, правила валідації та інтерфейси для взаємодії з іншими компонентами інформаційної системи.

Технічні деталі та взаємодія із зовнішнім середовищем винесені в шар інфраструктури Squadly.Infrastructure. Він містить конкретні реалізації

інтерфейсів доступу до бази даних PostgreSQL за допомогою технології Entity Framework Core, налаштування JWT-авторизації для забезпечення безпеки, а також логіку взаємодії з локальним файловим сховищем сервера. Точкою входу в серверний застосунок виступає презентаційний шар Squadly.API, представлений ASP.NET Core контролерами, які обробляють вхідні HTTPS-запити від клієнтської частини React, координують маршрутизацію та налаштовують механізм впровадження залежностей.

Такий комплексний підхід гарантує чітке розмежування відповідальності шарів, спрощує тестування і підвищує загальну надійність та якість супроводу програмного забезпечення [16].

1.4. Обґрунтування вибору інструментальних засобів

Вибір інструментальних засобів для розробки вебплатформи для організації групових проєктів ґрунтується на їхній ефективності, масштабованості, швидкодії та здатності забезпечити стабільну роботу додатку в багатокористувацькому режимі. Серверну частину системи доцільно реалізувати на базі кросплатформового фреймворку ASP.NET Core, що дозволяє створити швидкодіючий REST API [1]. Клієнтська частина будується як SPA за допомогою бібліотеки React та мови програмування TypeScript, що гарантує інтерактивність та стійкість інтерфейсу до помилок. Зберігання інформації здійснюється в реляційній базі даних PostgreSQL, що забезпечує підтримку цілісності даних, тоді як безпека користувачів контролюється через авторизаційні JWT-токени, а розгортання проєкту уніфікується засобами контейнеризації Docker.

Для розробки серверної частини платформи було обрано сучасний кросплатформовий фреймворк ASP.NET Core, що базується на об'єктно-орієнтованій мові програмування C#. Його вибір зумовлений надвисокою продуктивністю обробки HTTP-запитів, що є критично важливим для забезпечення швидкої роботи багатокористувацької системи в реальному часі. Використання асинхронного програмування за допомогою конструкцій `async/await` дозволяє серверу ефективно обробляти велику кількість одночасних

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

запитів без блокування потоків виконання. Модульна структура фреймворку дозволяє розділити серверну логіку на окремі компоненти, суттєво спрощуючи розробку та супровід коду відповідно до принципів чистої архітектури. Вбудовані механізми автентифікації, авторизації та впровадження залежностей надають готові інструменти для реалізації безпечного доступу та конфігурації платформи. Додатковою перевагою є кросплатформеність, завдяки якій серверний застосунок може функціонувати та розгортатися в різних операційних системах, включаючи сімейства Windows, Linux та macOS. Таким чином, фреймворк ASP.NET Core є оптимальним інструментом для побудови REST API, оскільки він забезпечує швидку взаємодію з базою даних, високу надійність бізнес-процесів та легкість модульного тестування серверної логіки[10, 15].

З метою спрощення взаємодії з реляційною базою даних на серверному рівні використовується об'єктно-реляційне відображення за допомогою технології Entity Framework Core. EF Core дозволяє розробнику працювати з даними як з об'єктами мови C#, автоматизуючи генерацію SQL-запитів та взаємодію з СУБД. EF Core суттєво прискорює процес розробки завдяки автоматичному відстеженню змін в об'єктах та механізму міграцій, який дає змогу гнучко керувати схемою бази даних та оновлювати її без втрати існуючої інформації. Використання підходу Code First дозволяє описувати структуру таблиць безпосередньо у вигляді класів доменного шару системи, що забезпечує чітку відповідність коду бекенду та структури бази даних [7].

Клієнтська частина платформи розробляється з використанням JavaScript-бібліотеки React у поєднанні з мовою програмування TypeScript, яка підтримує статичну типізацію [3, 4]. Компонентний підхід, покладений в основу React, дозволяє будувати інтерфейс із незалежних повторно використовуваних компонентів, що зменшує дублювання коду та спрощує розробку складних інтерактивних сторінок, таких як канбан-дошка, календарі чи аналітичні панелі. Завдяки механізму віртуального представлення забезпечується динамічне оновлення даних в інтерфейсі без повного перезавантаження сторінки, що значно покращує взаємодію користувача з платформою. Використання TypeScript додає

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

статичну типізацію, що дозволяє виявляти потенційні помилки ще на етапі написання коду, підвищує надійність інтерфейсу та полегшує його супровід при розширенні функціоналу [27]. Збірка клієнтської частини здійснюється за допомогою сучасного інструменту Vite, який забезпечує надзвичайно швидке гаряче оновлення модулів під час розробки та високу швидкість завантаження додатку користувачем [29].

Для організації надійного збереження та обробки структурованої інформації застосовується реляційна система керування базами даних PostgreSQL. Вона відзначається високою стабільністю, відповідністю стандартам ACID та можливістю ефективно працювати з великою кількістю взаємопов'язаних сутностей, включаючи профілі користувачів, проекти, завдання, коментарі, файлові вкладення та аналітичні дані. PostgreSQL надає потужні засоби для виконання складних аналітичних SQL-запитів, фільтрації та об'єднання таблиць, що є важливим для побудови детальної звітності та розрахунку рейтингів учасників. Завдяки надійності механізмів транзакцій система гарантує збереження цілісності даних навіть за умов високої інтенсивності роботи користувачів. Використання типу даних UUID як первинних ключів у базі даних забезпечує унікальність ідентифікаторів сутностей у межах всієї розподіленої системи та запобігає конфліктам при генерації нових записів на клієнтському або серверному рівнях [21].

Для автентифікації користувачів у системі застосовано стандарт JWT, що реалізує безсесійну модель авторизації. Завдяки цьому серверу не потрібно зберігати стан сесій у пам'яті чи звертатися до бази даних під час кожного запиту, що позитивно впливає на масштабованість платформи. Усередині токена міститься службова інформація ідентифікатор користувача та його роль, яка підписана секретним ключем сервера, тому підробити авторизаційні дані на стороні клієнта неможливо. Такий механізм забезпечує швидку й безпечну перевірку прав доступу до захищених кінцевих точок REST [14].

Для реалізації функцій реального часу, таких як миттєве отримання сповіщень та обмін повідомленнями в адміністративному чаті, застосовується

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

бібліотека ASP.NET Core SignalR. Вона спрощує процес додавання вебсокетової взаємодії до додатку, автоматично керуючи з'єднаннями та обираючи оптимальний транспортний протокол WebSockets, залежно від можливостей браузера користувача. SignalR дозволяє серверу ініціювати відправку повідомлень клієнтам у реальному часі, що забезпечує миттєве відображення нових подій на фронтенді без необхідності періодичного надсилання запитів з боку клієнта [2].

З метою полегшення розгортання програмного забезпечення та мінімізації проблем із сумісністю оточень використовується платформа контейнеризації Docker. Вона дозволяє упакувати серверний додаток, базу даних та інші компоненти в ізольовані контейнери, що містять усі необхідні залежності та системні бібліотеки. Завдяки цьому забезпечується абсолютна ідентичність роботи системи як на локальних комп'ютерах розробників, так і на продуктивних серверах, незалежно від типу базової операційної системи. Контейнеризація спрощує налаштування взаємодії між сервером API, базою даних PostgreSQL та файловим сховищем, а також створює надійну основу для автоматизації процесів безперервної інтеграції та розгортання платформи [6, 12].

Висновки до першого розділу

У першому розділі було проведено детальний аналіз предметної області, що дозволило визначити головні вимоги до майбутньої платформи для організації групових проєктів. Дослідження робочих процесів допомогло виявити основні проблеми, з якими стикаються команди, та обґрунтувати необхідність створення єдиного цифрового простору для спільної роботи.

Під час вивчення вже існуючих програмних продуктів було розглянуто та порівняно популярні сервіси Jira, Trello та Notion. На основі цього порівняння вдалося виділити їхні ключові переваги та недоліки, а також чітко сформулювати перелік функцій, які необхідно реалізувати у нашому власному програмному продукті.

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савицький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

З урахуванням усіх визначених вимог було обґрунтовано вибір монолітної клієнт-серверної архітектури системи. Крім того, сформовано оптимальний набір інструментальних засобів і технологій, які стануть надійною основою для подальшої технічної реалізації вебплатформи Squadly.

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБОРІЄНТОВАНОЇ ПЛАТФОРМИ

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

На етапі проєктування веборієнтованої платформи Squadly було визначено варіанти використання, які описують способи взаємодії користувачів із системою. Формування варіантів використання здійснювалося на основі попереднього аналізу предметної області групової роботи, функціональних вимог до інструментів планування та особливостей організації спільної діяльності у невеликих командах. Основним призначенням системи є об'єднання в одному середовищі тих процесів, які в традиційних підходах розділені між кількома окремими сервісами: керування проєктами, постановка та контроль виконання задач, перевірка результатів роботи, ведення внутрішньої документації та обмін повідомленнями.

У межах платформи реалізовано дворівневу модель ролей, що відрізняє Squadly від більшості систем-аналогів і дозволяє точніше відображати реальні робочі відносини в команді. На першому рівні визначається глобальна роль користувача в системі, яка задається при реєстрації або змінюється адміністратором. Глобальна роль визначає базові права у системі, зокрема можливість створювати нові проєкти. На другому рівні визначається локальна роль користувача в конкретному проєкті, яка формується організатором при додаванні учасника. Один і той самий користувач у різних проєктах може мати різні локальні ролі, що відповідає реальним умовам роботи у різних командах.

Адміністратор системи має найвищі привілеї та виконує функції технічного супроводу платформи. Він переглядає список усіх зареєстрованих користувачів, редагує їхні особисті дані, призначає або змінює глобальні ролі, видаляє облікові записи разом із пов'язаною інформацією. Адміністратор також керує всіма проєктами в системі, що дозволяє йому втручатися в роботу будь-якого проєкту в разі потреби. Окремим напрямком діяльності адміністратора є робота з персональними зверненнями користувачів через вбудований чат підтримки, який дозволяє оперативно реагувати на запитання та проблеми, а за

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				24
Змн.	Арк.	№ докум.	Підпис	Дата		

необхідності одразу з вікна чату змінювати роль користувача, надаючи йому додаткові права.

У межах конкретного проєкту організатор виконує роль керівника команди: він створює проєкт, формує його опис і параметри, додає учасників із призначенням локальних ролей, ставить задачі та визначає виконавців. Окрім контролю за загальним ходом робіт, організатор генерує звіти у форматах PDF та Excel, наповнює вікі-розділ і може переводити проєкт у стан завершеного чи архівованого. Для оперативного коригування робочого процесу йому також доступне перетягування задач на канбан-дошці від імені будь-якого учасника.

Ментор поєднує у проєкті функції перевіряючого та консультанта, опрацьовуючи задачі, які виконавці здали на перевірку. Узявши конкретну задачу собі, він блокує її для інших менторів, що унеможлиблює дублювання роботи. За результатами перевірки ментор або приймає задачу, тоді виконавцеві автоматично нараховуються бали відповідно до пріоритету, або повертає її на доопрацювання з обов'язковим коментарем щодо причини. Паралельно ментор веде приватні нотатки про кожного підопічного, доступні лише йому, для фіксації спостережень за роботою учасників.

Учасник проєкту працює над задачами, які призначив йому організатор. У межах своєї задачі він переглядає опис, прикріплені файли та історію коментарів, змінює статус перетягуванням на канбан-дошці, додає файли й спілкується з командою через коментарі та чат проєкту. Завершивши роботу, учасник оформлює здачу через окрему форму, де описує виконане, додає посилання на репозиторій або демонстрацію, зазначає витрачений час і підтверджує самоперевірку. У разі повернення задачі на доопрацювання він бачить причину в інтерфейсі та готує повторну здачу, а кількість задач обліковується окремо для формування звітних показників.

На рис. 2.1 наведено діаграму варіантів використання системи, яка відображає взаємодію всіх ролей з основними функціональними блоками платформи.

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савицький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

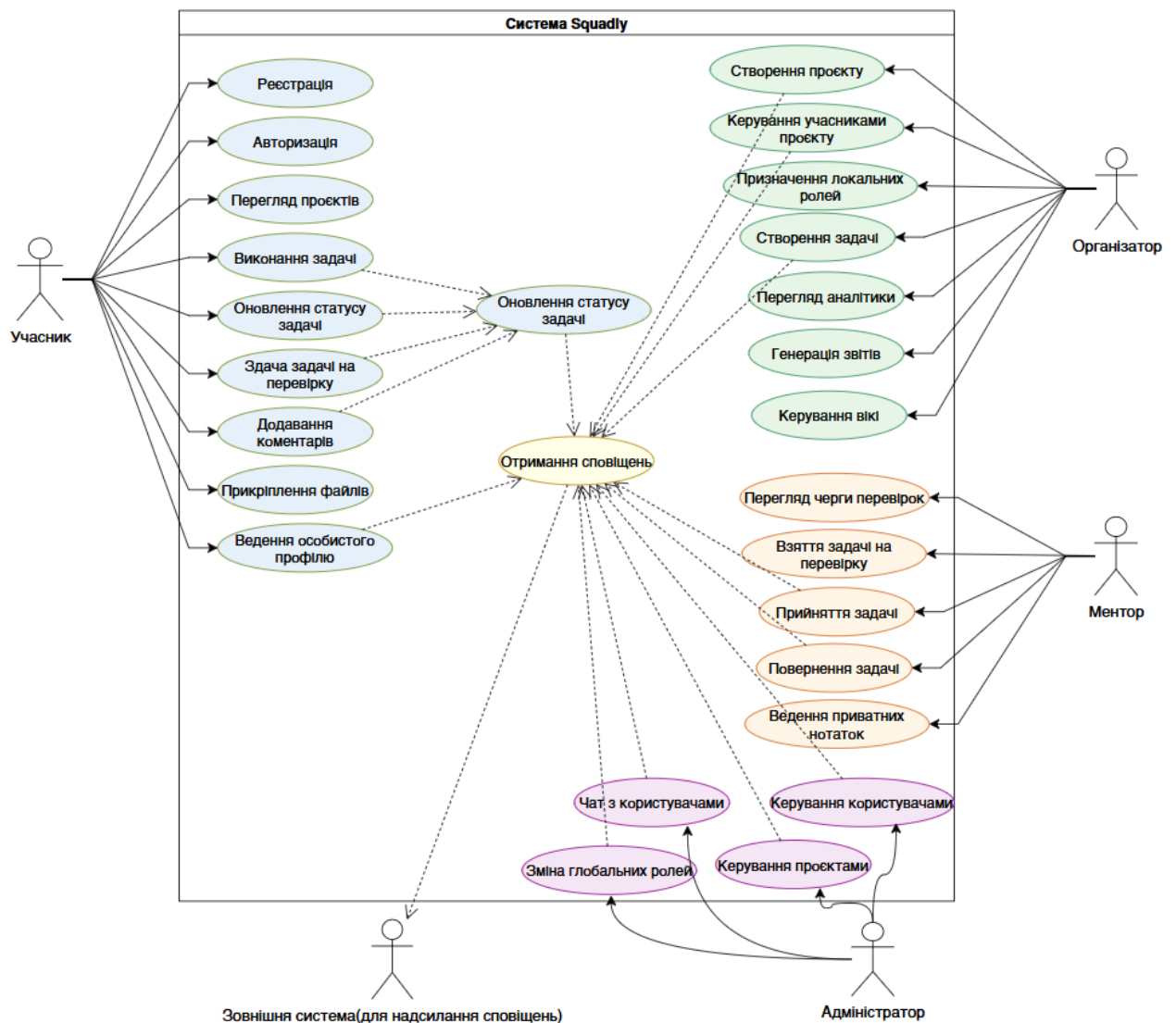


Рис. 2.1. Діаграма варіантів використання системи Squadly

Розглянуті варіанти використання визначають ключові сценарії взаємодії та є основою для побудови об'єктно-орієнтованої моделі вебплатформи. Наступним етапом проєктування є опис базових класів, які реалізують технічний функціонал системи та визначають зв'язки між ними. Побудована модель слугує фундаментом для формування реляційної структури бази даних та написання коду серверної частини.

Об'єктно-орієнтована модель системи Squadly розроблена на базі принципів чистої архітектури, де всі базові сутності зосереджені всередині ізолюваного доменного шару [8, 28]. Для уникнення дублювання загального коду та дотримання принципу DRY, всі класи успадковують властивості від базового абстрактного класу BaseEntity [11]. Цей компонент інкапсулює в собі

обов'язкові системні атрибути, серед яких унікальний ідентифікатор об'єкта, дата його додавання до системи, час останнього редагування, а також спеціальний прапорець для реалізації механізму м'якого видалення інформації.

Центральне місце в архітектурі посідає клас User, який описує зареєстрованого користувача системи та зберігає його ім'я, прізвище, контактний email, хеш пароля за алгоритмом BCrypt, глобальну роль, біографію та посилання на профільний аватар [22]. Основною організаційною одиницею виступає клас Project, який містить детальні відомості про створені проекти (назву, опис, пріоритет, дедлайни, масив JSON-тегів та колірний маркер для візуального оформлення в UI). Зв'язок «багато до багатьох» між користувачами та проектними просторами організовано за допомогою асоціативної сутності ProjectMembership, яка додатково фіксує унікальну локальну роль учасника (Учасник, Ментор чи Організатор) у конкретній команді.

Робочий процес та рух завдань описуються класом TaskItem, який відстежує стан виконання задачі за принципом скінченного автомата де етапи «До виконання», «В роботі», «На перевірці», «Потребує доопрацювання» та «Виконано». Історія перевірок та звіти виконавців фіксуються через сутність TaskSubmission, де кожна спроба здачі отримує власний порядковий номер, опис виконаної роботи, витрачений час у годинах та посилання на репозиторії. Для забезпечення повноцінної комунікації та збереження супровідних матеріалів використовуються класи Comment – це робочі обговорення всередині завдань, TaskAttachment – файлові вкладення обсягом до 50 мегабайтів та ProjectMessage, який відповідає за миттєвий обмін повідомленнями в чаті проекту через технологію SignalR.

Навчальну спрямованість платформи Squadly, що відрізняє її від типових систем керування проектами, реалізують класи Rating і MentorNote. Клас Rating автоматично обліковує накопичені бали користувачів, формуючи глобальну рейтингову таблицю, а MentorNote дає змогу викладачеві вести конфіденційні нотатки про хід роботи студентів. Наповнення внутрішньої бази знань проекту описує клас WikiPage, за допомогою якого організатори створюють

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

багатосторінкову документацію. Зворотний зв'язок і швидке інформування забезпечують класи Notification та AdminChatMessage, останній з яких ізолює повідомлення в чаті технічної підтримки з адміністратором.

На рис. 2.2 наведено діаграму класів системи, на якій показано основні сутності доменного шару, їхні атрибути та зв'язки між ними.

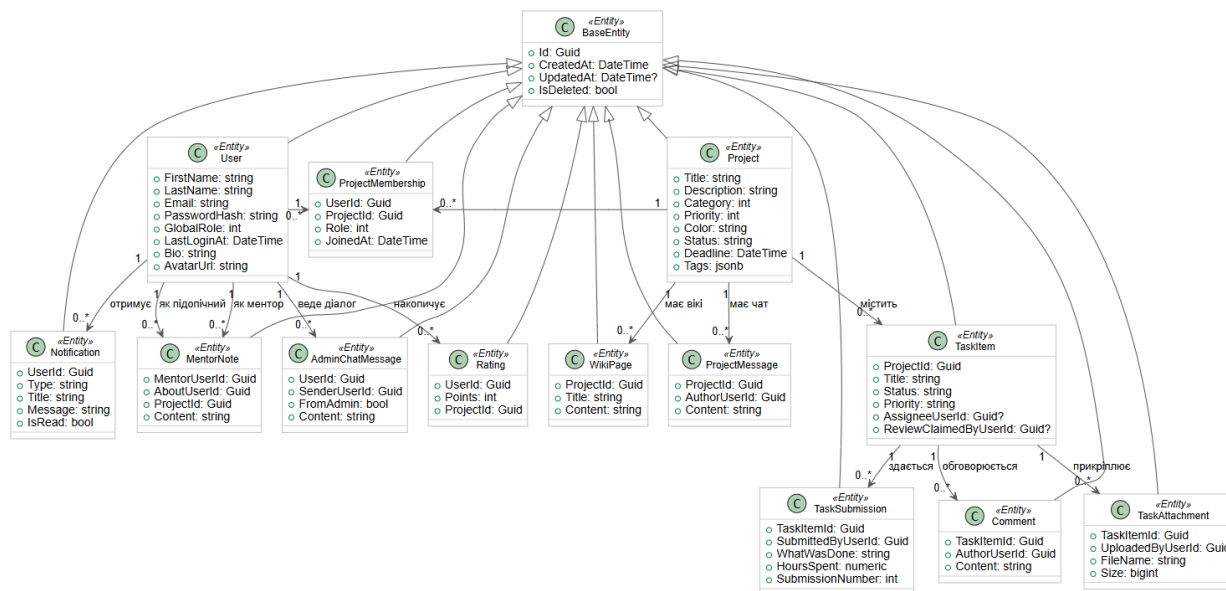


Рис. 2.2. Діаграма класів доменної моделі системи

Спроектована об'єктно-орієнтована модель концептуально відображає предметну область розроблюваної платформи та забезпечує повну програмну реалізацію всіх раніше визначених варіантів використання системи. Чіткий розподіл обов'язків між класами та ізоляція бізнес-сутностей у межах єдиного доменного шару дозволяють досягти гнучкості та високого рівня тестопридатності серверного коду відповідно до паттернів чистої архітектури. Описані сутності, типи їхніх атрибутів та встановлені між ними зв'язки з урахуванням реляційності визначають правила логічної цілісності даних.

2.2. Розробка бази даних системи

Для зберігання та опрацювання даних у платформі Squadly спроектовано реляційну базу даних на основі системи керування базами даних PostgreSQL 16. Вибір PostgreSQL зумовлено сукупністю властивостей, які точно відповідають характеру задач платформи. Серед них підтримка повноцінних транзакцій з рівнями ізоляції, наявність типу JSONB для зберігання слабоструктурованих

даних, ефективна робота з паралельними з'єднаннями, відсутність обмежень за обсягом таблиць у межах розумних значень, активна спільнота та регулярні оновлення безпеки. Інтеграція з серверною частиною забезпечується через Entity Framework Core 8 за допомогою провайдера Npgsql, що дозволяє описувати структуру таблиць у вигляді C# класів сутностей, формувати міграції автоматично та виконувати запити з використанням типобезпечного синтаксису [25]. Окремою перевагою PostgreSQL є його повна сумісність із контейнерним середовищем Docker, що дозволяє розгортати базу даних в ізольованому контейнері без встановлення на хостову операційну систему. Це спрощує перенесення системи між середовищами розробки, тестування та продакшну, гарантуючи ідентичну поведінку бази на всіх етапах. Крім того, відкритий вихідний код PostgreSQL та відсутність ліцензійних відрахувань роблять його економічно вигідним рішенням для навчальних і комерційних проєктів.

Структура бази даних відображає основні процеси платформи: керування користувачами і ролями, ведення проєктів і членств, постановку і контроль виконання задач, фіксацію детальних здач задач на перевірку, ведення коментарів і файлових вкладень, формування рейтингу, відображення сповіщень, спілкування у чатах проєктів та з адміністрацією, ведення приватних менторських нотаток, наповнення вікі-розділу проєкту. Загалом база даних налічує чотирнадцять основних таблиць, які пов'язані між собою зовнішніми ключами із заданою поведінкою при видаленні. Для тих сутностей, які логічно не повинні безповоротно видалятися (проєкти), реалізовано механізм м'якого видалення через прапорець IsDeleted.

Кожна таблиця бази даних має первинний ключ типу UUID, який створюється автоматично при додаванні нового запису. Усі дати в системі зберігаються у форматі UTC, що дозволяє уникнути помилок при роботі користувачів з різних часових поясів. Детальний опис кожної таблиці із зазначенням полів, типів даних та зв'язків між ними наведено на рис. 2.3

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савицький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

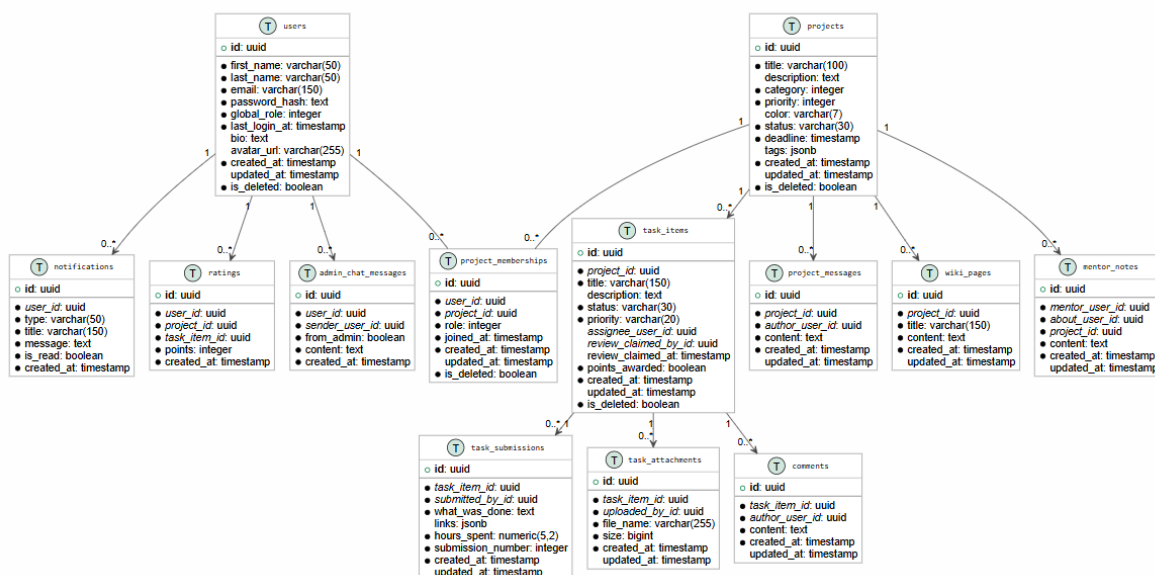


Рис. 2.3. Структура таблиць бази даних

Оснoву бази даних складає таблиця users (табл. 2.1), у якій зберігаються облікові записи всіх користувачів системи. Кожен запис містить особисті дані користувача, його глобальну роль, дані для входу та посилання на аватар. Електронна адреса є унікальною і використовується як логін при вході в систему. Глобальна роль зберігається як рядок і може мати значення Admin, Organizer або User. Хеш пароля формується за алгоритмом BCrypt, що унеможливорює зворотний підбір пароля.

