Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

зі спеціальності 121 «Інженерія програмного забезпечення»

На тему: *Розробка інформаційної системи управління проєктною діяльністю студентів за методологією Agile*

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ЗІПЗ-22
_____ *Аліканцев Н. Р.*

Керівник кваліфікаційної роботи _____ */ Стрюк А. М. /*

Завідувач кафедри _____ */ Стрюк А. М. /*

Кривий Ріг
2026

Криворізький національний університет
Факультет: *Інформаційних технологій*
Кафедра: *Моделювання та програмного забезпечення*
Освітньо-кваліфікаційний рівень: *бакалавр*
Спеціальність: *121 "Інженерія програмного забезпечення"*

ЗАТВЕРДЖУЮ
Зав. кафедри
_____*Стрюк А. М.*
«___»_____ 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу
студенту групи ЗІПЗ-22 Аліканцеву Нікіті Романовичу

1. Тема: *Розробка інформаційної системи управління проєктною діяльністю студентів за методологією Agile* затверджено наказом по КНУ № 286 с від «28» травня 2026 р.
2. Термін подання студентом закінченого проекту «20» червня 2026 р.
3. Вихідні дані по роботі: *Пояснювальна записка: 90 сторінок, 12 рисунків, 1 додаток, 28 використаних у роботі джерел*
4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): Аналіз предметної області, проєктування архітектури та бази даних системи, програмна реалізація та тестування інформаційної системи.
5. Перелік графічного демонстраційного матеріалу: Приклади існуючих програм. Діаграма потоків даних. Діаграми використання. Структура бази даних. Приклади роботи розробленої програми.

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	*Формулювання мети та задач роботи*	*13.02.2026-20.02.2026*
2	*Аналіз інформаційних джерел*	*21.02.2026-09.03.2026*
3	*Визначення вимог до програмного забезпечення*	*10.03.2026-18.03.2026*
4	*Розробка функціональної схеми*	*19.03.2026-23.03.2026*
5	*Розробка алгоритмів*	*24.03.2026-31.03.2026*
6	*Розробка бази даних*	*01.04.2026-14.04.2026*
7	*Розробка програмного забезпечення*	*15.04.2026-13.05.2026*
8	*Тестування програмного забезпечення*	*14.05.2026-20.05.2026*
9	*Оформлення пояснювальної записки*	*21.05.2026-12.06.2026*
10	*Розробка демонстраційних матеріалів*	*13.06.2026-17.06.2026*

Дата видачі завдання: «_____»_____ 2026 р.
Студент _____ Аліканцев Н. Р.
Керівник роботи _____ Стрюк А. М.

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи викладена на 90 сторінках, містить 12 рисунків, 1 додаток та 28 використаних літературних джерел.

Об’єкт дослідження – процес управління командною проєктною діяльністю студентів у закладах вищої освіти.

Предмет дослідження – інформаційні системи, гнучкі методології розробки (Scrum, Kanban, eduScrum) та архітектурні рішення для координації студентських команд.

Мета роботи – розробка та дослідження спеціалізованої інформаційної системи управління проєктною діяльністю студентів, яка базується на методології Agile та адаптована до специфічних потреб і графіку освітнього процесу в ЗВО.

В роботі проведено комплексний аналіз предметної області та існуючих PM-платформ, на основі чого сформовано перелік функціональних та нефункциональних вимог до освітньої системи. За допомогою UML-моделювання спроєктовано гнучку архітектуру платформи (модульний моноліт) та розроблено структуру реляційної бази даних. Програмно реалізовано інтерактивну вебсистему, яка підтримує управління беклогом, інтерактивні Канбан-дошки з миттєвою синхронізацією стану, логування витраченого часу, оцінювання спринтів та генерацію аналітичних метрик.

Інтегровано інтелектуальний модуль на базі великих мовних моделей (через API OpenAI / Google Gemini). Він у фоновому режимі аналізує стан проєкту, динаміку виконання завдань та активність розробників для автоматичного формування рекомендацій студентам та аналітичних звітів викладачу.

Ключові слова: AGILE, SCRUM, KANBAN, ІНФОРМАЦІЙНА СИСТЕМА, УПРАВЛІННЯ ПРОЄКТАМИ, ОСВІТНІЙ ПРОЦЕС, ШТУЧНИЙ ІНТЕЛЕКТ, NESTJS, REACT.JS, POSTGRESQL.

ABSTRACT

The explanatory note for this thesis is 90 pages long and includes 12 figures, 1 appendix, and 28 references.

The object of the study is the process of managing student team project activities in higher education institutions.

The subject of the study is information systems, agile development methodologies (Scrum, Kanban, eduScrum), and architectural solutions for coordinating student teams.

The aim of the work is to develop and study a specialized information system for managing student project activities, based on the Agile methodology and adapted to the specific needs and schedule of the educational process in higher education institutions.

The work conducts a comprehensive analysis of the subject area and existing PM platforms, based on which a list of functional and non-functional requirements for the educational system was formed. Using UML modeling, a flexible platform architecture (modular monolith) was designed and the structure of a relational database was developed. An interactive web system was implemented, supporting backlog management, interactive Kanban boards with real-time status synchronization, time tracking, sprint evaluation, and the generation of analytical metrics.

An intelligent module based on large language models (via the OpenAI / Google Gemini API) has been integrated. It analyzes the project status, task execution dynamics, and developer activity in the background to automatically generate recommendations for students and analytical reports for the instructor.

Keywords: AGILE, SCRUM, KANBAN, INFORMATION SYSTEM, PROJECT MANAGEMENT, EDUCATIONAL PROCESS, ARTIFICIAL INTELLIGENCE, NESTJS, REACT.JS, POSTGRESQL.

ЗМІСТ

Вступ 7
1 Аналіз предметної області 9
1.1 Специфіка управління проєктами в освітньому середовищі 9
1.2 Огляд існуючих систем управління проєктами та їх порівняльний аналіз 10
1.3 Принципи Agile та доцільність їх застосування для студентських