Таблиця 2.1

Опис таблиці “users”

Назва поля	Тип даних	PK	FK	Опис
id	uuid	+	—	Унікальний ідентифікатор користувача
first_name	varchar(50)	—	—	Ім'я користувача
last_name	varchar(50)	—	—	Прізвище користувача
email	varchar(150)	—	—	Електронна адреса (унікальний індекс)
password_hash	text	—	—	Хеш пароля (BCrypt)
global_role	integer	—	—	0 - User, 1 - Organizer, 2 - Admin

продовж. табл. 2.1

last_login_at	timestamp	–	–	Час останнього входу в систему
bio	text	–	–	Короткі відомості про себе
avatar_url	varchar(255)	–	–	Посилання на зображення профілю
created_at	timestamp	–	–	Дата реєстрації
updated_at	timestamp	–	–	Дата останнього оновлення
is_deleted	boolean	–	–	Ознака м'якого видалення

Таблиця projects (табл. 2.2) описує всі проєкти, створені в системі. Вона містить назву, опис, категорію, пріоритет, колір, термін виконання та набір тегів. Категорія та пріоритет зберігаються як цілі числа, що відповідають значенням переліків у програмному коді. Поле tags зберігає масив тегів у форматі JSONB.

Таблиця 2.2

Опис таблиці “projects”

Назва поля	Тип даних	PK	FK	Опис
id	uuid	+	–	Унікальний ідентифікатор проєкту
title	varchar(100)	–	–	Назва проєкту
description	text	–	–	Опис проєкту
category	integer	–	–	0 - Development, 1 - Design і так далі
priority	integer	–	–	0 - Low, 1 - Medium, 2 - High
color	varchar(7)	–	–	Колірний маркер картки
status	varchar(30)	–	–	Стан (Active, Completed, Archived)
start_date	timestamp	–	–	Дата початку проєкту
deadline	timestamp	–	–	Кінцевий термін проєкту
goal	text	–	–	Мета проєкту

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

tags	jsonb	–	–	Масив тегів у форматі JSON
wiki_content	text	–	–	Вміст вікі-сторінки проєкту
created_by_user_id	uuid	–	+	Творець проєкту (users.id)
created_at	timestamp	–	–	Дата створення
updated_at	timestamp	–	–	Дата оновлення
is_deleted	boolean	–	–	Ознака м'якого видалення

Таблиця `project_memberships` (табл. 2.3) зв'язує користувачів із проєктами та зберігає локальну роль кожного учасника. Один користувач може брати участь у кількох проєктах з різними ролями. На пару полів `user_id` та `project_id` накладено унікальний індекс, що не дозволяє додати одного користувача до проєкту двічі. При видаленні користувача чи проєкту всі членства видаляються автоматично.

Таблиця 2.3

Опис таблиці “project_memberships”

Назва поля	Тип даних	PK	FK	Опис
id	uuid	+	–	Унікальний ідентифікатор членства
user_id	uuid	–	+	Користувач (users.id)
project_id	uuid	–	+	Проект (projects.id)
role	integer	–	–	0 - Participant, 1 - Organizer, 2 - Mentor
joined_at	timestamp	–	–	Дата додавання до проєкту
created_at	timestamp	–	–	Дата створення запису
updated_at	timestamp	–	–	Дата оновлення
is_deleted	boolean	–	–	Ознака видалення

Таблиця `task_items` (табл. 2.4) зберігає всі завдання в межах проєктів. Кожне завдання має назву, опис, статус, пріоритет, термін виконання та виконавця. Статус може мати одне з п'яти значень: `ToDo`, `InProgress`, `InReview`,

NeedsRevision, Done. Окремі поля відповідають за процес перевірки ментором той, хто взяв на перевірку, хто прийняв рішення та чи нараховано бали.

Таблиця 2.4

Опис таблиці “task_items”

Назва поля	Тип даних	РК	FK	Опис
id	uuid	+	–	Унікальний ідентифікатор задачі
title	varchar(150)	–	–	Назва завдання
description	text	–	–	Опис завдання
status	varchar(30)	–	–	Стан (ToDo, InProgress, InReview, NeedsRevision, Done)
priority	varchar(20)	–	–	Пріоритет (Low, Medium, High)
due_date	timestamp	–	–	Кінцевий термін виконання
project_id	uuid	–	+	Проект (projects.id)
assignee_user_id	uuid	–	+	Виконавець (users.id)
review_claimed_by_user_id	uuid	–	+	Ментор, що перевіряє
review_claimed_at	timestamp	–	–	Час взяття на перевірку
reviewed_by_user_id	uuid	–	+	Ментор, що прийняв рішення
review_comment	text	–	–	Коментар ментора при поверненні
points_awarded	boolean	–	–	Ознака нарахування балів
created_at	timestamp	–	–	Дата створення
updated_at	timestamp	–	–	Дата оновлення
is_deleted	boolean	–	–	Ознака м'якого видалення

Таблиця task_submissions (табл. 2.5) зберігає історію всіх задач завдань на перевірку. Кожна спроба має свій порядковий номер submission_number, який збільшується автоматично. Виконавець описує виконану роботу, вказує витрачений час та додає посилання на результати.

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

Таблиця 2.5

Опис таблиці “task_submissions”

Назва поля	Тип даних	PK	FK	Опис
id	uuid	+	–	Унікальний ідентифікатор задачі
task_item_id	uuid	–	+	Завдання (task_items.id)
submitted_by_user_id	uuid	–	+	Виконавець (users.id)
what_was_done	text	–	–	Опис виконаної роботи
links	jsonb	–	–	Масив посилань на результати
hours_spent	numeric(5,2)	–	–	Кількість витрачених годин
submission_number	integer	–	–	Порядковий номер спроби
created_at	timestamp	–	–	Дата задачі
updated_at	timestamp	–	–	Дата оновлення
is_deleted	boolean	–	–	Ознака видалення

Таблиця comments (табл. 2.6) зберігає коментарі учасників до завдань. Коментарі дозволяють вести обговорення в межах конкретної задачі та не залежать від чату проєкту.

Таблиця 2.6

Опис таблиці “comments”

Назва поля	Тип даних	PK	FK	Опис
id	uuid	+	–	Унікальний ідентифікатор коментаря
content	text	–	–	Текст коментаря
task_item_id	uuid	–	+	Завдання (task_items.id)
author_user_id	uuid	–	+	Автор (users.id)
created_at	timestamp	–	–	Дата написання
updated_at	timestamp	–	–	Дата редагування
is_deleted	boolean	–	–	Ознака видалення

Таблиця task_attachments (табл. 2.7) містить відомості про файли, завантажені до завдань. Для кожного файлу зберігається його оригінальна назва, внутрішня назва у сховищі сервера, розмір у байтах та MIME-тип

Таблиця 2.7

Опис таблиці “task_attachments”

Назва поля	Тип даних	РК	FK	Опис
id	uuid	+	–	Унікальний ідентифікатор вкладки
task_item_id	uuid	–	+	Завдання (task_items.id)
uploaded_by_user_id	uuid	–	+	Хто завантажив (users.id)
file_name	varchar(255)	–	–	Назва файлу у сховищі
original_file_name	varchar(255)	–	–	Оригінальна назва файлу
content_type	varchar(100)	–	–	MIME-тип файлу
file_size	bigint	–	–	Розмір файлу в байтах
created_at	timestamp	–	–	Дата завантаження
updated_at	timestamp	–	–	Дата оновлення
is_deleted	boolean	–	–	Ознака видалення

Таблиця project_messages (табл. 2.8) містить історію повідомлень чату проєкту. Повідомлення доставляються учасникам у реальному часі через SignalR.

Таблиця 2.8

Опис таблиці “project_messages”

Назва поля	Тип даних	РК	FK	Опис
id	uuid	+	–	Унікальний ідентифікатор повідомлення
project_id	uuid	–	+	Проект (projects.id)
author_user_id	uuid	–	+	Автор (users.id)
content	text	–	–	Текст повідомлення
created_at	timestamp	–	–	Дата відправлення
updated_at	timestamp	–	–	Дата оновлення
is_deleted	boolean	–	–	Ознака видалення

Таблиця notifications (табл. 2.9) зберігає сповіщення про події в системі, а саме призначення завдань, нові коментарі, результати перевірки. Кожне сповіщення містить посилання на пов'язаний об'єкт related_id для швидкого переходу.

Таблиця 2.9

Опис таблиці “notifications”

Назва поля	Тип даних	РК	FK	Опис
id	uuid	+	–	Унікальний ідентифікатор сповіщення
user_id	uuid	–	+	Отримувач (users.id)
type	varchar(50)	–	–	Тип сповіщення
title	varchar(150)	–	–	Заголовок
message	text	–	–	Вміст сповіщення
is_read	boolean	–	–	Ознака прочитання
read_at	timestamp	–	–	Дата прочитання
related_id	uuid	–	–	Ідентифікатор пов'язаного об'єкта
related_type	varchar(50)	–	–	Тип пов'язаного об'єкта
created_at	timestamp	–	–	Дата створення
updated_at	timestamp	–	–	Дата оновлення
is_deleted	boolean	–	–	Ознака видалення

Таблиця ratings (табл. 2.10) фіксує нарахування балів користувачам. Кожен запис містить кількість балів points, причину нарахування reason та посилання на пов'язаний об'єкт. Сума балів відображається в рейтинговій таблиці.

Таблиця 2.10

Опис таблиці “ratings”

Назва поля	Тип даних	РК	FK	Опис
id	uuid	+	–	Унікальний ідентифікатор нарахування
user_id	uuid	–	+	Користувач (users.id)

продовж; табл. 2.10

points	integer	–	–	Кількість балів
reason	text	–	–	Причина нарахування
related_id	uuid	–	–	Ідентифікатор пов'язаного об'єкта
related_type	varchar(50)	–	–	Тип пов'язаного об'єкта
created_at	timestamp	–	–	Дата нарахування
updated_at	timestamp	–	–	Дата оновлення
is_deleted	boolean	–	–	Ознака видалення

Таблиця mentor_notes (табл. 2.11) містить приватні нотатки менторів про підопічних у межах конкретного проєкту. Ці записи видимі лише автору нотатки і використовуються для фіксації спостережень за роботою учасника.

Таблиця 2.11

Опис таблиці “mentor_notes”

Назва поля	Тип даних	РК	FK	Опис
id	uuid	+	–	Унікальний ідентифікатор нотатки
mentor_user_id	uuid	–	+	Автор-ментор (users.id)
about_user_id	uuid	–	+	Підопічний (users.id)
project_id	uuid	–	+	Проект (projects.id)
content	text	–	–	Текст нотатки
created_at	timestamp	–	–	Дата створення
updated_at	timestamp	–	–	Дата оновлення
is_deleted	boolean	–	–	Ознака видалення

Таблиця wiki_pages (табл. 2.12) описує сторінки документації проєкту. Організатори можуть створювати та редагувати сторінки, інші учасники мають доступ лише для читання. Поле order визначає послідовність відображення сторінок у меню.

Таблиця 2.12

Опис таблиці “wiki_pages”

Назва поля	Тип даних	PK	FK	Опис
id	uuid	+	–	Унікальний ідентифікатор сторінки
project_id	uuid	–	+	Проект (projects.id)
created_by_user_id	uuid	–	+	Автор (users.id)
title	varchar(150)	–	–	Заголовок сторінки
content	text	–	–	Вміст сторінки
order	integer	–	–	Порядковий номер у меню
created_at	timestamp	–	–	Дата створення
updated_at	timestamp	–	–	Дата редагування
is_deleted	boolean	–	–	Ознака видалення

Таблиця admin_chat_messages (табл. 2.13) зберігає повідомлення чату між користувачами та адміністрацією. Поле from_admin дозволяє розрізняти, хто написав повідомлення – користувач чи адміністратор.

Таблиця 2.13

Опис таблиці “admin_chat_messages”

Назва поля	Тип даних	PK	FK	Опис
id	uuid	+	–	Унікальний ідентифікатор повідомлення
user_id	uuid	–	+	Власник діалогу (users.id)
sender_user_id	uuid	–	+	Відправник (users.id)
from_admin	boolean	–	–	Ознака відправлення адміністратором
content	text	–	–	Текст повідомлення
is_read	boolean	–	–	Ознака прочитання

		Пастухов О. К.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

read_at	timestamp	–	–	Час прочитання
created_at	timestamp	–	–	Дата відправлення
updated_at	timestamp	–	–	Дата оновлення
is_deleted	boolean	–	–	Ознака видалення

При видаленні проєкту автоматично видаляються всі пов'язані задачі, членства, повідомлення чату, нотатки менторів та вікі-сторінки. При видаленні задачі видаляються її коментарі, задачі та вкладення. Управління схемою бази даних реалізовано через систему міграцій Entity Framework Core. Кожна зміна структури супроводжується створенням файлу міграції, який автоматично застосовується при запуску застосунку на основі записів у службовій таблиці __EFMigrationsHistory. Це забезпечує синхронізацію стану бази з версією коду без ручного виконання SQL-сценаріїв.

2.3 Проєктування та реалізація алгоритмів роботи системи

Створення проєкту є ключовим етапом роботи з платформою для будь-якої нової команди. Процес починається з того, що користувач відкриває майстер створення проєкту. Цей інструмент реалізований як зручне модальне діалогове вікно, яке візуально розділене на три послідовні кроки для спрощення сприйняття інформації. На першому етапі організатор вводить базові дані, назву та загальний опис проєкту. Після переходу на другий крок система пропонує обрати категорію з переліку семи запропонованих варіантів, визначити пріоритет виконання та вибрати колірний маркер із палітри з восьми кольорів для кращої візуальної ідентифікації картки. На третьому кроці користувач вказує дату початку робіт, кінцевий дедлайн, головну мету та додає необхідні пошукові теги. Важливим елементом контролю є те, що перехід до наступного етапу здійснюється після коректного заповнення всіх обов'язкових полів. Такий архітектурний підхід суттєво мінімізує ймовірність введення некоректних даних. Детальний алгоритм цього процесу наведено на рис. 2.4.

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

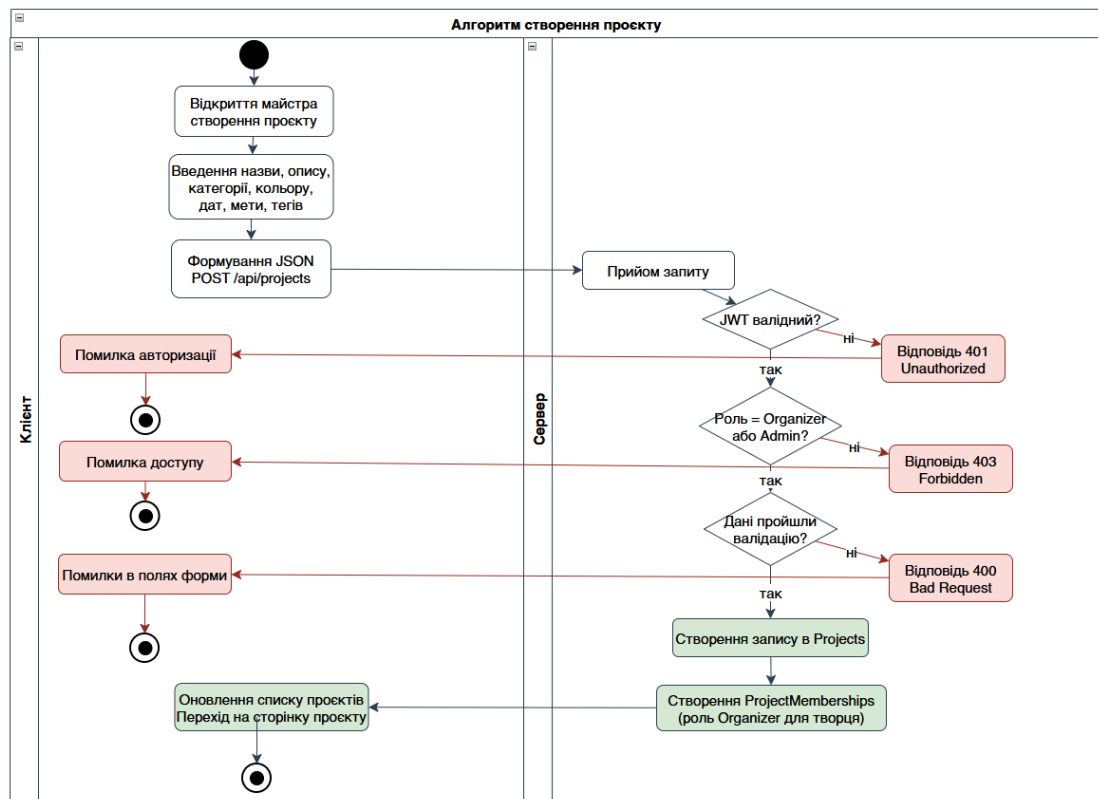


Рис. 2.4. Діаграма діяльності створення проєкту

Після натиснення кнопки створення клієнтська частина формує JSON-запит до серверної кінцевої точки та передає всі введені дані. Серверна сторона спочатку перевіряє автентифікацію користувача через JWT-токен, потім перевіряє його глобальну роль. Створення проєктів дозволено лише користувачам з ролями Організатор або Адмін. Для звичайних користувачів сервер повертає відповідь з кодом 403. Якщо перевірка ролей пройдена, сервер виконує валідацію переданих даних та перевіряє довжину назви, формат дат, наявність відомої категорії, валідність кольору. У разі помилки валідації клієнт отримує детальний опис проблемного поля. Якщо дані коректні, сервер створює запис у таблиці projects, одночасно створює запис у таблиці projectMemberships з роллю Organizer для творця проєкту, та повертає клієнту створений об'єкт. Клієнт оновлює список проєктів і переходить на сторінку щойно створеного проєкту.

Життєвий цикл задачі є центральним механізмом платформи. Він реалізований як скінченний автомат із п'ятьма станами та переходами, які мають жорсткі правила. Діаграма станів задачі наведена на рис. 2.5.

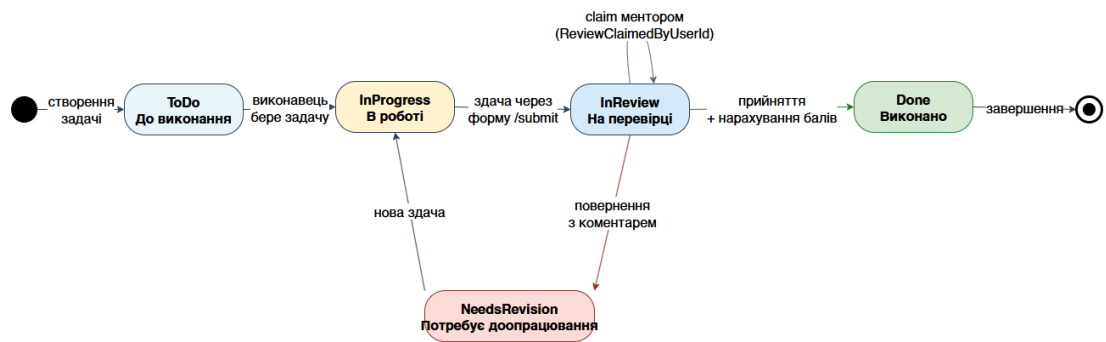


Рис. 2.5. Діаграма станів задачі

Початковим станом будь-якої задачі є «До виконання» тобто ToDo, у якому задача перебуває після створення організатором. З цього стану задача переходить у «В роботі» тобто InProgress у момент, коли виконавець бере задачу до виконання, перетягуючи її на канбан-дошці в колонку «В роботі». Альтернативно цей перехід може виконати організатор від імені виконавця.

Зі стану «В роботі» задача переходить у «На перевірці» тобто InReview лише через спеціальну дію задачі задачі. Цей перехід не може бути виконаний простим перетягуванням, оскільки потребує оформлення повноцінного звіту виконавця через окрему форму. У стані «На перевірці» задача очікує дії ментора. Жоден учасник проєкту, окрім менторів та організаторів, не може змінити стан задачі.

Перший ментор, який обирає задачу на перевірку, фіксує цей факт у полі ReviewClaimedByUserId, що автоматично блокує задачу для інших менторів проєкту. Натомість вони бачать у інтерфейсі позначку з іменем ментора, який наразі перевіряє задачу.

Зі стану «На перевірці» можливі два сценарії розвитку. У разі схвалення ментор натискає кнопку прийняття, і задача переходить у стан «Виконано» (Done): сервер автоматично нараховує виконавцеві бали за відповідну задачу, у таблиці tasks встановлює прапорець PointsAwarded для запобігання повторному нарахуванню та створює новий запис у таблиці Ratings. Якщо ж робота потребує виправлень, ментор повертає задачу у стан «Потребує доопрацювання» (NeedsRevision), причому текстовий коментар із поясненням причини є обов'язковим і зберігається у полі ReviewComment.

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

Зі стану «Потребує доопрацювання» задача переходить назад у «В роботі» після того, як виконавець оформлює нову задачу. Кількість таких циклів не обмежена. Зі стану «Виконано» задача не може бути повернута в будь-який інший стан, що забезпечує цілісність історичних даних у звітах.

Здача задачі на перевірку є окремим алгоритмом, який реалізує ключовий бізнес-процес платформи. Алгоритм наведений на рис. 2.6.

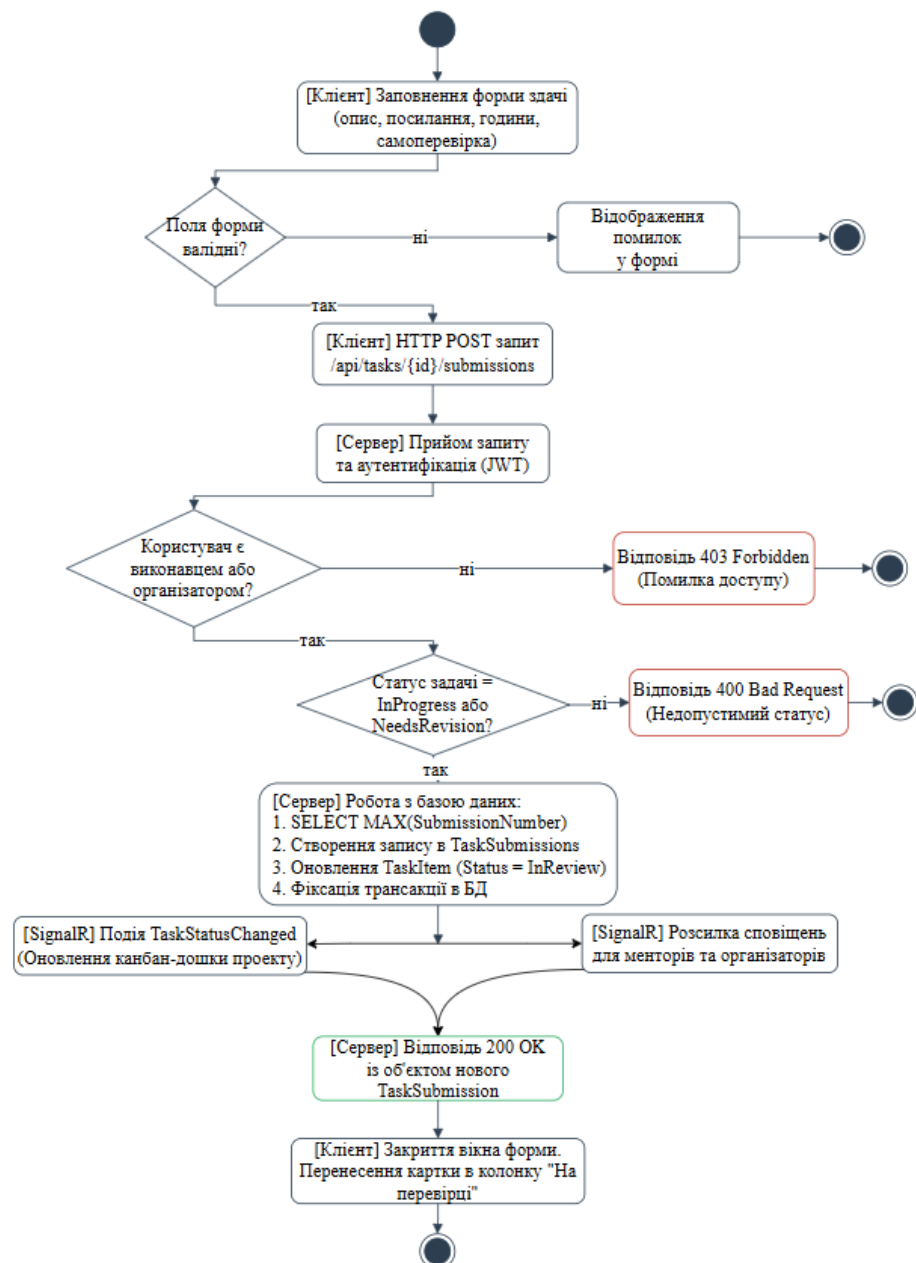


Рис. 2.6. Діаграма діяльності здачи задачі на перевірку

Процес починається з того, що виконавець відкриває модальне вікно деталей задачі та натискає кнопку «Здати на перевірку». Запускається форма

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

здачі, у якій користувач описує виконану роботу у текстовому полі з мінімальною довжиною десять символів. Додаткові поля включають масив посилань на репозиторій, демонстраційні матеріали чи зовнішні документи, та кількість витрачених годин. Перед відправкою клієнт перевіряє валідність полів, а саме довжина опису, реалістичність значення годин від 0,1 до 24, наявність підтвердження самоперевірки.

Після відправки клієнтська частина надсилає JSON-запит до серверного ендпоінту. Сервер перевіряє автентифікацію, перевіряє, що поточний користувач є виконавцем задачі або організатором проєкту, перевіряє поточний статус задачі. Задача дозволена лише для задач у статусах «В роботі» або «Потребує доопрацювання». Для інших статусів сервер повертає помилку. У разі позитивної перевірки сервер визначає наступний порядковий номер задачі через запит максимального значення поля SubmissionNumber серед існуючих задач цієї задачі. Створюється новий запис у таблиці TaskSubmissions з обчисленим порядковим номером, статус задачі змінюється на «На перевірці», у разі повернення з доопрацювання поле ReviewComment очищується. Сервер надсилає повідомлення через SignalR-хаб усім підключеним учасникам проєкту, що задача змінила статус. Усі ментори і організатори проєкту отримують сповіщення про нову задачу для перевірки.

Опрацювання задачі ментором є завершальним алгоритмом циклу. Він наведений на рис. 2.7.

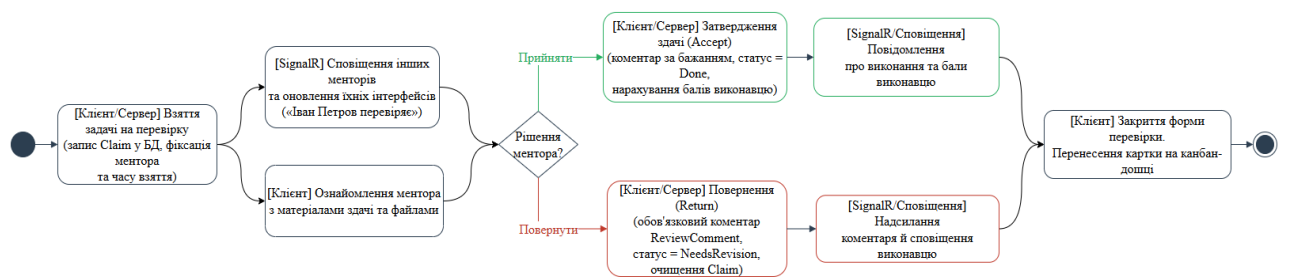


Рис. 2.7. Діаграма діяльності опрацювання задачі ментором

Процес починається з того, що ментор відкриває сторінку центру перевірок або сторінку проєкту й бачить задачу зі статусом «На перевірці». Якщо задачу ще ніхто не взяв на перевірку, поряд із нею відображається кнопка «Взяти на

перевірку». Натиснення цієї кнопки виконує запит до сервера, який встановлює поточного користувача як значення поля ReviewClaimedByUserId та фіксує час у ReviewClaimedAt. Сервер також надсилає SignalR-повідомлення усім менторам та організаторам про взяття задачі на перевірку, після чого інтерфейс інших користувачів оновлюється: замість кнопки взяття вони бачать позначку з іменем ментора, який перевіряє задачу.

Після цього ментор знайомиться з матеріалами здачі: переглядає опис виконаної роботи, відкриває посилання та прикріплені файли, читає попередні коментарі. Прийнявши рішення, ментор натискає одну з двох кнопок. У разі прийняття опціонально заповнюється коментар, після чого виконується запит до сервера. Сервер перевіряє, що ментор є тим, хто взяв задачу на перевірку, або організатором проекту, після чого встановлює статус «Виконано» і ReviewedByUserId, у разі першого прийняття нараховує бали через сервіс IRatingService та встановлює прапорець PointsAwarded, а також надсилає SignalR-повідомлення про зміну статусу й сповіщення виконавцеві.

У разі повернення ментор обов'язково заповнює коментар із поясненням причини повернення без нього кнопка повернення неактивна. Запит до сервера встановлює статус «Потребує доопрацювання», зберігає коментар у полі ReviewComment, очищає поля claim та надсилає повідомлення виконавцеві. У своєму інтерфейсі виконавець бачить червоний індикатор повернення задачі з причиною доопрацювання та може одразу запустити нову задачу.

2.4 Реалізація функціоналу платформи для організації групових проєктів

Важливою частиною розробленої системи є модуль для завантаження виконавцями результатів своєї роботи та їх подальшої перевірки менторами. Описаний процес об'єднує сутності бази даних, як завдання (TaskItem), історія задач (TaskSubmission) та профілі користувачів (User). Спільна робота моделей дозволяє автоматично зберігати дати перевірки, вести облік витраченого часу та рахувати кількість спроб задачі завдання. Логіку взаємодії та особливості коду

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

можна детально розглянути на прикладі контролера завдань і супутніх сервісів збереження даних.

На програмному рівні описаний процес послідовно проходить крізь усі шари платформи. Транспортний шар представлено контролерами Web API з авторизацією через JWT-токени, прикладний через DTO-моделями та валідацією запитів, а інфраструктурний відповідає за роботу з базою даних через Entity Framework Core і розсилання сповіщень засобами SignalR [19]. Доступ контролера до бази даних, сервісів автентифікації, системи сповіщень, логіки рейтингів та вузла веб-сокетів організовано через механізм впровадження залежностей. Завдяки такому пошаровому поділу досягається слабка пов'язаність компонентів із чітким розмежуванням їхніх зон відповідальності.

Взаємодія з клієнтським застосунком реалізована в класі TasksController, який наслідує стандартний ControllerBase. Для відправки виконаної роботи використовується метод HTTP POST, який приймає унікальний ідентифікатор завдання та об'єкт SubmitTaskDto. Цей об'єкт містить текстовий опис виконаної роботи, список посилань на результати розробки, кількість витрачених годин і підтвердження пройденої самоперевірки.

```
[HttpPost("{id}/submit")]
```

```
public async Task<IActionResult> Submit(Guid id, [FromBody] SubmitTaskDto dto)
```

```
{
```

```
    var userId = GetUserId();
```

```
    var task = await _db.Tasks.FirstOrDefaultAsync(t => t.Id == id);
```

```
    if (task == null) return NotFound(new { message = "Завдання не знайдено" });
```

```
    try { await _auth.EnsureMemberAsync(task.ProjectId, userId); }
```

```
    catch (UnauthorizedAccessException ex) { return StatusCode(403, new { message = ex.Message }); }
```

```
    if (task.AssigneeUserId != userId)
```

```
        return StatusCode(403, new { message = "Здавати задачу може тільки призначений виконавець" });
```

Перед збереженням запису сервер проводить кілька обов'язкових перевірок. Система визначає особу користувача, перевіряє факт закріплення завдання за ним, а також аналізує його поточний статус, який повинен мати

		Пастухов О. К.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

значення InProgress або NeedsRevision. Також обов'язковою є згода користувача з пунктом про самоперевірку. Якщо всі фільтри пройдено успішно, система рахує номер поточної спроби шляхом додавання одиниці до кількості попередніх задач і створює новий екземпляр сутності TaskSubmission.

```
var submissionNumber = await _db.TaskSubmissions.CountAsync(s => s.TaskItemId ==
task.Id) + 1;

var submission = new TaskSubmission
{
    TaskItemId = task.Id,
    SubmittedByUserId = userId,
    WhatWasDone = dto.WhatWasDone.Trim(),
    Links = NormalizeLinks(dto.Links),
    HoursSpent = dto.HoursSpent,
    SubmissionNumber = submissionNumber
};

_db.TaskSubmissions.Add(submission);

task.Status = "InReview";
task.ReviewClaimedByUserId = null;
task.ReviewClaimedAt = null;
task.UpdatedAt = DateTime.UtcNow;
await _db.SaveChangesAsync();
```

Окремої уваги заслуговує допоміжний метод NormalizeLinks. Він відповідає за очищення та форматування списку посилань перед їх збереженням у базу даних. Це запобігає потраплянню неправильних адрес у систему. Під час збереження змін паралельно запускаються два процеси. Перший розсилає персональні сповіщення для менторів та організаторів проєкту. Другий створює подію TaskStatusChanged у каналі SignalR. Повідомлення через веб-сокети надсилаються адресно тільки тим користувачам, які переглядають проєкт. Такий підхід значно економить мережевий трафік і дозволяє миттєво оновити положення картки завдання на канбан-дошці.

Наступний крок передбачає закріплення завдання ментором під власну перевірку. Сервер перевіряє роль користувача і блокує можливість одночасного

		Пастухов О. К.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

взяття роботи кількома експертами. Для запобігання конфліктам система перевіряє поле ReviewClaimedByUserId. Якщо два ментори одночасно намагаються перевірити одну роботу, сервер відхиляє другий запит і повертає повідомлення з іменем ментора, який першим почав перевірку. Якщо все добре, у базу записуються дані про експерта та час початку опрацювання. Відразу після цього подія TaskReviewClaimed через веб-сокети робить кнопку перевірки неактивною для інших колег.

Фінальним етапом є винесення вердикту у вигляді схвалення або повернення роботи на виправлення. У разі затвердження статус завдання змінюється на успішний стан Done. Важливим моментом є надійний механізм нарахування балів. Перевірка прапорця PointsAwarded гарантує відсутність подвійного нарахування балів виконавцю навіть у разі випадкового повторного запиту через збій мережі. Водночас тимчасові дані про закріплення задачі видаляються з бази, оскільки процес перевірки успішно завершено.

Альтернативним варіантом є виявлення помилок або невідповідностей вимогам. У такому випадку ментор відхиляє виконану роботу. Для цього обов'язково потрібно додати текстовий коментар із детальним поясненням причин відхилення та вказівками щодо виправлення. Серверна логіка суворо вимагає наявності такого зворотного зв'язку. Без коментаря запит буде просто відхилено. Після цього сервер змінює статус завдання на стан NeedsRevision, зберігає текст коментаря і повністю очищує інформацію про тимчасове закріплення завдання.

Після успішного збереження всіх даних сервер розсилає оновлений статус користувачам через SignalR. Це дозволяє виконавцю миттєво побачити коментар, виправити помилки та здати роботу ще раз. Водночас після зняття блокування завдання знову стає доступним для перевірки будь-яким іншим ментором проєкту.

Описаний підхід створює надійний цикл взаємодії між учасниками команди. Використання Entity Framework Core гарантує збереження всіх даних під час зміни статусів та нарахування балів. Робота з технологією SignalR

		Пастухов О. К.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

забезпечує зручне користування системою завдяки швидкому оновленню інтерфейсу без перезавантаження сторінок браузера. Поділ логіки на окремі блоки робить програмний код зрозумілим, а застосування асинхронних функцій гарантує швидку роботу платформи [17].

Висновки до другого розділу

У другому розділі виконано проєктування вебплатформи для організації групових проєктів. Визначено варіанти використання системи, виділено ролі користувачів та сформовано вимоги до її функціонування. На основі сформованих вимог розроблено об'єктно-орієнтовану структуру платформи, яка відображає основні сутності предметної області та зв'язки між ними.

Спроектовано реляційну базу даних та сформовано структуру таблиць для зберігання інформації про облікові записи, проєкти, завдання й історію їх здач. Розроблена модель даних забезпечує виконання бізнес-логіки платформи, а реалізована політика каскадних видалень підтримує цілісність інформації та автоматичне очищення залежних записів при видаленні сутностей.

Змодельовано алгоритми роботи системи у вигляді діаграм діяльності. Проаналізовано процеси створення проєкту, заповнення форми здачі завдання виконавцем та цикл опрацювання роботи ментором, що дозволило деталізувати взаємодію між клієнтською та серверною частинами додатка.

Описано програмну реалізацію ключових функціональних можливостей. На прикладі модуля перевірки завдань продемонстровано роботу API-контролерів, механізми обробки конкурентних запитів та логіку нарахування балів. Для збереження даних використано Entity Framework Core, а для двостороннього зв'язку та оновлення інтерфейсів інтегровано технологію SignalR.

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ ДЛЯ ОРГАНІЗАЦІЇ ГРУПОВИХ ПРОЄКТІВ

3.1 Архітектура розгортання та налаштування параметрів платформи

Встановлення та налаштування параметрів системи передбачає підготовку серверного середовища, розгортання серверної та клієнтської частин, а також налагодження мережових зв'язків між компонентами. Вебплатформа для організації групових проєктів реалізована як сучасний розподілений вебдодаток. До її складу входять серверна частина API на базі фреймворку ASP.NET Core, клієнтська частина у вигляді SPA, реляційна база даних для надійного збереження інформації та система обміну повідомленнями на базі технології SignalR.

Використання контейнерного підходу для розгортання платформи має низку практичних переваг над традиційним встановленням програмного забезпечення прямо на операційну систему сервера. Контейнеризація забезпечує абсолютно однакове середовище для роботи програми незалежно від обраного фізичного або віртуального обладнання. Оскільки кожен контейнер містить власний ізольований набір бібліотек і налаштувань, це повністю вирішує відому проблему розбіжностей між локальним комп'ютером розробника та робочим сервером. Важливим додатковим плюсом такого рішення є зручна можливість незалежного масштабування окремих компонентів системи в разі збільшення навантаження. Для спрощення розгортання основна конфігурація серверного середовища описана у файлах проєкту та зберігається разом із вихідним кодом у системі контролю версій Git. Це дозволяє відтворити середовище розгортання у будь-який момент часу та забезпечує стабільну поведінку системи в різних режимах експлуатації. Файл налаштувань для контейнерів керує порядком запуску сервісів та зв'язками між ними. Він також визначає правила перевірки готовності кожного компонента перед стартом наступного. Наприклад, серверна частина програми почне працювати тільки тоді, коли база даних буде повністю готова приймати запити [5]. Такий підхід дозволяє надійно уникнути збоїв підключення під час початкового запуску системи.

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савицький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

Розгортання платформи включає кілька послідовних кроків. На першому кроці на цільовому середовищі готується інфраструктура, встановлюється середовище виконання Docker та система управління базами даних. На другому кроці отримується вихідний код системи з репозиторію та налаштовуються параметри оточення. На третьому кроці виконується запуск серверної частини всередині ізольованих контейнерів. Після запуску бекенду автоматично застосовуються міграції бази даних через інструментарій Entity Framework Core, що формує актуальну структуру таблиць. Завершальним кроком є збірка та розгортання клієнтської частини, статичні файли якої передаються користувачеві через окремий вебсервер, що забезпечує оптимальну швидкість завантаження інтерфейсу.

Окремої уваги заслуговує процес автоматичного застосування міграцій бази даних під час старту серверного застосунку. Реалізований механізм перевіряє поточний стан схеми бази даних шляхом звернення до спеціальної службової таблиці, яка фіксує історію застосованих змін структури. У разі виявлення розбіжностей між фактичним станом бази та очікуваною структурою, визначеною в коді програми, система послідовно застосовує всі відсутні міграції в хронологічному порядку їх створення. Такий підхід виключає необхідність ручного виконання SQL-сценаріїв адміністратором під час кожного оновлення платформи та гарантує синхронізацію версії бази даних із версією програмного коду в будь-якому середовищі розгортання, включаючи локальне, тестове чи робоче.

Фізична архітектура розробленої платформи базується на вдосконаленій трирівневій моделі з використанням контейнеризації. Зв'язки між основними апаратними вузлами, середовищами виконання та протоколи обміну даними продемонстровано на діаграмі розгортання (рис. 3.1).

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

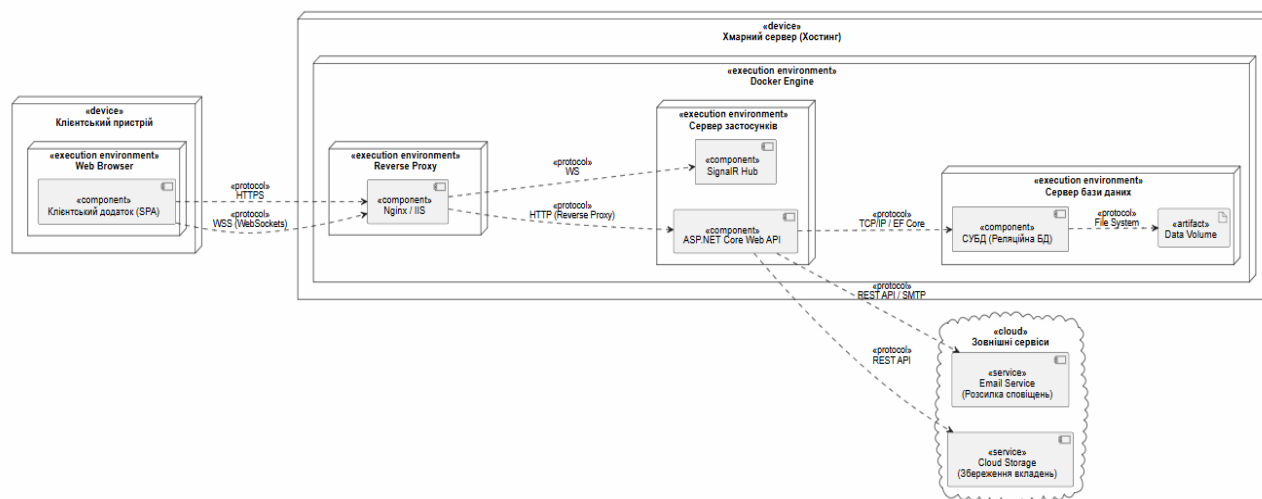


Рис. 3.1. Діаграма розгортання

Структуру, наведену на діаграмі розгортання, можна поділити на кілька складових. Клієнтська частина представлена пристроєм користувача, де у браузері працює SPA. Цей додаток бере на себе візуалізацію та обробку інтерфейсу, а з бекендом спілкується по захищених протоколах HTTPS та WebSockets.

Серверна інфраструктура розгорнута на базі Docker Engine. Усі зовнішні запити спочатку потрапляють на вебсервер Nginx, який працює як зворотний проксі та перенаправляє трафік на основний сервер застосунків. Використання окремого проксі-сервера перед застосунком має кілька важливих переваг: Nginx ефективно обробляє статичні файли клієнтської частини без залучення ресурсів серверної логіки, забезпечує балансування навантаження у разі масштабування системи та додає додатковий рівень безпеки, приховуючи внутрішню структуру серверних компонентів від прямого доступу з мережі Інтернет. Вже безпосередньо сервер застосунків обробляє HTTP-запити за допомогою ASP.NET Core Web API і керує вебсокет-з'єднаннями через SignalR. Для зберігання інформації про користувачів, команди і завдання використовується реляційна база даних. Щоб інформація не втрачалася при перезапуску чи оновленні контейнерів, дані фізично винесені в окремий Data Volume. Окрім внутрішніх компонентів, архітектура також передбачає взаємодію із зовнішніми

сервісами для email-розсилок та хмарного збереження прикріплених до завдань файлів.

Мережева взаємодія між компонентами системи організована через внутрішню віртуальну мережу, створену засобами оркестрації контейнерів. Кожен сервіс отримує власне мережеве ім'я, за яким інші компоненти системи можуть звертатися до нього незалежно від фактичної IP-адреси контейнера, що може змінюватися при кожному перезапуску. Така схема адресації значно спрощує конфігурацію з'єднань між сервісами та підвищує надійність системи в умовах динамічного перерозподілу ресурсів середовищем виконання контейнерів.

Щодо конфігурації, то загальні налаштування системи, такі як таймаути та базові URL-адреси, прописані у файлі appsettings.json. А от чутливі дані, зокрема рядки підключення до бази даних та секретні ключі для JWT-токенів, передаються виключно через змінні середовища операційної системи сервера. Такий підхід є стандартною практикою для забезпечення безпеки та виключає потрапляння конфіденційної інформації до системи контролю версій разом із вихідним кодом проєкту. Окремо налаштовується перелік дозволених джерел запитів для механізму міжсайтового доступу CORS, що обмежує можливість звернення до серверного API виключно з довірених клієнтських адрес.

Режим роботи платформи залежить від системної змінної ASPNETCORE_ENVIRONMENT. Якщо проєкт запускається в режимі розробки, автоматично підключається інтерактивна документація Swagger, яка дозволяє тестувати API прямо з браузера. У робочому середовищі ця документація та всі тестові маршрути вимикаються задля безпеки. Для випуску нової версії платформи достатньо стягнути останні зміни з коду, перевірити змінні середовища і перезібрати Docker-контейнери. Процес оновлення виконується без втрати накопичених даних, оскільки інформація бази даних та завантажені файли зберігаються в окремих томах, незалежних від життєвого циклу самих контейнерів застосунку.

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савицький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

Для забезпечення безперервної доступності платформи передбачено механізм автоматичного перезапуску контейнерів у разі їхнього аварійного завершення роботи. Параметр політики перезапуску контролює поведінку системи оркестрації контейнерів: якщо будь-який із сервісів припиняє роботу внаслідок неочікуваної помилки, відповідний контейнер автоматично перезапускається без необхідності ручного втручання адміністратора. Це підвищує загальну відмовостійкість платформи та зменшує час простою системи у випадку непередбачуваних збоїв.

3.2. Структура інтерфейсу вебплатформи

Клієнтська частина платформи Squadly побудована як односторінковий застосунок SPA на базі React із маршрутизацією через React Router [24, 26]. Усі сторінки, крім авторизації та реєстрації, захищені компонентом AuthGuard, який перенаправляє неавторизованого користувача на форму входу. Верхня навігаційна панель фіксована і містить логотип системи, основні посилання (Головна, Проекти, Мої задачі, Рейтинг), випадаюче меню з додатковими розділами (Аналітика, Звіти, Календар, Менторство), пошук по системі, іконку сповіщень із лічильником непрочитаних повідомлень та аватар користувача з випадаючим меню профілю.

Першим екраном, який бачить новий користувач, є сторінка реєстрації (рис. 3.2). Форма розташована по центру екрана у білому блоці із заокругленими краями та легкою тінню. Поля імені та прізвища розміщені в одному рядку для економії простору, нижче знаходяться поля електронної пошти та пароля. Поле пароля використовує прихований ввід. Під формою розміщено текстове посилання для переходу на сторінку авторизації для тих, хто вже має обліковий запис. Після заповнення всіх полів та натискання кнопки «Зареєструватись» система створює новий обліковий запис і перенаправляє користувача на форму входу.

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

Рис. 3.2. Сторінка реєстрації нового користувача

Сторінка авторизації (рис. 3.3) виконана в аналогічному стилі і містить два поля (електронна пошта та пароль) із кнопкою «Увійти». У разі введення невірних даних над формою з'являється червоний блок з текстом помилки. Під час обробки запиту кнопка змінює текст на «Завантаження...» та стає неактивною, що запобігає повторному натисканню. Після успішної авторизації JWT-токен та дані користувача зберігаються в сесійному сховищі браузера, а сам користувач потрапляє на головну сторінку.

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

Рис. 3.3. Сторінка авторизації користувача з хибною поштою

Головна сторінка системи (рис. 3.4) відкривається одразу після авторизації. У верхній частині відображається привітання з іменем поточного користувача. Нижче розташовані три основні картки зі статистикою: кількість проєктів, загальна кількість задач та кількість задач, призначених саме цьому користувачу. Перші дві картки є інтерактивними і ведуть на відповідні сторінки. Під ними знаходиться блок зі статусами задач, де кольоровими індикаторами (сірий, синій, зелений) відмічені категорії «До виконання», «В роботі» та «Виконано». Додатково відображається градієнтний індикатор прогресу із відсотком виконання та текстом скільки виконано задач. У правій частині сторінки виведено топ-3 лідерів рейтингу з іконками медалей а внизу знаходиться список п'яти останніх створених задач із кольоровими бейджами статусів.

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

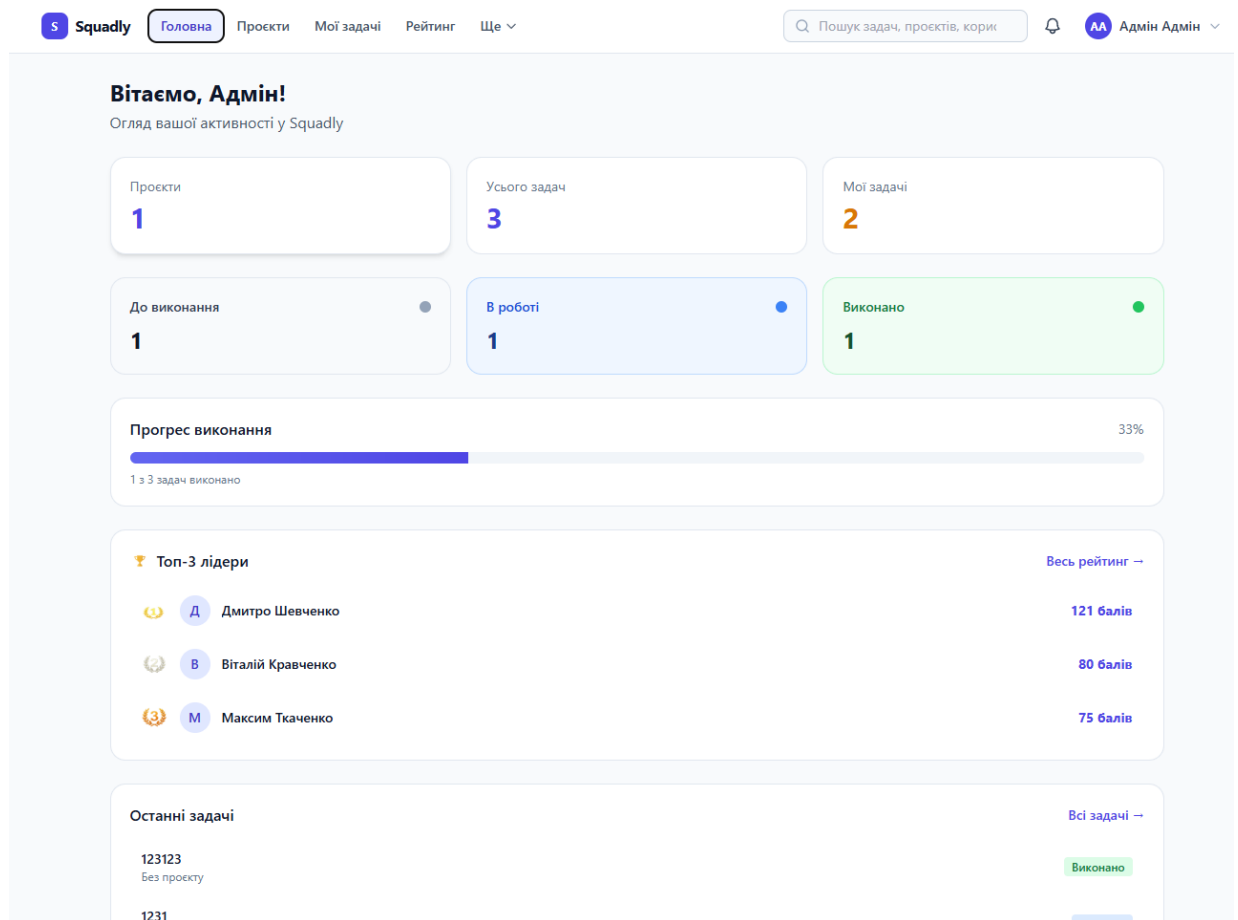


Рис. 3.4. Головна сторінка платформи

Сторінка проєктів (рис. 3.5) починається зі статистичних карток, що показують загальну кількість проєктів, кількість активних, завершених та архівних. Під ними розташовані кнопки-фільтри («Усі», «Активні», «Завершені», «Архів»), кожна з яких містить числовий лічильник. Для користувачів із роллю Організатора або Адміністратора у верхньому правому куті доступна кнопка «Створити проєкт». Самі проєкти відображаються у вигляді сітки карток. Кожна картка містить кольорову бічну смужку, позначку категорії з відповідною іконкою (Web, Mobile, Data, Game тощо), позначку пріоритету, назву, скорочений опис (до двох рядків), кількість учасників, дату дедлайну з індикацією червоний колір для прострочених, жовтий якщо залишилось менше трьох днів та список тегів.

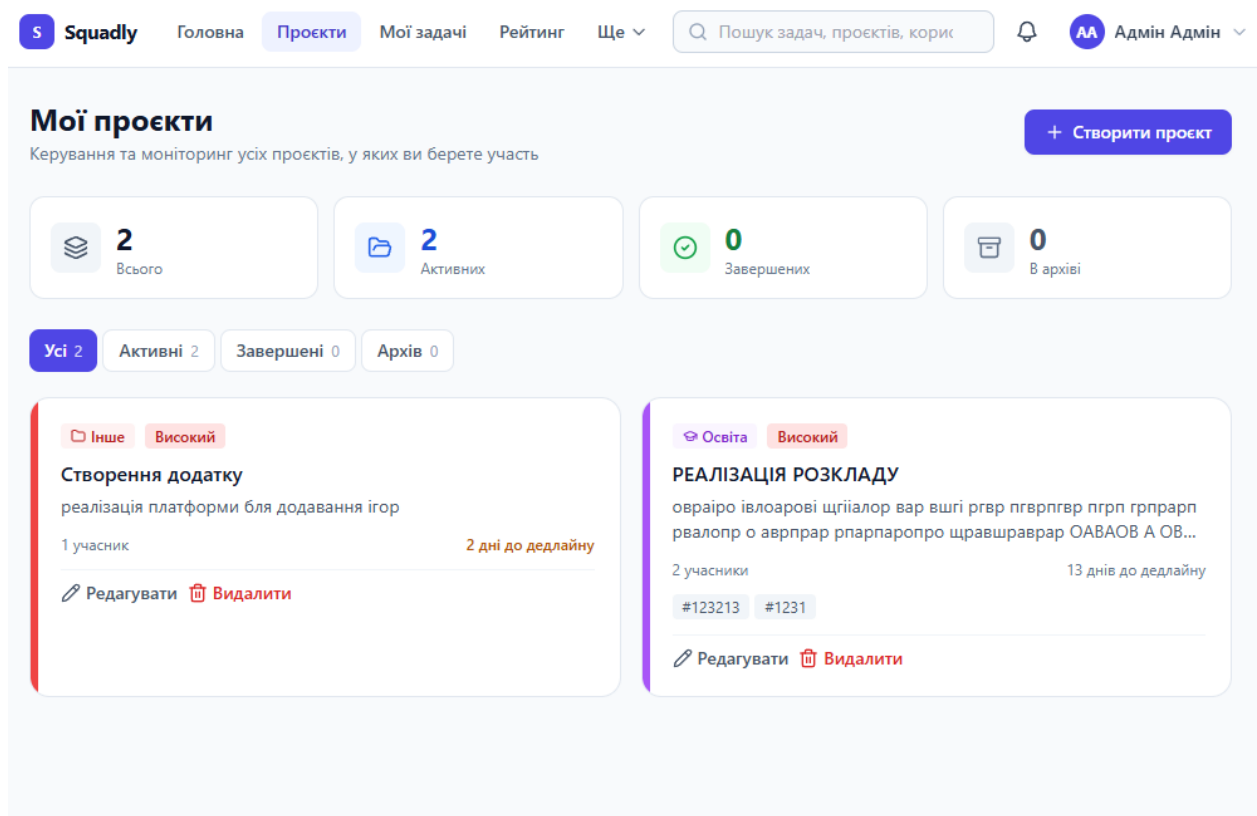


Рис. 3.5. Сторінка зі списком проектів

Створення нового проекту реалізовано через покроковий майстер, який відкривається у модальному вікні (рис. 3.6). Процес поділений на три етапи. На першому кроці «Основне» користувач вводить назву проекту (мінімум 3 символи), опис, обирає категорію із восьми запропонованих варіантів (у вигляді кнопок з іконками) та колір картки проекту із палітри кольорових кружечків.

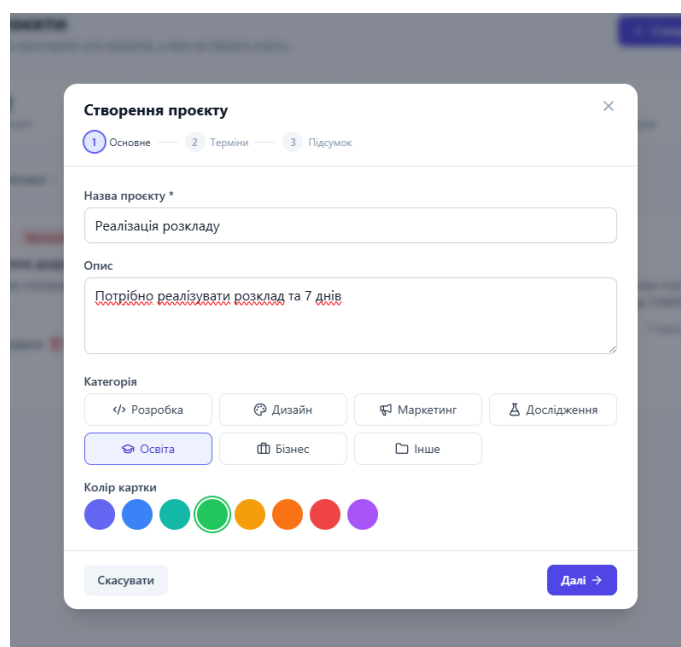


Рис. 3.6. Покроковий майстер створення проекту, перший крок

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Савицький Р. С.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

На другому кроці (рис. 3.7) «Терміни» вказуються дата старту, термінів виконання (з валідацією, що термін виконання не раніше за дату старту), пріоритет (три кнопки: Низький, Середній, Високий), мета проєкту (до 1000 символів із лічильником) та мітки (до 10 штук).

Створення проєкту

✓ Основне — 2 Терміни — 3 Підсумок

Дата старту: 18.06.2026

Дедлайн: 23.06.2026

Пріоритет: Низький, Середній, Високий

Мета проєкту: Реалізувати розклад за відведений час

38/1000

Теги (максимум 10): напр. frontend

Додати

#розклад #навчання

← Назад Далі →

Рис. 3.7. Покроковий майстер створення проєкту, другий крок

Третій крок (рис. 3.8) «Підсумок» показує всі введені дані у зручному вигляді для перевірки перед створенням. Між кроками розташовані кнопки «Назад» та «Далі», а прогрес відображається візуальними індикаторами зверху.

Створення проєкту

✓ Основне — ✓ Терміни — 3 Підсумок

Назва: Реалізація розкладу

Опис: Потрібно реалізувати розклад та 7 днів

Категорія: Освіта

Пріоритет: Високий

Колір: Зелений

Старт: 2026-06-18

Дедлайн: 2026-06-23

Мета: Реалізувати розклад за відведений час

Теги: #розклад #навчання

Перевір дані. Після створення ти автоматично станеш організатором.

← Назад Створити ✓

Рис. 3.8. Покроковий майстер створення проєкту, третій крок

При переході всередину проєкту відкривається сторінка його деталей (рис. 3.9). У верхній частині розміщена шапка з назвою, описом, категорією, пріоритетом, кількістю учасників, датами старту та дедлайну, а також метою проєкту. Для організатора доступні кнопки зміни статусу проєкту (Активний, Завершений, Архівний) та редагування. Під шапкою розташована горизонтальна панель вкладок із іконками: «Задачі», «Календар», «Учасники», «Чат», «Аналітика», «Звіт» та «Wiki». Кожна вкладка завантажує відповідний компонент без перезавантаження сторінки. За замовчуванням після переходу до робочого простору відразу активується вкладка із завданнями для швидкого початку роботи та візуальної оцінки поточного прогресу.

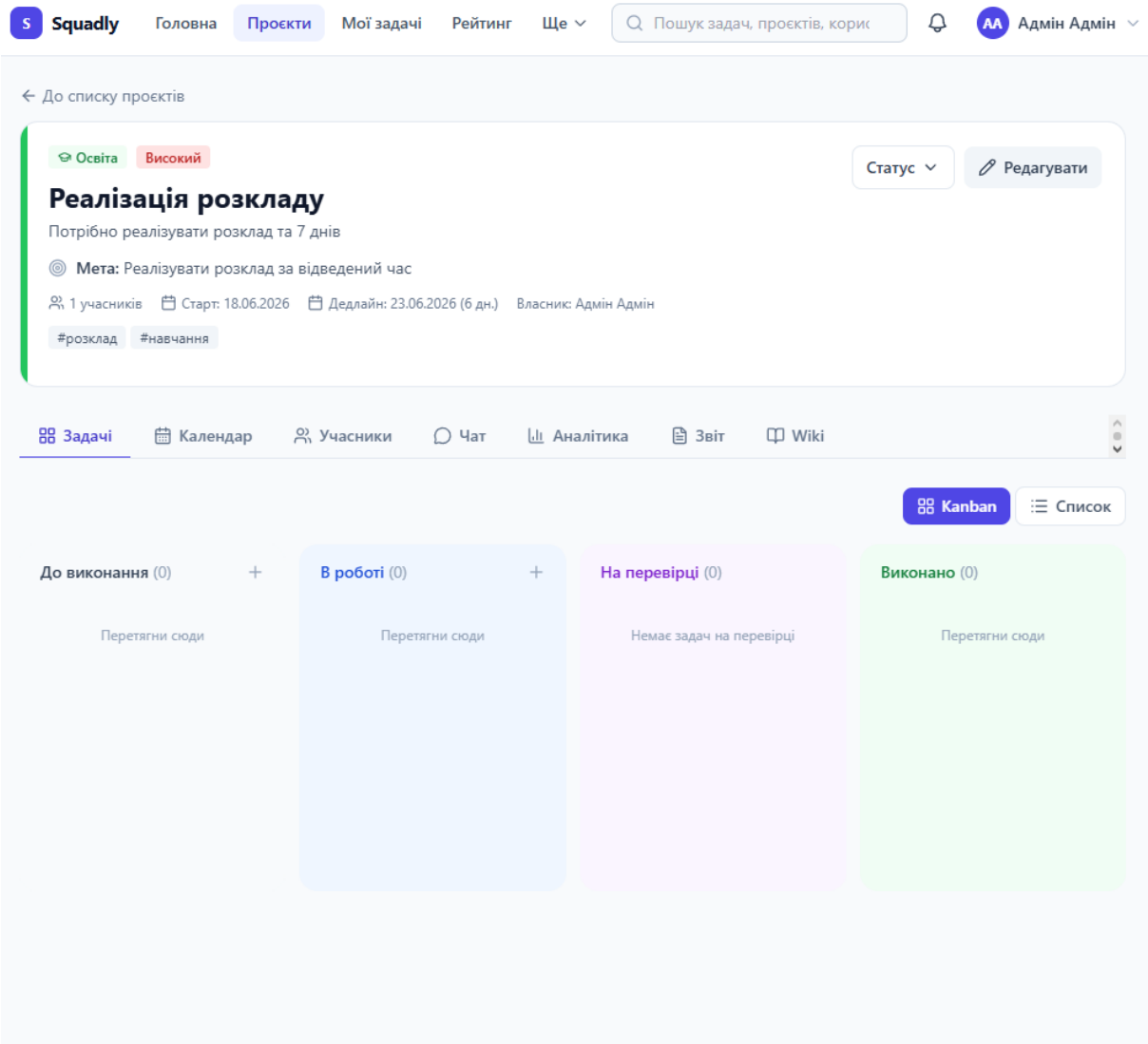


Рис. 3.9. Сторінка деталей проєкту

Основною вкладкою для щоденної роботи є Kanban-дошка (рис. 3.10). Вона розділена на чотири вертикальні колонки відповідно до статусів задач: «До виконання», «В роботі» (включає задачі на доопрацюванні), «На перевірці» та «Виконано». Кожна задача представлена карткою із назвою, описом, бейджем пріоритету, іменем виконавця та датою дедлайну. Задачі, повернені на доопрацювання, додатково виділяються помаранчевою рамкою та містять текст причини повернення. Перетягування карток між колонками реалізоване через механізм перетягування, при цьому система враховує обмеження бізнес-логіки: наприклад, не можна перетягнути задачу безпосередньо у колонку «Виконано» (це робиться тільки через перевірку ментором). На кожній картці відображаються контекстні кнопки залежно від ролі. Виконавець бачить кнопку «Здати», а ментор — кнопки «Перевірити», «Рішення» та «Відпустити». У верхній частині кожної колонки є кнопка «+» для швидкого створення нової задачі одразу в потрібному статусі. У правому верхньому куті вкладки розташований перемикач між режимами Kanban та Список.

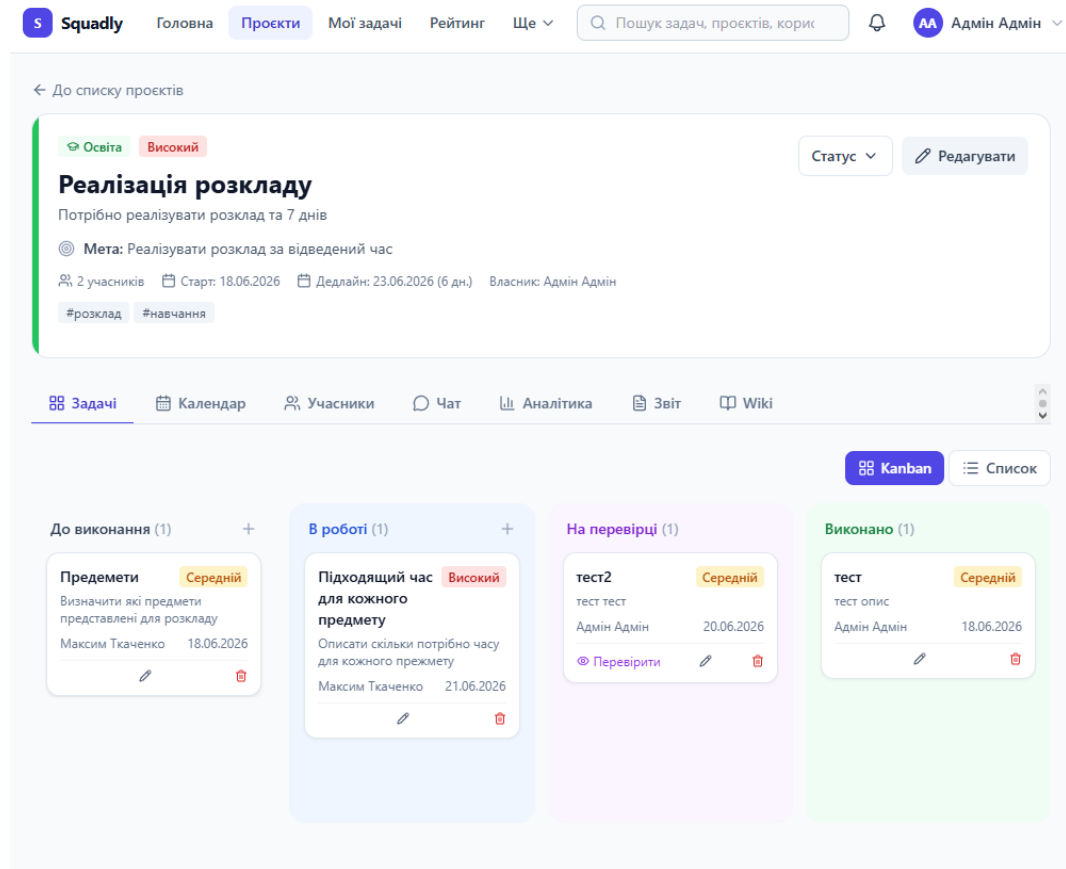


Рис. 3.10. Інтерфейс Kanban-дошки проекту

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савицький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

Альтернативним режимом перегляду задач є табличний список (рис. 3.11). У цьому режимі задачі згруповані за статусом і відображаються у вигляді рядків, які можна згортати та розгортати. Кожен рядок містить кольоровий індикатор пріоритету, назву, опис, ім'я виконавця, термін виконання та бейджі. Натискання на рядок відкриває модальне вікно з детальною інформацією.

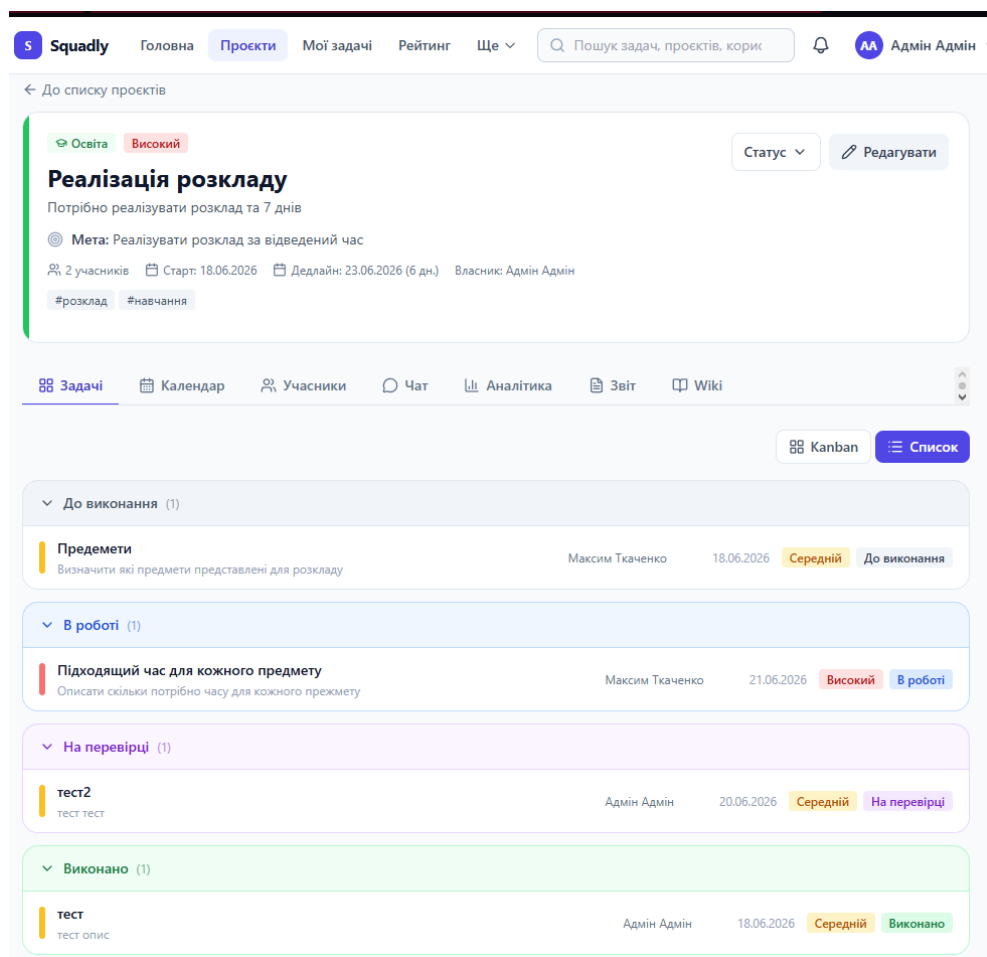


Рис. 3.11. Табличний вигляд задач проекту

Модальне вікно деталей задачі (рис. 3.12) організоване у вигляді чотирьох вкладок: «Інформація», «Здачі», «Коментарі» та «Файли». На вкладці «Інформація» відображаються всі параметри задачі, включаючи виконавця, проєкт, термін виконання та дату створення. Для задач із статусом «Потребує доопрацювання» додатково показується помаранчевий блок із причиною повернення.

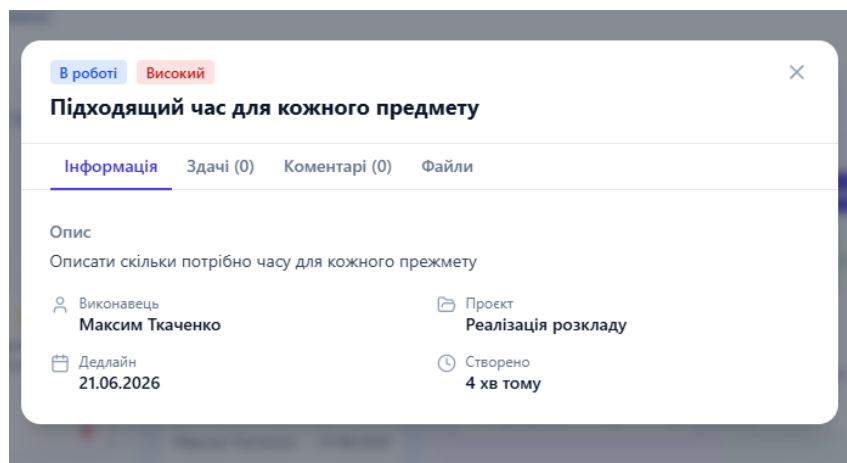


Рис. 3.12. Модальне вікно інформації задачі

Вкладка «Здачі» (рис. 3.13) містить хронологію всіх спроб задачі у вигляді вертикальної шкали часу, де для кожної спроби вказано номер, автора, опис виконаної роботи, кількість витрачених годин та прикріплені посилання.

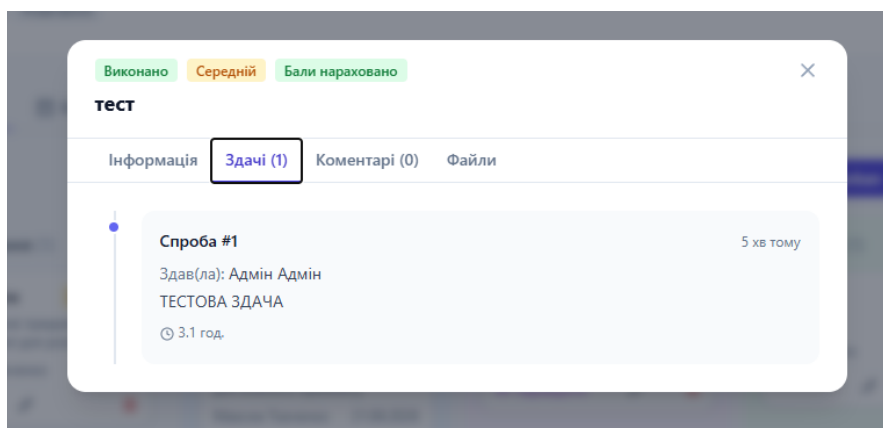


Рис. 3.13. Модальне вікно Здачі задачі

Вкладка «Коментарі» (рис. 3.14) дозволяє учасникам обмінюватися повідомленнями по задачі з можливістю видалення власних коментарів.

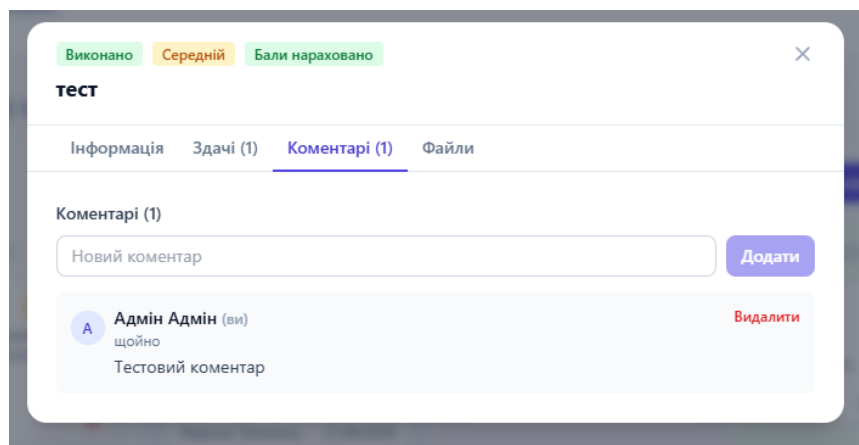


Рис. 3.14. Модальне вікно Коментарі задачі

Вкладка «Файли» (рис. 3.15) призначена для роботи з додатковими документами та вкладеннями до конкретного завдання. У цьому розділі користувачі додають потрібні робочі матеріали за допомогою окремої кнопки прикріплення файлу.

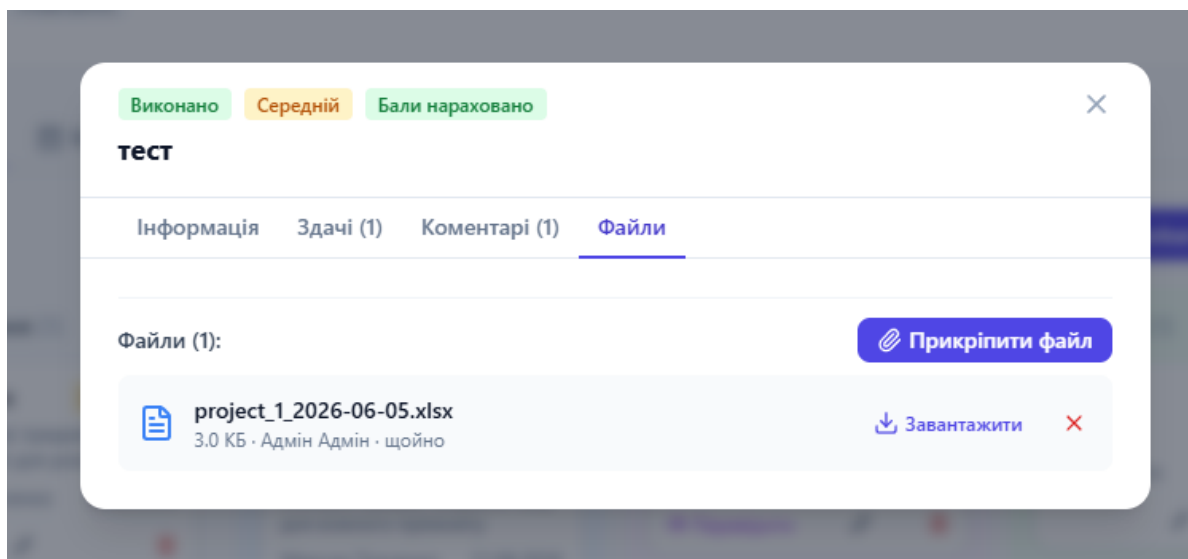


Рис. 3.15. Модальне вікно Файли задачі

Для забезпечення зручного управління робочим процесом розроблено спеціальне модальне вікно створення нової задачі (рис. 3.16). Виклик цього компонента відбувається безпосередньо з головного інтерфейсу поточного проєкту. Під час появи вікна основний контент сторінки перекривається напівпрозорим затемненим фоном. Таке візуальне рішення дозволяє ефективно сфокусувати увагу користувача виключно на процесі введення необхідних даних і запобігає випадковим клікам по інших елементах системи.

Сама форма заповнення складається з кількох логічних блоків. Першим є обов'язкове текстове поле для введення назви, яке в подальшому слугуватиме основним коротким ідентифікатором картки завдання на дошці. Нижче розташована розширена текстова область для опису, де організатор може максимально деталізовано викласти всі вимоги та умови виконання роботи. Для швидкого налаштування додаткових параметрів розробники інтегрували зручні випадаючі списки. З їх допомогою можна миттєво вибрати початковий статус виконання, вказати рівень важливості через поле пріоритету та одразу призначити відповідального виконавця серед підтверджених учасників команди.

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

Рис. 3.16. Модальне вікно створення нової задачі в межах проєкту

Для здачі задачі на перевірку виконавець використовує спеціальну форму (рис. 3.17), яка відкривається окремим модальним вікном. Форма містить текстове поле «Що зроблено» від 10 до 3000 символів із лічильником, поле для додавання посилань на результати до 10 штук, кожне відображається окремим тегом з можливістю видалення, числове поле витрачених годин з кроком 0. та обов'язковий прапорець самоперевірки «Я перевірів(ла) свою роботу перед задачею». Без активації цього прапорця кнопка «Здати на перевірку» залишається неактивною.

Здача задачі

ТЕСТ ЗДАЧІ ЗАДАЧІ НА ПЕРЕВІРКУ

Що зроблено *

ТЕСТ ЩО ЗРОБЛЕНО

16/3000

Файли прикріплюйте у вкладці «Файли» задачі — вони будуть видимі менторові під час перевірки. Тут вкажіть посилання на репозиторій, демо або документи (необов'язково, якщо є файли).

Посилання (GitHub, Drive, демо)

https://...

Додати

Витрачено годин

2,6

Опційно. Максимум 24 годин на одну задачу

☒ Я перевірів(ла) свою роботу перед задачею

Скасувати

Здати на перевірку

Рис. 3.17. Модальне вікно задачі завдання на перевірку

Центр менторства (рис. 3.18) є окремою сторінкою, де ментори та організатори бачать усі задачі, що очікують на перевірку. Сторінка поділена на дві вкладки. Перша вкладка «Очікують перевірки» містить список задач зі статусом InReview. Для кожної задачі відображається назва проєкту (клікабельний бейдж), пріоритет, назва задачі, ім'я виконавця з аватаром та час задачі. Якщо задачу ще ніхто не взяв, відображається кнопка «Взяти». Якщо інший ментор вже працює над перевіркою, замість кнопки з'являється індикатор з іконкою замка та іменем перевіряючого. Друга вкладка «Активність по проєктах» містить сітку проєктних карток із індикатор прогресу виконання.

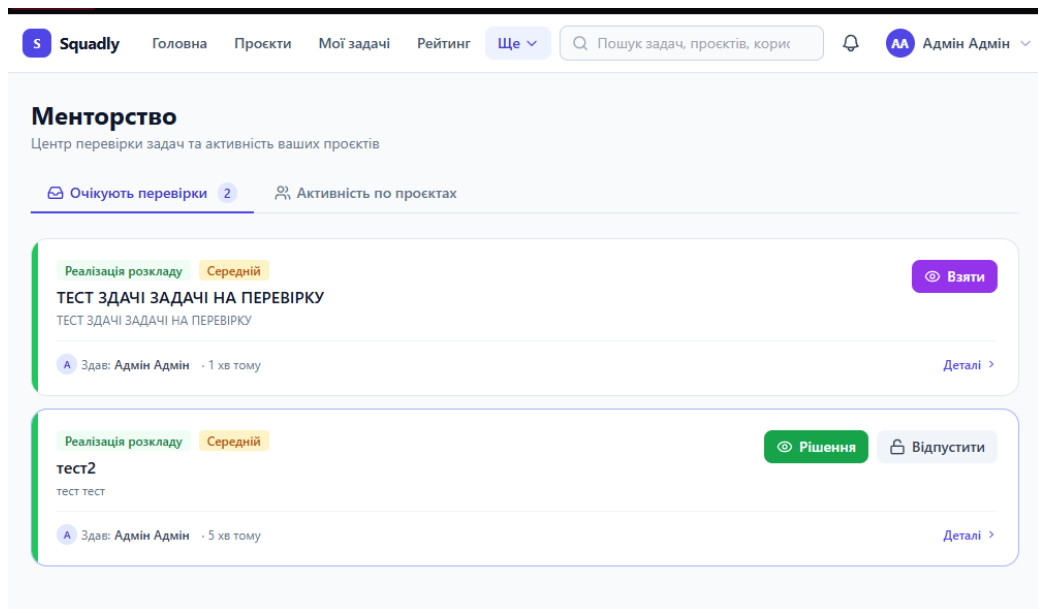


Рис. 3.18. Інтерфейс центру менторства

Для детальної роботи з підопічними ментор може перейти на сторінку менторства по конкретному проекту (рис. 3.19). Тут відображається список усіх учасників проекту з персональними метриками: кількість задач, виконаних задач, задач у роботі та набраних балів. Для кожного підопічного доступний індикатор прогресу виконання. При виборі конкретного учасника відкривається детальна сторінка з двома вкладками (рис. 3.20) «Задачі» список усіх задач цього виконавця та «Нотатки» приватні нотатки ментора, які видно тільки йому. Нотатки можна створювати, редагувати та видаляти.

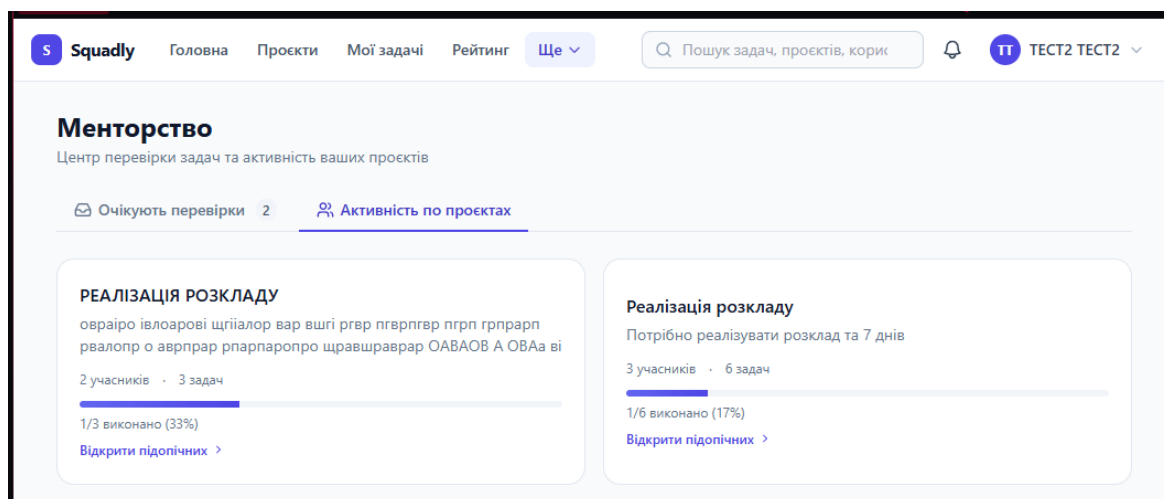


Рис. 3.19. Сторінка менторства по проекту з нотатками

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Савицький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

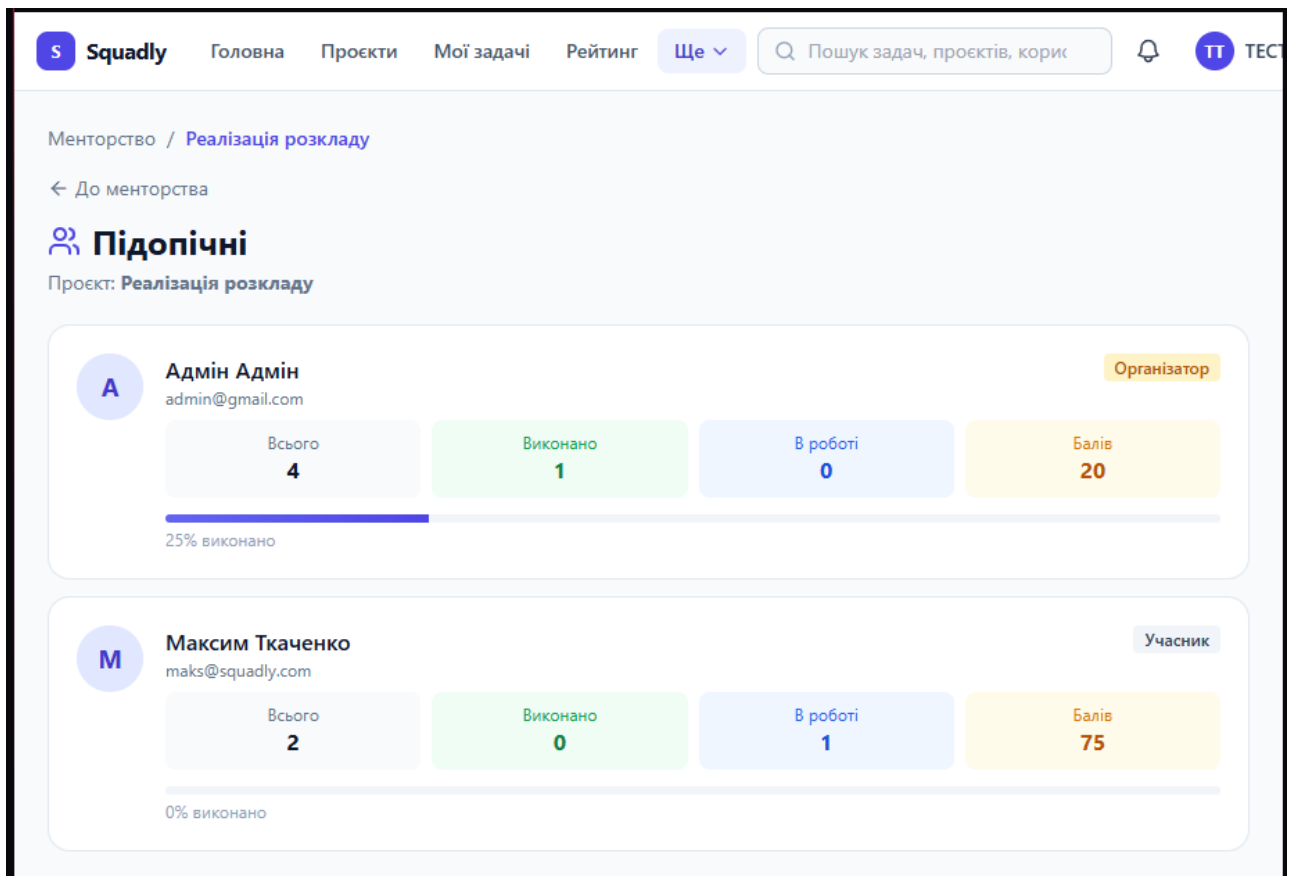


Рис. 3.20. Сторінка менторства по підопічним з нотатками

Сторінка «Мої задачі» (рис. 3.21) відображає тільки ті задачі, які призначені поточному користувачу як виконавцю. У верхній частині розташовані кнопки фільтрації за статусами виконання із відповідними числовими лічильниками. Задачі автоматично згруповані за проектами для зручності орієнтації. Елементи зі статусом доопрацювання виділяються помаранчевою рамкою і містять блок із причиною повернення від ментора. Для завдань на етапі виконання відображається кнопка швидкої здачі. Кожна картка додатково містить кольорову позначку рівня пріоритету та точну дату дедлайну. Успішно завершені роботи одразу отримують зелену мітку про нарахування балів. Такий підхід дозволяє виконавцю зручно відстежувати власну успішність та строки виконання в єдиному місці. Це звільняє користувача від необхідності постійно перемикатися між вкладками різних проєктів та допомагає краще концентруватися на першочергових цілях.

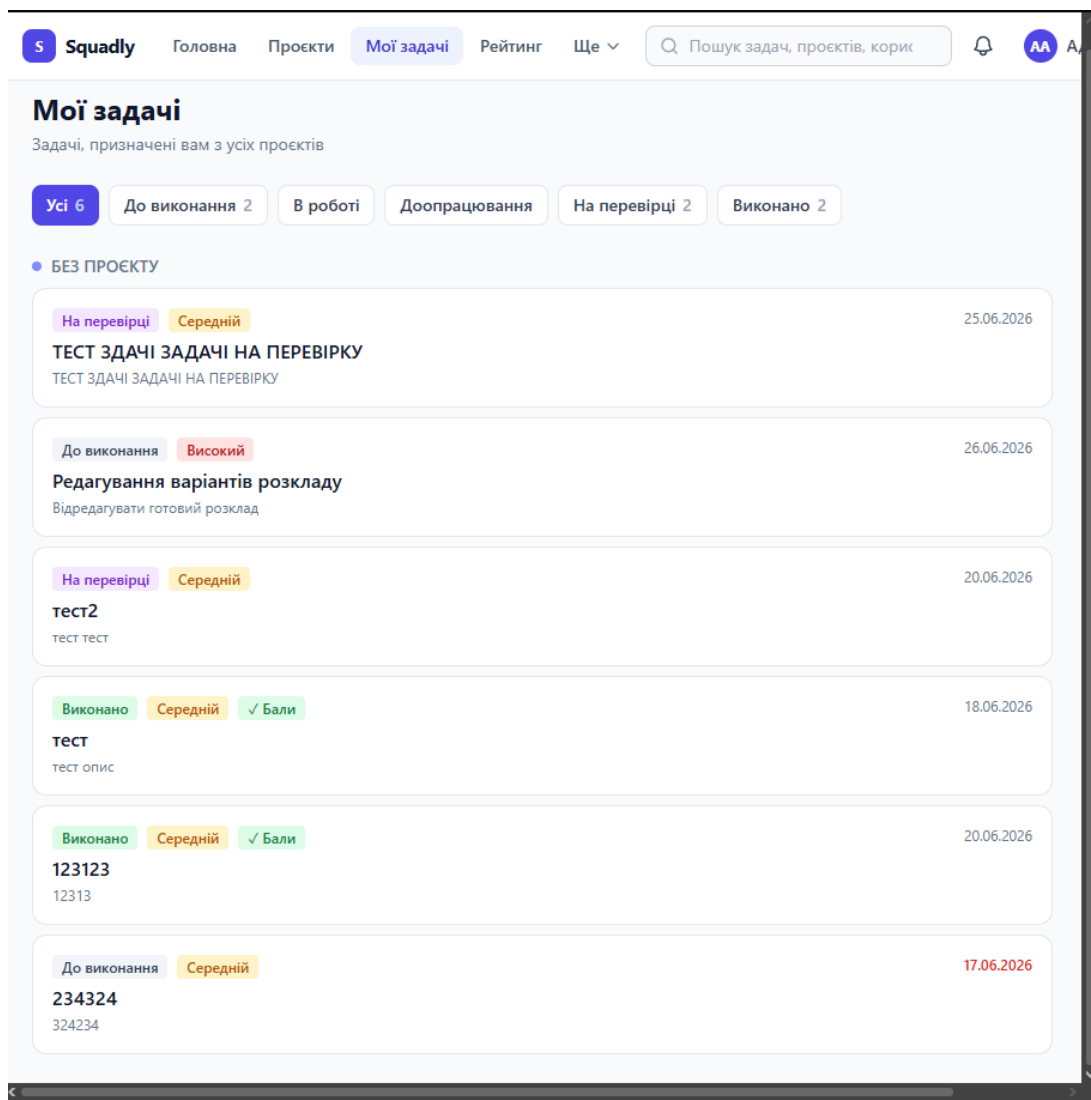


Рис.3.21. Сторінка персональних задач користувача

Сторінка рейтингу (рис. 3.22) візуалізує успіхи всіх учасників платформи. У верхній частині розташовані три збільшені картки лідерів із іконками золотої, срібної та бронзової медалей. Під ними відображається повний рейтинговий список у вигляді таблиці з колонками: позиція, аватар та ім'я, кількість виконаних задач і загальна сума балів. Рядок поточного користувача виділяється фоном та має почначку «ви». Топ-3 рядки мають градієнтне забарвлення. Внизу сторінки розташований інформаційний блок «Як заробити бали?» із таблицею нарахування (20 балів за високий пріоритет, 10 за середній, 5 за низький, 1 за коментар).

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

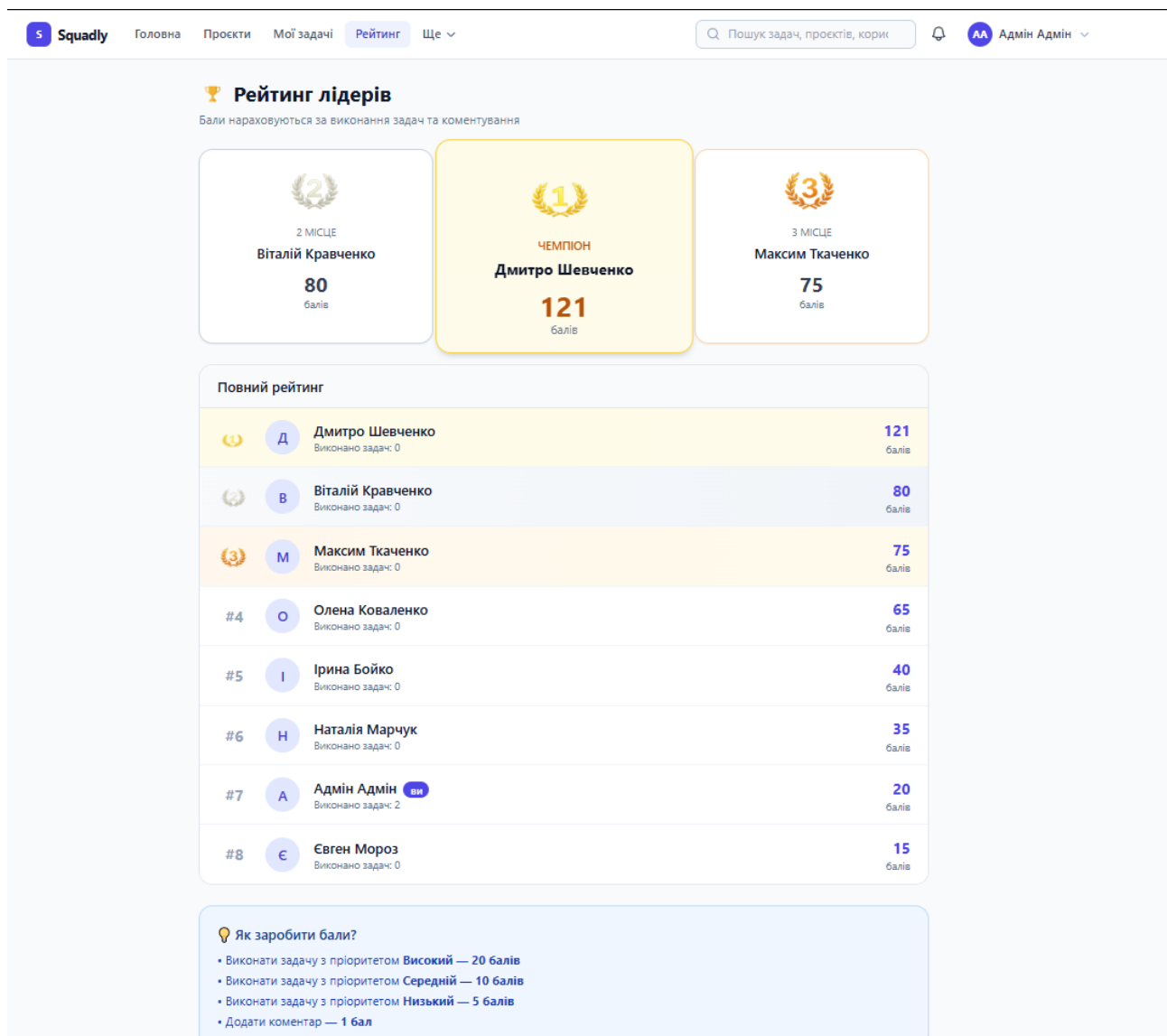


Рис. 3.22. Сторінка рейтингу лідерів

Сторінка аналітики (рис. 3.23) надає комплексну статистику по всіх проєктах користувача. У верхній частині розміщено п'ять KPI-карток (Проектів, Задач, Користувачів, Коментарів, Відсоток виконання). Нижче розташовані чотири інтерактивні графіки, побудовані за допомогою бібліотеки Recharts: кругова діаграма розподілу задач за статусом, кругова діаграма задач за пріоритетом, лінійний графік активності за останні 14 днів (лінії «Створено задач» та «Коментарів») і стовпчикова діаграма задач по проєктах (топ-10) із групуванням за статусами. Додатково відображається список топ-виконавців із горизонтальними індикаторами прогресу.

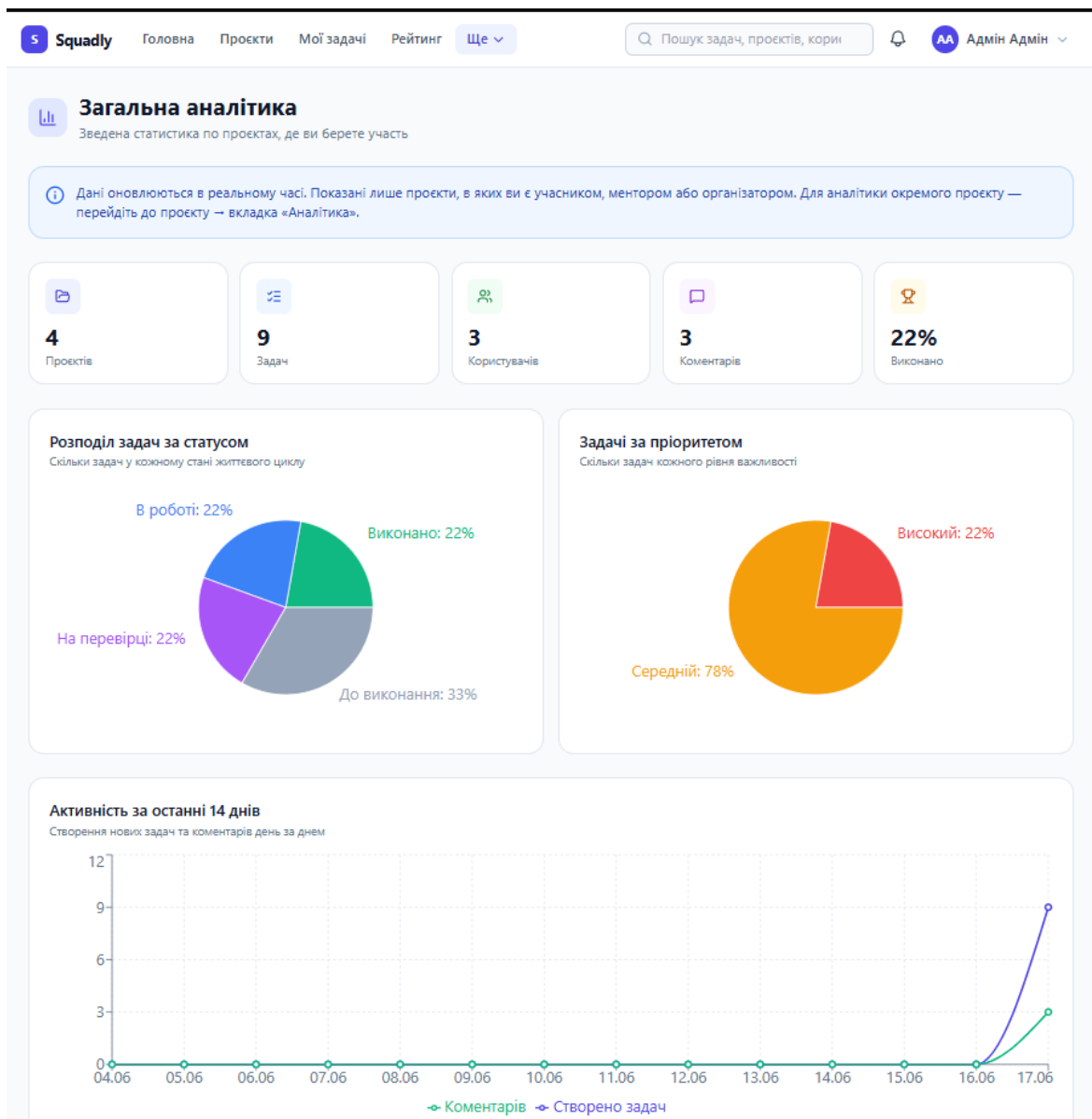


Рис. 3.23. Сторінка аналітики платформи

Календар задач (рис. 3.24) представляє альтернативний спосіб перегляду задач за їхніми дедлайнами. Навігація здійснюється кнопками переключення місяців та кнопкою «Сьогодні» для повернення до поточної дати. Поточний день виділений кольоровим фоном. У кожній комірці календаря відображається до трьох задач із кольоровою індикацією статусу, якщо задач більше, з'являється напис «+» з відповідною кількістю. Прострочені задачі позначаються червоним індикатором. У верхній частині сторінки відображаються загальні лічильники прострочених та виконаних задач. При натисканні на задачу відкривається модальне вікно з її деталями.

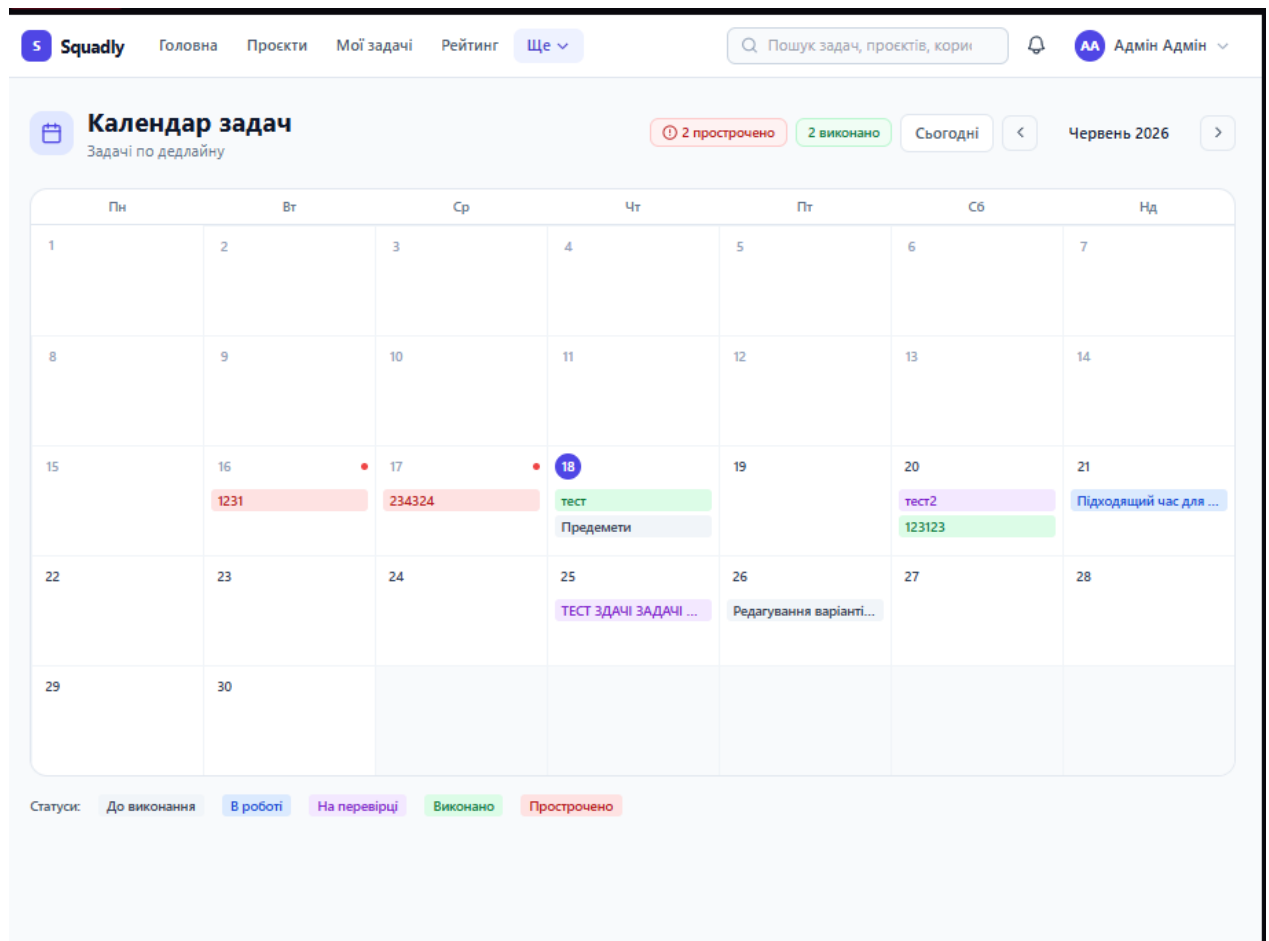


Рис. 3.24. Календарний вигляд задач

Сторінка сповіщень (рис. 3.25) містить хронологічний список усіх повідомлень системи. Доступні фільтри «Всі» та «Непрочитані» з числовими лічильниками, а також кнопка «Прочитати всі» для масового позначення. Кожне сповіщення складається з іконки, заголовка, тексту повідомлення та відносного часу. Непрочитані сповіщення позначені синім індикатором. Для кожного сповіщення доступні кнопки переходу до відповідної задачі, позначення як прочитаного та видалення. Лічильник непрочитаних повідомлень у навігаційній панелі через SignalR.

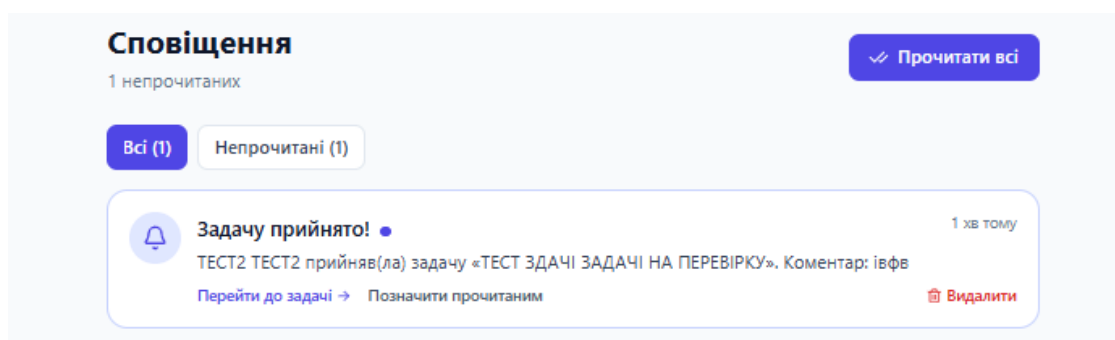


Рис. 3.25. Сторінка сповіщень користувача

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

Сторінка профілю (рис. 3.26) відображає персональні дані користувача та його статистику. У верхній частині розташований блок із аватаром (ініціал у кольоровому колі), повним іменем та електронною адресою. Кнопка «Редагувати профіль» перемикає блок у режим редагування, де можна змінити ім'я та прізвище. Нижче знаходиться секція «Моя статистика» з чотирма КРІ-картками: кількість балів, місце в рейтингу, кількість виконаних задач та кількість коментарів. Додатково відображаються горизонтальні індикатор прогресу з розподілом виконаних задач за пріоритетами (Високий, Середній, Низький) із відсотками.

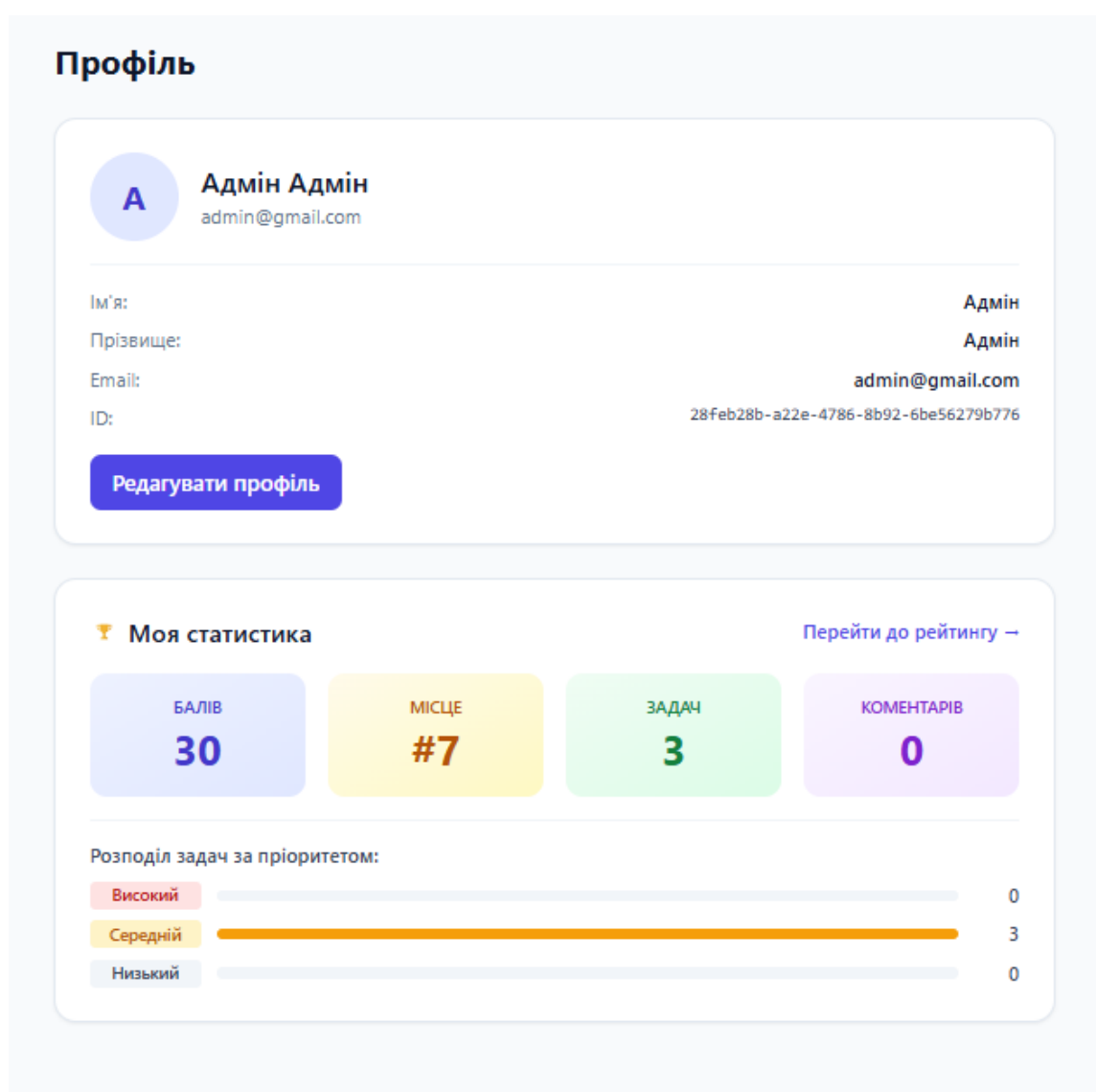


Рис. 3.26. Сторінка особистого профілю користувача

Сторінка звітів (рис. 3.27) дозволяє генерувати актуальні аналітичні документи у двох форматах на основі даних із бази у реальному часі. Верхня

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Савіцький Р. С.				72
Змн.	Арк.	№ докум.	Підпис	Дата		

частина інтерфейсу відображає ключові показники особистої активності користувача та містить кнопку для завантаження персонального документа у форматі PDF. Нижче розташовано список активних проєктів із кнопками експорту даних у зручні для друку або подальшого аналізу файли разом із корисними підказками про їхні переваги.

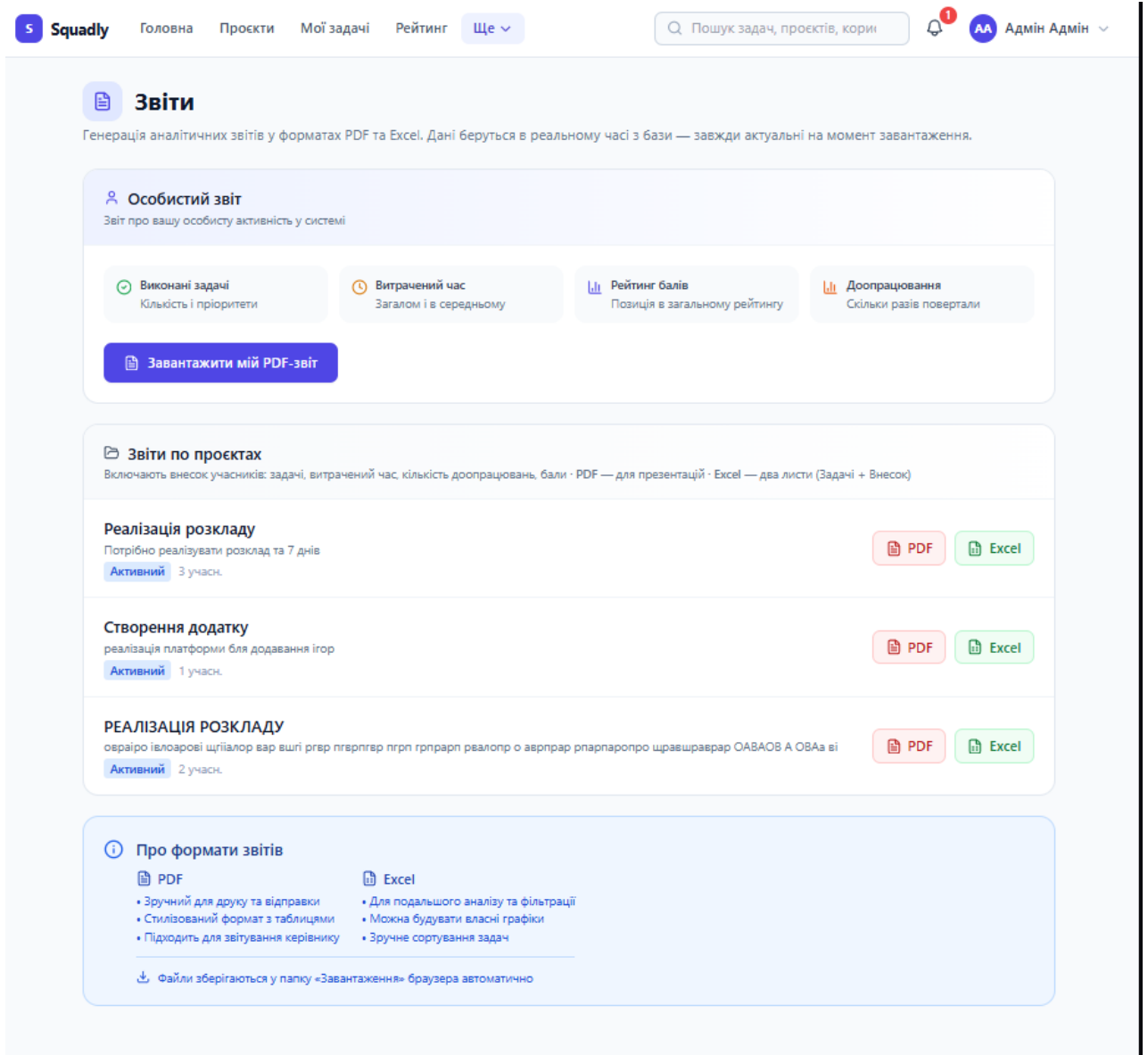


Рис. 3.27. Сторінка генерації звітів

У верхній частині розташована секція «Особистий звіт» з персональним PDF-звітом (рис. 3.28) [23]. Цей документ містить зведену статистику користувача та детальну інформацію про його активність. У ньому відображаються кількість виконаних завдань, загальний витрачений час,

кількість доопрацювань та повний перелік призначених задач із зазначенням статусу.

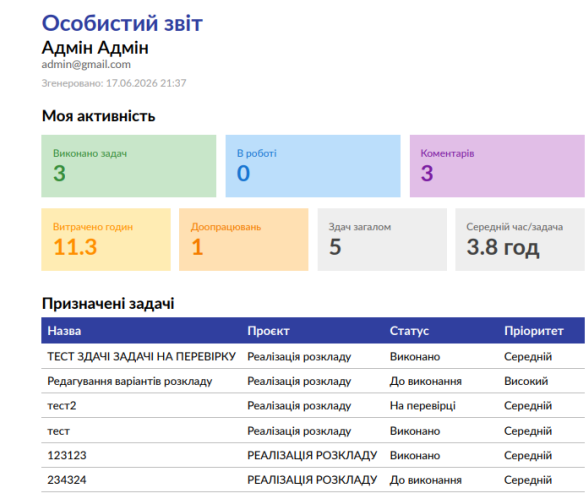


Рис. 3.28. Сторінка генерації особистого звіту

Нижче знаходиться секція «Звіти по проєктах» (рис. 3.29), де для кожного проєкту доступні дві кнопки: червона для завантаження PDF-звіту та зелена для Excel-звіту. Ці файли надають загальну статистику виконання робіт та наочно показує відсоток прогресу всієї команди. Документ також містить зведені показники кожного окремого учасника та детальний перелік усіх проєктних завдань для зручного аналізу результатів.

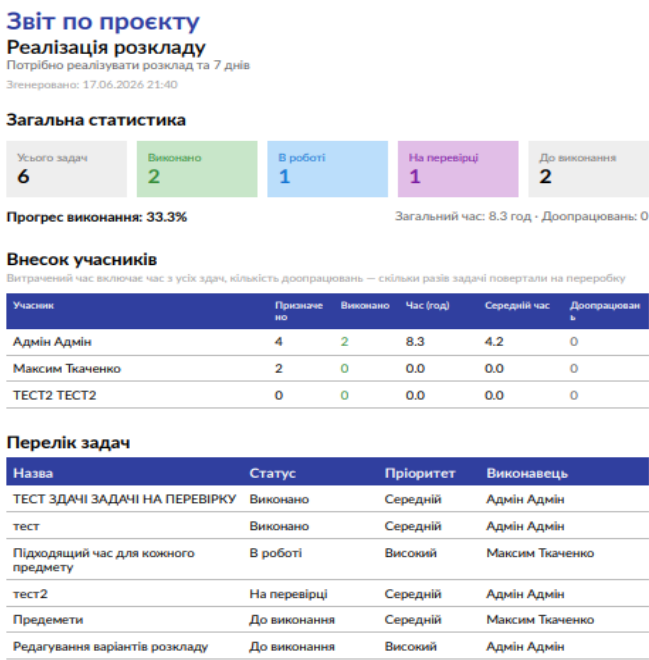


Рис. 3.29. Сторінка генерації звіту по проєктах

Адміністративна панель (рис. 3.30) доступна виключно користувачам із роллю адмін і містить чотири вкладки. Вкладка «Користувачі» відображає таблицю всіх зареєстрованих облікових записів із можливістю зміни ролей через випадаючий список, редагування на місці персональних даних та видалення із захистом від видалення власного акаунту.

Користувач	Email	Роль	Проектів	Дії
T Test User	test8@squadly.com	Користувач	0	
T Test66@gmail.com Test6@gmail.com	test6@gmail.com	Користувач	0	
Є Євген Мороз	evgen@squadly.com	Користувач	0	
I Ірина Бойко	ira@squadly.com	Користувач	0	
A Адмін Адмін (ви)	admin@gmail.com	Адміністратор	3	
A Анна Мельник	anna@squadly.com	Організатор	5	
A Артем Руденко	artem@squadly.com	Організатор	3	
В Віталій Кравченко	vitaliy@squadly.com	Користувач	0	
Д Дмитро Шевченко	dmitro@squadly.com	Користувач	0	
М Максим Ткаченко	maks@squadly.com	Користувач	0	
Н Наталія Марчук	nata@squadly.com	Користувач	0	
О Олександр Пастухов	pastukhovc@gmail.com	Користувач	0	
О Олена Коваленко	olena@squadly.com	Користувач	0	
T ТЕСТ2 ТЕСТ2	test2@gmail.com	Організатор	1	

Рис. 3.30. Адміністративна панель системи Користувачі

Вкладка «Усі проекти» (рис. 3.31) містить таблицю для управління та редагування всіх створених робочих просторів. Для кожного елемента списку відображається його повна назва, ім'я власника, кількість залучених учасників та загальна статистика виконання завдань. Завдяки окремим кнопкам дій адміністратор має змогу швидко змінити налаштування потрібного проекту або повністю видалити його з бази даних.

Адмін-панель

Керування користувачами, проектами та системою

Користувачі

Усі проекти

Звернення

Статистика

Назва	Власник	Учасників	Задачі	Дії
впвап вап	TEST2 TEST2	1	0/1	
Реалізація розкладу Потрібно реалізувати розклад та 7 днів	Адмін Адмін	3	2/6	
Створення додатку реалізація платформи для додавання ігор	Адмін Адмін	1	0/0	
Маркетингова кампанія "Spring" Промо-матеріали та лендінг	Анна Мельник	2	0/0	
Дизайн-система "Nebula" Створення компонентів для Figma	Анна Мельник	3	1/3	
Гра "Space Explorer" Інді-гра на руші Unity	Артем Руденко	2	1/2	
AI Чат-бот для університету Розробка Telegram-бота на основі GPT	Артем Руденко	3	1/3	
Система моніторингу серверів DevOps утиліта	Артем Руденко	3	0/2	
Мобільний додаток HealthTracker Застосунок для відстеження активності	Анна Мельник	0	0/0	
Розробка веб-платформи Squadly Головний проєкт для курсової роботи	Анна Мельник	3	1/4	
РЕАЛІЗАЦІЯ РОЗКЛАДУ оверіро іеловорів щтіалор еар вигі ргпр пгерлгпр пгріл грпярлп реалопр о ав...	Адмін Адмін	2	1/3	
Крипто-гаманець Мобільний застосунок для Ethereum	Анна Мельник	2	0/0	

Рис. 3.31. Адміністративна панель системи Проекти

Вкладка «Звернення» (рис. 3.32) інтегрує двопанельний чат із користувачами (список звернень зліва, область повідомлень справа), який працює в реальному часі через окремий SignalR-хаб.

Адмін-панель

Керування користувачами, проектами та системою

Користувачі

Усі проекти

Звернення

Статистика

Звернення користувачів

Відповідайте на запитання користувачів. Тут же можна змінити роль користувача.

Пошук по email або імені...

Т

Test66@gmail.com Test6@g...

test6@gmail.com

Вик: dfs

Користувач

10 год тому

Рис. 3.32. Адміністративна панель системи Звернення

Вкладка «Статистика» (рис. 3.33) показує загальні числові показники системи. Цей розділ містить п'ять окремих карток із даними про загальну кількість зареєстрованих користувачів, адміністраторів, організаторів, а також кількість створених проєктів та завдань.

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

Адмін-панель

Керування користувачами, проєктами та системою

Користувачі

Усі проєкти

Звернення

Статистика

УСЬОГО
КОРИСТУВАЧІВ

6

АДМІНІСТРАТОРІВ

1

ОРГАНІЗАТОРІВ

2

ПРОЄКТІВ

3

ЗАДАЧ

4

Рис. 3.33. Адміністративна панель системи Статистика

Важливою складовою управління проєктом є контроль за його учасниками та розподіл рівнів доступу. На рис. 3.34 зображено вкладку «Учасники», яка доступна в межах робочого простору проєкту. Цей розділ інтерфейсу дозволяє організаторам формувати команду та налаштовувати ролі кожного з її членів.

На сторінці представлено список поточних учасників у вигляді карток, кожна з яких містить аватар (згенерований на основі ініціалів), повне ім'я користувача, його електронну адресу та поточну роль у проєкті. Система підтримує три основні ролі на рівні проєкту.

Оскільки поточний користувач має права організатора, йому доступні елементи керування складом команди. За допомогою випадаючих списків навпроти кожного користувача організатор може змінювати ролі учасників. Також передбачено кнопку видалення для виключення користувача з проєкту. Для залучення нових членів команди у верхньому правому куті інтерфейсу розташована кнопка «Додати», яка ініціює процес запрошення нових учасників до спільної роботи.

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

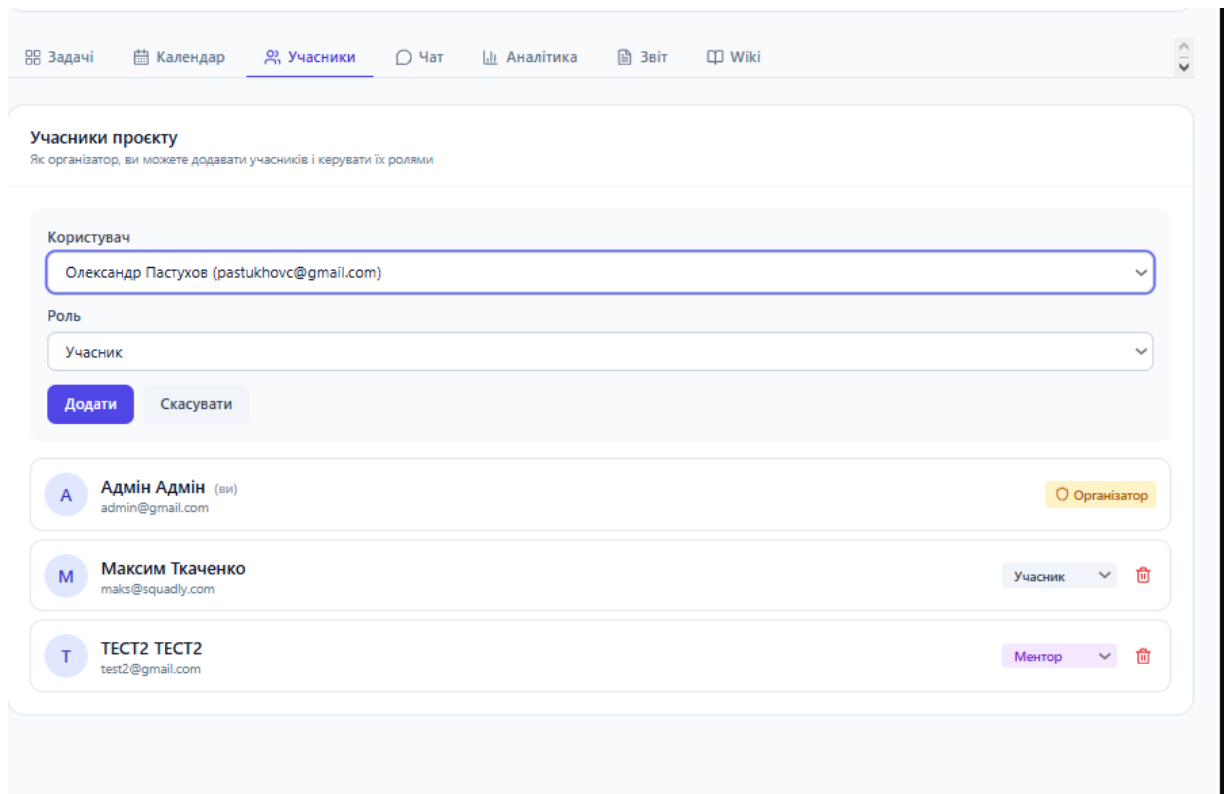


Рис.3.34. Вкладка управління учасниками та їх ролями в межах проєкту

Для забезпечення оперативної комунікації між членами команди у веборієнтованій платформі реалізовано внутрішній проєктний чат (рис. 3.35). Доступ до нього здійснюється через відповідну вкладку в меню навігації проєкту. Цей модуль дозволяє учасникам обговорювати робочі питання без необхідності перемикатися на сторонні месенджери, що централізує всю проєктну інформацію в єдиному середовищі.

Інтерфейс чату візуально розділений на три основні зони: заголовок, область повідомлень та панель вводу. У заголовку відображається назва проєкту (або конкретної кімнати), загальна кількість повідомлень у поточній історії та індикатор стану підключення («Online»), який свідчить про успішне встановлення вебсокет-з'єднання через технологію SignalR.

Область повідомлень відображає хронологію спілкування. Власні повідомлення користувача вирівнюються по правому краю вікна та виділяються фірмовим кольором разом з аватаром та позначкою часу відправки, тоді як повідомлення інших учасників відображаються зліва. У нижній частині розташовано поле для введення тексту з підказкою та кнопка «Надіслати».

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

Завдяки використанню SignalR, нові повідомлення з'являються в усіх підключених до проєкту учасників миттєво та без необхідності ручного оновлення сторінки.

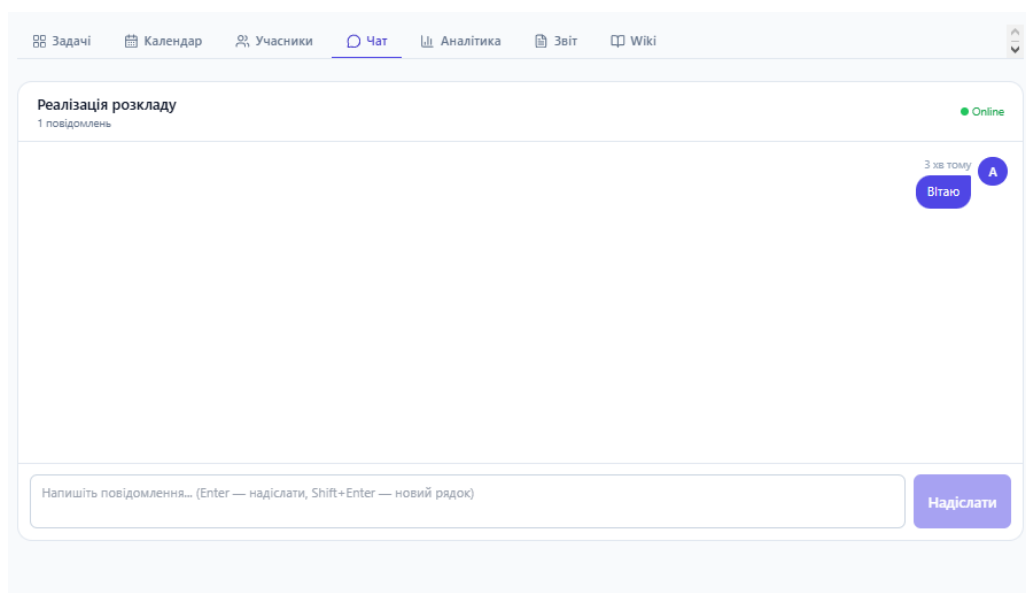


Рис. 3.35. Вкладка проєктного чату для комунікації команди

Для збереження проєктної документації, інструкцій та бази знань у системі передбачено модуль «Wiki» (рис. 3.36). Ця вкладка дозволяє команді спільно створювати та редагувати структуровані текстові документи, що особливо важливо для довготривалих проєктів зі складною бізнес-логікою або специфічними правилами роботи.

Інтерфейс модуля розділений на дві функціональні частини. Ліва бічна панель («Розділи») виконує роль навігаційного меню бази знань. Вона містить перелік існуючих сторінок та інструмент для швидкого створення нового розділу.

Основна (права) частина екрана відведена безпосередньо під редактор контенту. У режимі редагування користувачу доступне велике текстове поле для введення інформації. Система підтримує базове форматування тексту за допомогою спеціальних символів. Для зручності користувачів під текстовим редактором розташовано інформаційний блок із підказками щодо правил форматування тексту. Завершення роботи над документом підтверджується

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

кнопкою «Зберегти», після чого змінений контент стає доступним усім учасникам проєкту.

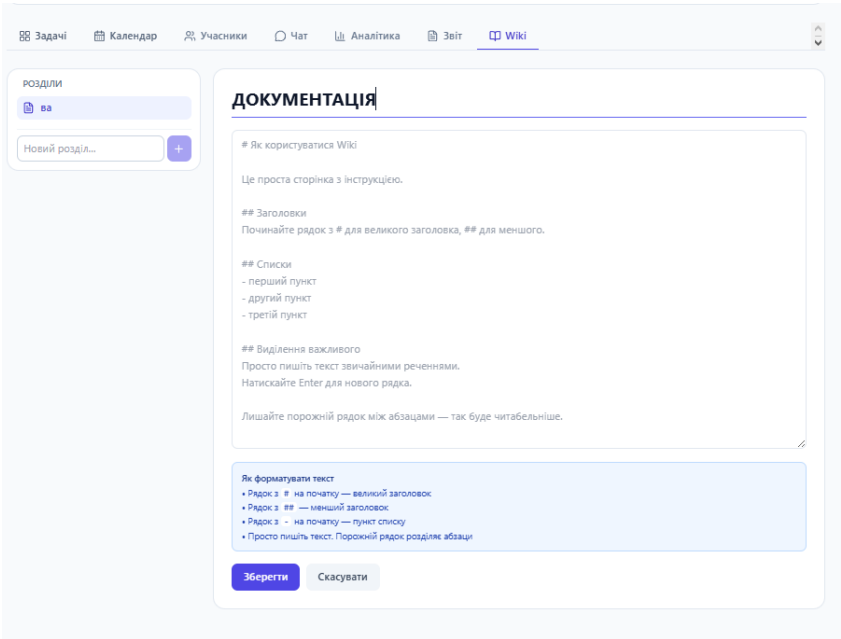


Рис. 3.36. Вкладка Wiki для ведення проєктної документації та бази знань команди

Реалізовано вкладку «Аналітика» (рис. 3.37), яка виконує роль інформаційного дашборда і забезпечує агрегацію ключових показників ефективності команди в режимі реального часу.

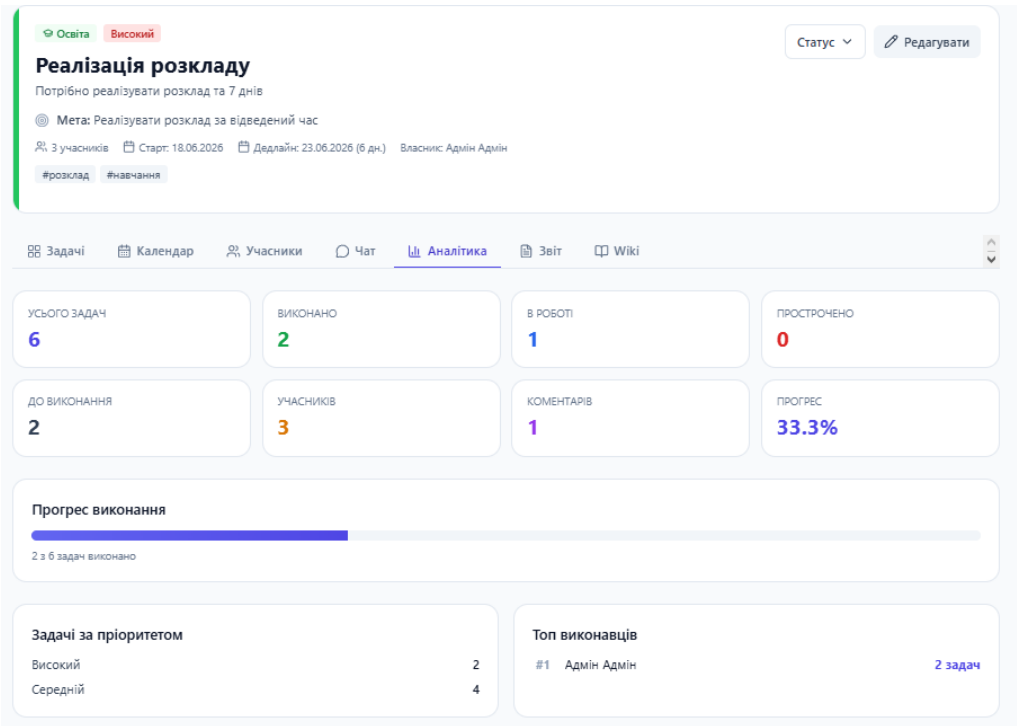


Рис. 3.37. Вкладка аналітики з ключовими показниками ефективності проєкту

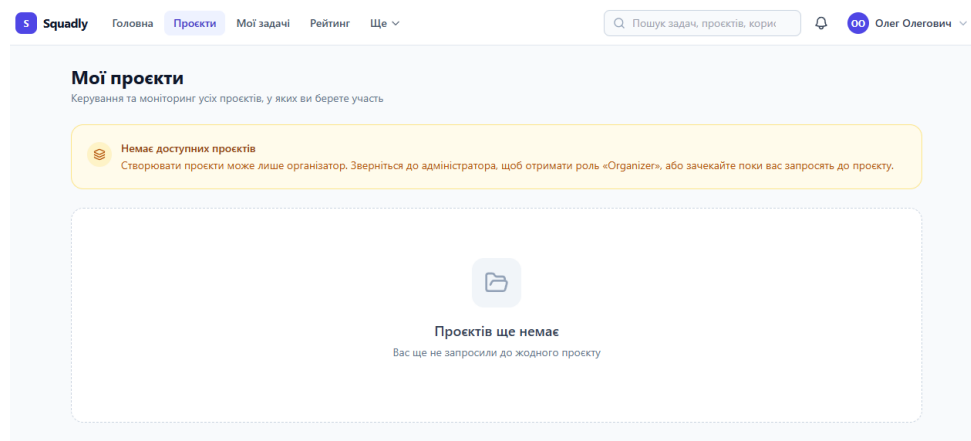
Верхня частина сторінки, яка залишається статичною при перемиканні вкладок, містить базову інформацію: назву, опис, категорію (наприклад, «Освіта»), пріоритет, теги, а також дати старту і дедлайну проєкту з розрахунком днів, що залишилися. Тут також розташовані кнопки зміни глобального статусу проєкту та переходу в режим редагування параметрів.

Основна область вкладки «Аналітика» поділена на кілька логічних блоків. Перший блок містить набір карток із ключовими кількісними показниками: загальна кількість задач, скільки з них виконано, знаходяться в роботі, прострочено чи очікують виконання. Також виводиться кількість учасників, залишених коментарів та загальний відсоток прогресу проєкту.

Під картками показників розташовано графічний індикатор «Прогрес виконання», що візуалізує співвідношення закритих задач до їх загальної кількості. Нижня частина дашборда містить деталізовану статистику, розділену на віджети «Задачі за пріоритетом» та «Топ виконавців».

Важливим аспектом якісного UX є правильна обробка ситуацій, коли дані для відображення відсутні. На рис. 3.38 наведено приклад сторінки «Мої проєкти» для нового користувача, який ще не бере участі в жодному проєкті. Замість порожнього екрана система виводить інформативне повідомлення «Проектів ще немає» та спеціальний блок-підказку. Цей блок пояснює правила доступу: оскільки користувач має базову роль, він не може самостійно створювати проєкти. Система чітко інструктує його щодо подальших дій.

У нижньому правому куті екрана на всіх сторінках платформи закріплена кнопка, яка відкриває модуль зв'язку з підтримкою.



		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		81

Рис. 3.38. Відображення порожнього стану сторінки проєктів з інформаційними підказками

Інтерфейс вікна підтримки виконаний у вигляді класичного месенджера. Верхня панель містить назву модуля та кнопку закриття вікна. Центральна частина відведена під історію листування на (рис. 3.39) зображено початковий стан із привітанням, а внизу розташовано поле для вводу тексту. Як і в проєктному чаті, тут підтримується відправка повідомлень клавішею Enter. Повідомлення з цього модуля автоматично потрапляють у вкладку «Звернення» адміністративної панелі (описаної раніше), забезпечуючи централізовану обробку запитів системою.

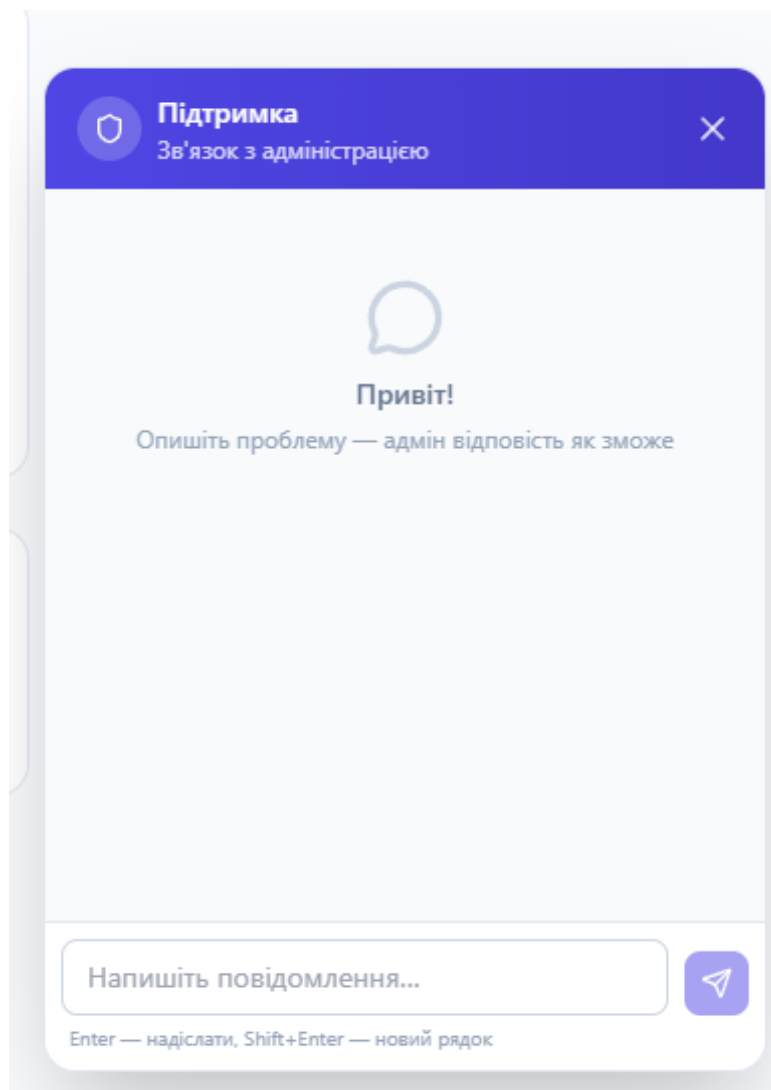


Рис. 3.39. Інтерфейс чату для прямого зв'язку користувача з адміністрацією платформи

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Савіцький Р. С.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

3.3 Тестування платформи

Тестування платформи Squadly виконано за багаторівневим підходом, який поєднує автоматизоване юніт-тестування доменної моделі, ручне функціональне тестування основних бізнес-сценаріїв через інтерактивну документацію API та клієнтський інтерфейс, інтеграційне тестування взаємодії компонентів у контейнерному середовищі, а також тестування адаптивності інтерфейсу під різні роздільні здатності екранів. Такий підхід дозволяє виявити дефекти на різних рівнях від помилок у обчисленнях окремих полів сутностей до проблем інтеграції бекенду і фронтенду при реальній взаємодії користувача із системою.

Автоматизоване юніт-тестування доменної моделі реалізоване у виділеному проєкті Squadly.Tests, який є частиною загального рішення. Тести розроблено на основі фреймворку NUnit версії 4.2.2 у поєднанні з пакетами Microsoft.NET.Test.Sdk та NUnit3TestAdapter, що забезпечують інтеграцію тестового запуску у командний рядок dotnet та у Visual Studio Test Explorer[20]. Додатково підключено пакет NUnit.Analyzers для статичного аналізу тестового коду й попередження поширених помилок написання тестів, а також coverlet.collector для збору метрик покриття коду тестами.

Тестова група UserTests перевіряє правильність формування обчислюваної властивості FullName, яка об'єднує ім'я і прізвище користувача з коректним обрізанням пробілів у випадку, коли одне з полів порожнє. Тестується також ініціалізація нового екземпляра користувача: відсутність null-значень у рядкових полях, генерація унікального ідентифікатора GUID при кожному створенні, ненульова ініціалізація колекцій членств і призначених задач. Перевірка ненульовості колекцій є важливою, оскільки звернення до null-колекції в LINQ-запитах призводить до NullReferenceException у виробничому коді.

Тестова група ProjectTests перевіряє доменні сутності проєкту значення Active як статусу за замовчуванням при створенні нового проєкту, можливість переходу проєкту у стан Completed або Archived, відсутність прапорця IsDeleted у щойно створеного запису. Окремо параметризованим тестом з використанням атрибутів TestCase перевірено відповідність числових значень enum ProjectStatus

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Савицький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

очікуваним: Active відповідає нулю, Completed відповідає одиниці, Archived відповідає двійці. Така перевірка критична, оскільки числові значення enum зберігаються у базі даних, і неконтрольована зміна порядку елементів призведе до некоректної інтерпретації наявних даних.

Запуск юніт-тестів у локальному середовищі здійснюється командою `dotnet test` з кореневої директорії `backend`. Результат виконання містить кількість пройдених і провалених тестів, час виконання, а також детальний звіт про кожну помилку, якщо такі трапилися. На рис. 3.40 наведено зразок виводу консолі після успішного запуску всіх тестів

```
PS C:\Users\VINGA\Desktop\Squadly\backend> dotnet test
Determining projects to restore...
C:\Users\VINGA\Desktop\Squadly\backend\Squadly.Infrastructure\Squadly.Infrastructure.csproj : warning NU1902: Package 'System.IdentityModel.Tokens.Jwt' 7.0.3 has a known moderate severity vulnerability, https://github.com/advisories/GHSA-59j7-ghrg-fj52 [C:\Users\VINGA\Desktop\Squadly\backend\Squadly.sln]
All projects are up-to-date for restore.
Squadly.Domain -> C:\Users\VINGA\Desktop\Squadly\backend\Squadly.Domain\bin\Debug\net8.0\Squadly.Domain.dll
Squadly.Tests -> C:\Users\VINGA\Desktop\Squadly\backend\Squadly.Tests\bin\Debug\net8.0\Squadly.Tests.dll
Test run for C:\Users\VINGA\Desktop\Squadly\backend\Squadly.Tests\bin\Debug\net8.0\Squadly.Tests.dll (.NETCoreApp,Version=v8.0)
VSTest version 17.11.1 (x86)

Starting test execution, please wait...
A total of 1 test files matched the specified pattern.

Passed! - Failed: 0, Passed: 12, Skipped: 0, Total: 12, Duration: 67 ms - Squadly.Tests.dll (net8.0)
PS C:\Users\VINGA\Desktop\Squadly\backend>
```

Рис. 3.40. Результат виконання юніт-тестів у консолі

Основний обсяг функціонального тестування платформи виконано через інтерактивну документацію Swagger UI, яка автоматично генерується на основі контролерів та DTO-класів за допомогою бібліотеки `Swashbuckle.AspNetCore`. Swagger UI є галузевим стандартом для опису REST API і дозволяє переглядати повний перелік кінцевих точок, ознайомлюватися зі схемами вхідних і вихідних даних, виконувати тестові запити безпосередньо з браузера без необхідності використання спеціалізованих інструментів. Доступ до інтерфейсу здійснюється за адресою `/swagger` після запуску бекенду в режимі розробки.

Усі кінцеві точки платформи згруповані за контролерами, що відповідають доменним областям: `Auth` для автентифікації та реєстрації, `Users` для роботи з обліковими записами, `Projects` для керування проєктами, `Tasks` для роботи із задачами, `TaskSubmissions` для здачі задач на перевірку, `Comments` для

коментарів, Notifications для сповіщень, Wiki для внутрішньої документації проєкту, Ratings для рейтингів та балів, Reports для генерації звітів у форматах PDF та Excel, AdminChat для чату з адміністрацією, Search для глобального пошуку, Admin для адміністративних операцій. На рис. 3.41 наведено вигляд Swagger UI з розгорнутим переліком усіх ендпоінтів.

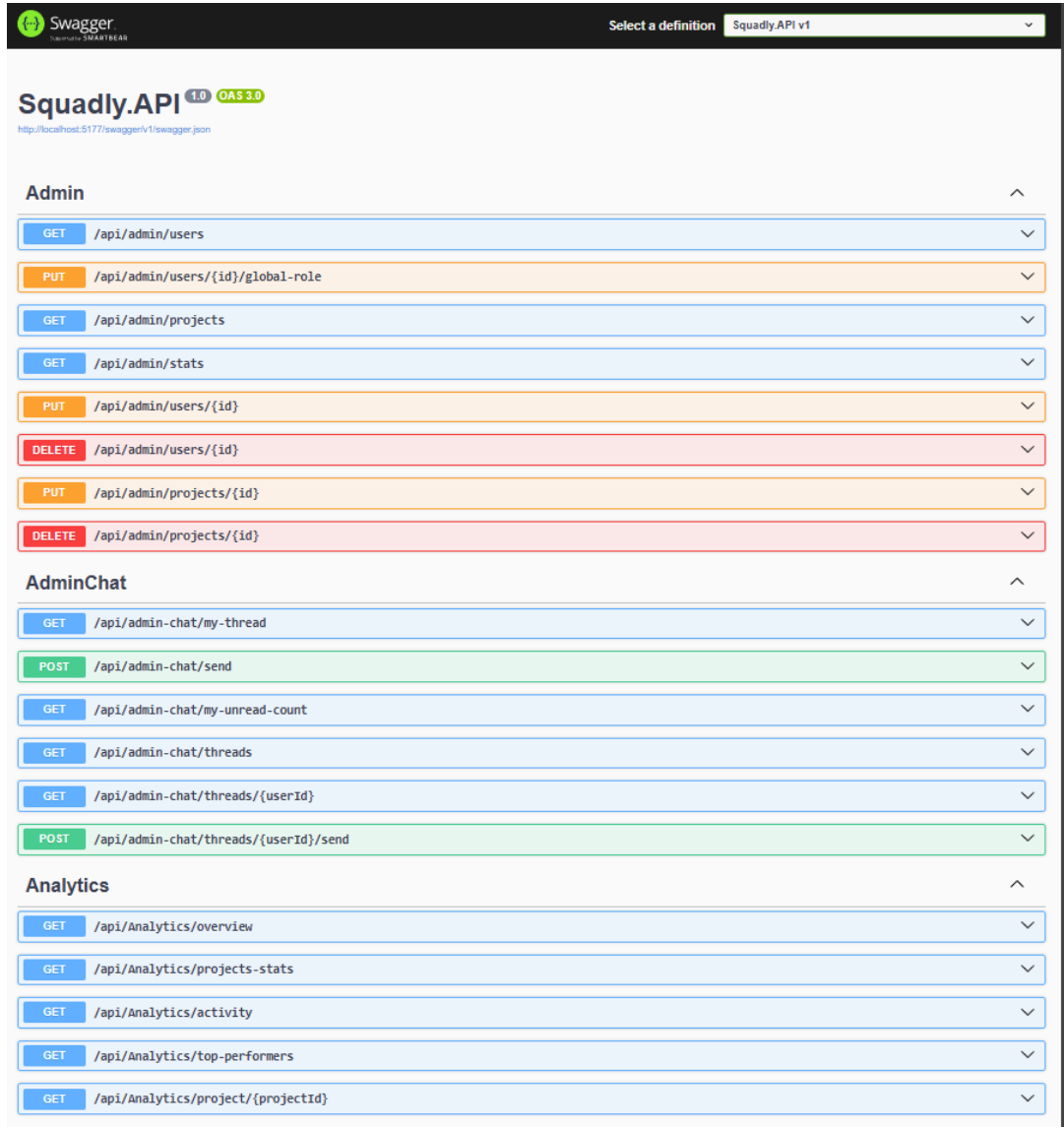


Рис. 3.41. Інтерфейс Swagger UI з переліком кінцевих точок API

На рис. 3.41 наведено приклад тестового запиту до кінцевої точки реєстрації нового користувача з некоректно заповненими даними. У полі електронної адреси введено значення «не-email», яке не відповідає формату e-mail, а в полі пароля було уведено рядок з трьох символів, що не задовольняє вимогу мінімальної довжини. Через інтерфейс Swagger UI виконано HTTP-запит

методом POST за адресою /api/auth/register з відповідним JSON-тілом, що містить значення полів firstName, lastName, email та password.

Сервер коректно опрацював такий запит, виявив помилку на етапі валідації вхідних даних і повернув відповідь зі статусом 400 Bad Request. У блоці Response body відображено структурований JSON-об'єкт з полем message, що містить локалізоване українською мовою повідомлення «Невірний формат email». Заголовки відповіді показують тип контенту application/json з кодуванням utf-8, що забезпечує коректну передачу кирилических символів повідомлення.

Така поведінка системи демонструє декілька важливих аспектів реалізації. По-перше, валідація вхідних даних виконується на серверній стороні незалежно від клієнтської валідації, що гарантує цілісність даних навіть у разі обходу клієнтського інтерфейсу. По-друге, повідомлення про помилки повертаються українською мовою через окремий обробник з кастомізованим текстом, що покращує користувацький досвід та спрощує локалізацію інтерфейсу. По-третє, формат відповіді є єдиним для всіх типів помилок, що дозволяє клієнтській частині мати уніфікований обробник без додаткової логіки для різних сценаріїв.

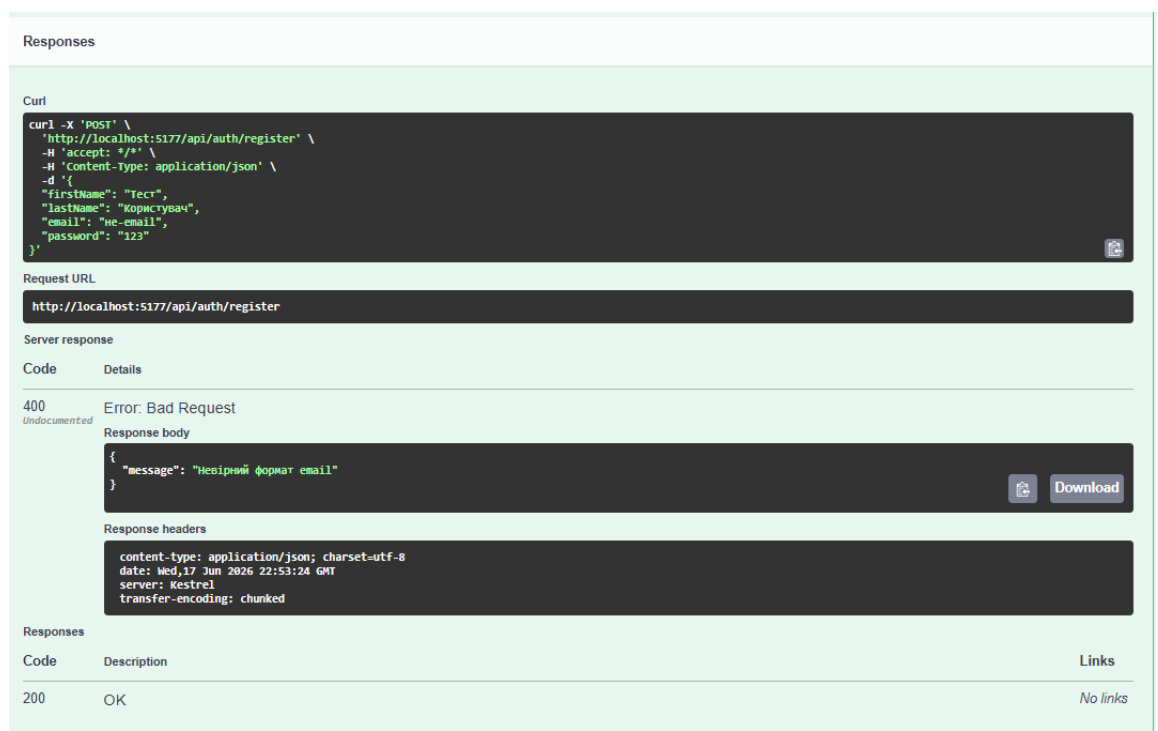


Рис. 3.42. Обробка некоректного запиту реєстрації через Swagger UI

Висновки до третього розділу

Третій розділ присвячено практичним аспектам реалізації веборієнтованої платформи Squadly. Тут детально висвітлено архітектуру розгортання програмного продукту, побудову користувацького інтерфейсу та підсумки проведеного тестування.

Процес розгортання системи базується на технологіях контейнеризації Docker. Серверна і клієнтська частини разом із базою даних PostgreSQL працюють як незалежні контейнери під управлінням інструменту docker-compose. Таке рішення гарантує ідентичну роботу платформи як на локальному комп'ютері розробника, так і на робочому сервері, а також значно полегшує процедуру базового встановлення.

Значну увагу приділено структурі візуального інтерфейсу та ключовим сценаріям взаємодії користувачів із системою. Охоплено процеси реєстрації та входу, створення проєктів, інструменти керування завданнями, а також процедури здачі та рецензування робіт. Крім того, описано функціонал для ведення внутрішньої бази знань Wiki, перегляду аналітичних даних, генерації звітів і спілкування у командних чи адміністративних чатах.

Для підтвердження надійності платформи було виконано її комплексне багаторівневе тестування. Доменна модель перевірена за допомогою автоматизованих юніт-тестів на базі фреймворку NUnit. Робота кінцевих точок HTTP та реакція системи на некоректні вхідні дані протестовані через інтерактивну панель Swagger UI.

Загальні результати перевірок довели правильність виконання всіх закладених алгоритмів роботи. Розроблена система повністю відповідає сформованим раніше функціональним вимогам і є готовою до повноцінного впровадження та використання в реальних умовах.

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		87

ВИСНОВКИ

У межах випускної кваліфікаційної роботи було розроблено вебплатформу Squadly, призначену для організації групових проєктів. Створена система об'єднує в єдиному цифровому середовищі засоби керування проєктами, постановки й контролю виконання задач, перевірки результатів менторами, ведення внутрішньої вікі-документації та командної комунікації. Платформа охоплює повний цикл колективної роботи від ініціалізації проєкту та розподілу завдань до їх перевірки й генерації підсумкових звітів.

Виконано аналіз предметної області та досліджено провідні програмні аналоги, зокрема Trello, Jira та Notion. На основі проведеного аналізу виявлено їхні переваги й обмеження та сформовано перелік функціональних і нефункціональних вимог до власної платформи. Обґрунтовано вибір технологічного стеку для реалізації серверної та клієнтської частин системи.

Архітектуру платформи спроектовано за принципами чистої архітектури з поділом на чотири рівні: доменний, прикладний, інфраструктурний та презентаційний. Доменна модель налічує чотирнадцять сутностей, що описують предметну область, зокрема проєкти, задачі, користувачів, повідомлення, сповіщення та вікі-сторінки. Визначено сценарії використання для чотирьох ролей користувачів і реалізовано дворівневу модель ролей глобальну та проєктну.

Здійснено практичну реалізацію платформи з повним покриттям сформульованих вимог. Серверну частину побудовано на базі ASP.NET Core 8 із використанням Entity Framework Core для взаємодії з реляційною базою даних PostgreSQL 16. Реалізовано механізм автентифікації та авторизації на основі JWT-токенів із розмежуванням прав за ролями, канбан-дошку з валідацією переходів між статусами, систему призначення та перевірки задач ментором, модуль сповіщень і групових чатів через бібліотеку SignalR, генерацію звітів у форматах PDF та Excel, а також адміністративну панель. Клієнтську частину створено з використанням бібліотеки React 18 та мови TypeScript, збірку виконано засобами Vite. Систему контейнеризовано за допомогою Docker

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				88
Змн.	Арк.	№ докум.	Підпис	Дата		

Compose й налаштовано конвеєр безперервної інтеграції та доставки у GitLab CI/CD.

Проведене багаторівневе тестування підтвердило коректність роботи основних функціональних можливостей платформи. Юніт-тести, реалізовані на базі фреймворку NUnit, покрили доменну логіку сутностей User та Project. Функціональне тестування програмного інтерфейсу через Swagger UI охопило основні сценарії взаємодії з контролерами системи. Додатково перевірено роботу модулів реального часу та адаптивність інтерфейсу на різних роздільних здатностях екранів.

Розроблена платформа Squadly повністю реалізує функціональні можливості, визначені на етапі аналізу вимог, та готова до практичного впровадження в умовах реальної експлуатації. Створений програмний продукт може використовуватися як інструмент організації спільної роботи в студентських колективах, навчальних групах та малих робочих командах, що підтверджує досягнення мети, поставленої у вступі до випускної кваліфікаційної роботи.

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		89

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ASP.NET Core documentation. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/aspnet/core/> (дата звернення: 15.06.2026).
2. ASP.NET Core SignalR overview. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction> (дата звернення: 15.06.2026).
3. Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps. 2nd ed. Sebastopol : O'Reilly Media, 2020. 310 p.
4. Cherny B. Programming TypeScript: Making Your JavaScript Applications Scale. Sebastopol : O'Reilly Media, 2019. 324 p.
5. Compose specification. Docker Documentation. URL: <https://docs.docker.com/compose/compose-file/> (дата звернення: 15.06.2026).
6. Docker Documentation. Docker Inc. URL: <https://docs.docker.com/> (дата звернення: 15.06.2026).
7. Entity Framework Core documentation. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/ef/core/> (дата звернення: 15.06.2026).
8. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. Boston : Addison-Wesley, 2003. 560 p.
9. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2002. 533 p.
10. Freeman A. Pro ASP.NET Core 6: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages. 9th ed. New York : Apress, 2022. 1170 p.
11. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston : Addison-Wesley, 1994. 395 p.
12. GitLab CI/CD documentation. GitLab Inc. URL: <https://docs.gitlab.com/ee/ci/> (дата звернення: 15.06.2026).

		Пастухов О. К.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Савицький Р. С.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

- 13.ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. Geneva : ISO, 2011. 34 p.
- 14.Jones M., Bradley J., Sakimura N. RFC 7519: JSON Web Token (JWT). Internet Engineering Task Force, 2015. URL: <https://www.rfc-editor.org/rfc/rfc7519> (дата звернення: 15.06.2026).
- 15.Lock A. ASP.NET Core in Action. 2nd ed. Shelter Island : Manning Publications, 2021. 832 p.
- 16.Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Upper Saddle River : Prentice Hall, 2017. 432 p.
- 17.Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship. Upper Saddle River : Prentice Hall, 2008. 464 p.
- 18.Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. Sebastopol : O'Reilly Media, 2021. 612 p.
- 19.Nottingham M., Wilde E. RFC 7807: Problem Details for HTTP APIs. Internet Engineering Task Force, 2016. URL: <https://www.rfc-editor.org/rfc/rfc7807> (дата звернення: 15.06.2026).
- 20.NUnit Documentation. The NUnit Project. URL: <https://docs.nunit.org/> (дата звернення: 15.06.2026).
- 21.PostgreSQL 16 Documentation. The PostgreSQL Global Development Group. URL: <https://www.postgresql.org/docs/16/> (дата звернення: 15.06.2026).
- 22.Provos N., Mazières D. A Future-Adaptable Password Scheme. Proceedings of the FREENIX Track: 1999 USENIX Annual Technical Conference. Monterey, California, 1999. P. 81–91.
- 23.QuestPDF Documentation. URL: <https://www.questpdf.com/documentation> (дата звернення: 15.06.2026).
- 24.React: The library for web and native user interfaces. URL: <https://react.dev/> (дата звернення: 15.06.2026).

		Пастухов О. К.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Савицький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		91

25. Smith J. P. Entity Framework Core in Action. 2nd ed. Shelter Island : Manning Publications, 2021. 624 p.
26. Tailwind CSS Documentation. Tailwind Labs Inc. URL: <https://tailwindcss.com/docs> (дата звернення: 15.06.2026).
27. TypeScript Handbook. Microsoft Corporation. URL: <https://www.typescriptlang.org/docs/handbook/intro.html> (дата звернення: 15.06.2026).
28. Vernon V. Implementing Domain-Driven Design. Boston : Addison-Wesley, 2013. 656 p.
29. Vite – Next Generation Frontend Tooling. URL: <https://vitejs.dev/guide/> (дата звернення: 15.06.2026).
30. Пастухов О. К., Савіцький Р. С. Порівняльний аналіз GraphQL і HATEOAS у розрізі архітектурних рішень сучасних API. Сучасні інформаційні системи та технології : матеріали VIII Всеукраїнської науково-практичної інтернет-конференції молодих вчених та студентів, м. Хмельницький, 24 листопада 2025 р. Хмельницький : ХНУ, 2025.

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Савіцький Р. С.				92
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Пастухов О. К.			ІІЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		93

Лістинг вихідного коду

Лістинг А.1:

```

[HttpPost("{id}/submit")]
public async Task<IActionResult> Submit(Guid id, [FromBody] SubmitTaskDto dto)
{
    var userId = GetUserId();

    var task = await _db.Tasks.FirstOrDefaultAsync(t => t.Id == id);
    if (task == null) return NotFound(new { message = "Завдання не знайдено" });

    try { await _auth.EnsureMemberAsync(task.ProjectId, userId); }
    catch (UnauthorizedAccessException ex) { return StatusCode(403, new { message
= ex.Message }); }

    if (task.AssigneeUserId != userId)
        return StatusCode(403, new { message = "Здавати задачу може тільки
призначений виконавець" });

    if (task.Status != "InProgress" && task.Status != "NeedsRevision")
        return BadRequest(new { message = "Здати можна лише задачу зі статусом «В
роботі» або «Потребує доопрацювання»" });

    if (!dto.SelfChecked)
        return BadRequest(new { message = "Підтвердіть, що ви перевірили роботу
(чекбокс самоперевірки)" });

    var submissionNumber = await _db.TaskSubmissions
        .CountAsync(s => s.TaskItemId == task.Id) + 1;

    var submission = new TaskSubmission
    {
        TaskItemId = task.Id,
        SubmittedByUserId = userId,
        WhatWasDone = dto.WhatWasDone.Trim(),
        Links = NormalizeLinks(dto.Links),
        HoursSpent = dto.HoursSpent,
        SubmissionNumber = submissionNumber
    };

    _db.TaskSubmissions.Add(submission);
    task.Status = "InReview";

```

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		94

```

task.ReviewClaimedByUserId = null;
task.ReviewClaimedAt = null;
task.UpdatedAt = DateTime.UtcNow;

await _db.SaveChangesAsync();

var reviewerIds = await _db.ProjectMemberships
    .Where(pm => pm.ProjectId == task.ProjectId
        && (pm.Role == ProjectRole.Mentor || pm.Role == ProjectRole.Organizer)
        && pm.UserId != userId)
    .Select(pm => pm.UserId)
    .ToListAsync();

foreach (var reviewerId in reviewerIds)
{
    await _notifications.CreateAsync(
        userId: reviewerId,
        type: "TaskSubmitted",
        title: "Задачу здано на перевірку",
        message: $"Задачу «{task.Title}» здано на перевірку (спроба
#{submissionNumber})",
        relatedId: task.Id,
        relatedType: "Task"
    );
}

await _hub.Clients.Group($"project-{task.ProjectId}")
    .SendAsync("TaskStatusChanged", new { taskId = task.Id, status =
"InReview" });

return Ok(new { message = "Задачу здано на перевірку", submissionNumber });
}

[HttpPost("{id}/claim-review")]
public async Task<IActionResult> ClaimReview(Guid id)
{
    var userId = GetUserId();

    var task = await _db.Tasks.FirstOrDefaultAsync(t => t.Id == id);
    if (task == null) return NotFound(new { message = "Завдання не знайдено" });

    var membership = await _db.ProjectMemberships

```

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савицький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		95

```

        .FirstOrDefaultAsync(pm => pm.ProjectId == task.ProjectId && pm.UserId ==
userId);
        if (membership == null)
            return StatusCode(403, new { message = "Ви не є учасником цього проекту"
});
        if (membership.Role != ProjectRole.Mentor && membership.Role !=
ProjectRole.Organizer)
            return StatusCode(403, new { message = "Перевіряти можуть лише ментор або
організатор" });

        if (task.Status != "InReview")
            return BadRequest(new { message = "Задача не на перевірці" });

        if (task.ReviewClaimedByUserId.HasValue)
        {
            var claimerName = await _db.Users
                .Where(u => u.Id == task.ReviewClaimedByUserId.Value)
                .Select(u => u.FirstName + " " + u.LastName)
                .FirstOrDefault() ?? "Хтось";
            return BadRequest(new { message = $"Задачу вже перевіряє {claimerName}"
});
        }

        task.ReviewClaimedByUserId = userId;
        task.ReviewClaimedAt = DateTime.UtcNow;
        task.UpdatedAt = DateTime.UtcNow;

        await _db.SaveChangesAsync();

        await _hub.Clients.Group($"project-{task.ProjectId}")
            .SendAsync("TaskReviewClaimed", new { taskId = task.Id, claimedByUserId =
userId });

        return Ok(new { message = "Ви взяли задачу на перевірку" });
    }

    [HttpPost("{id}/accept")]
    public async Task<IActionResult> Accept(Guid id, [FromBody] ReviewTaskDto dto)
    {
        var userId = GetUserId();

        var task = await _db.Tasks.FirstOrDefaultAsync(t => t.Id == id);
        if (task == null) return NotFound(new { message = "Завдання не знайдено" });

```

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савицький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		96

```

var membership = await _db.ProjectMemberships
    .FirstOrDefaultAsync(pm => pm.ProjectId == task.ProjectId && pm.UserId ==
userId);

if (membership == null || (membership.Role != ProjectRole.Mentor &&
membership.Role != ProjectRole.Organizer))
    return StatusCode(403, new { message = "Приймати можуть лише ментор або
організатор" });

if (task.Status != "InReview")
    return BadRequest(new { message = "Задача не на перевірці" });

if (task.ReviewClaimedByUserId.HasValue
    && task.ReviewClaimedByUserId != userId
    && membership.Role != ProjectRole.Organizer)
{
    return BadRequest(new { message = "Задачу перевіряє інший ментор" });
}

task.Status = "Done";
task.ReviewedByUserId = userId;
task.ReviewComment = dto.Comment?.Trim();
task.ReviewClaimedByUserId = null;
task.ReviewClaimedAt = null;
task.UpdatedAt = DateTime.UtcNow;

if (!task.PointsAwarded && task.AssigneeUserId.HasValue)
{
    await _ratings.AwardForTaskCompletionAsync(
        userId: task.AssigneeUserId.Value,
        priority: task.Priority,
        taskId: task.Id
    );
    task.PointsAwarded = true;
}

await _db.SaveChangesAsync();

await _hub.Clients.Group($"project-{task.ProjectId}")
    .SendAsync("TaskStatusChanged", new { taskId = task.Id, status = "Done"
});

return Ok(new { message = "Задачу прийнято" });

```

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Савіцький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		97

```

}

[HttpPost("{id}/return")]
public async Task<IActionResult> Return(Guid id, [FromBody] ReviewTaskDto dto)
{
    var userId = GetUserId();

    var task = await _db.Tasks.FirstOrDefaultAsync(t => t.Id == id);
    if (task == null) return NotFound(new { message = "Завдання не знайдено" });

    var membership = await _db.ProjectMemberships
        .FirstOrDefaultAsync(pm => pm.ProjectId == task.ProjectId && pm.UserId ==
userId);
    if (membership == null || (membership.Role != ProjectRole.Mentor &&
membership.Role != ProjectRole.Organizer))
        return StatusCode(403, new { message = "Повертати можуть лише ментор або
оператор" });

    if (task.Status != "InReview")
        return BadRequest(new { message = "Задача не на перевірку" });

    if (string.IsNullOrEmpty(dto.Comment))
        return BadRequest(new { message = "Вкажіть причину повернення" });

    task.Status = "NeedsRevision";
    task.ReviewedByUserId = userId;
    task.ReviewComment = dto.Comment.Trim();
    task.ReviewClaimedByUserId = null;
    task.ReviewClaimedAt = null;
    task.UpdatedAt = DateTime.UtcNow;

    await _db.SaveChangesAsync();

    await _hub.Clients.Group($"project-{task.ProjectId}")
        .SendAsync("TaskStatusChanged", new { taskId = task.Id, status =
"NeedsRevision" });

    return Ok(new { message = "Задачу повернуто на доопрацювання" });
}

private static List<string> NormalizeLinks(List<string>? links)
{

```

		Пастухов О. К.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Савицький Р. С.				
Змн.	Арк.	№ докум.	Підпис	Дата		98

```

        if (links == null) return new List<string>();
        return links
            .Where(l => !string.IsNullOrWhiteSpace(l))
            .Select(l => l.Trim())
            .Where(l => l.Length <= 500)
            .Distinct(StringComparer.OrdinalIgnoreCase)
            .Take(10)
            .ToList();
    }

```

Лістинг А.2:

```

using System.ComponentModel.DataAnnotations;

namespace Squadly.Application.DTOs.Tasks;

public class CreateTaskDto
{
    [Required(ErrorMessage = "Назва задачі обов'язкова")]
    [StringLength(200, MinimumLength = 2, ErrorMessage = "Назва має бути від 2 до 200 символів")]
    public string Title { get; set; } = string.Empty;

    [StringLength(2000, ErrorMessage = "Опис не може бути довшим за 2000 символів")]
    public string? Description { get; set; }

    [Required(ErrorMessage = "Статус обов'язковий")]
    [RegularExpression("^(ToDo|InProgress|Done)$", ErrorMessage = "Невірний статус")]
    public string Status { get; set; } = "ToDo";

    [Required(ErrorMessage = "Пріоритет обов'язковий")]
    [RegularExpression("^(Low|Medium|High)$", ErrorMessage = "Невірний пріоритет")]
    public string Priority { get; set; } = "Medium";

    [Required(ErrorMessage = "Проект обов'язковий")]
    public Guid ProjectId { get; set; }
    public Guid? AssigneeUserId { get; set; }
    public DateTime? DueDate { get; set; }
}

```

		Пастухов О. К.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Савіцький Р. С.				99
Змн.	Арк.	№ докум.	Підпис	Дата		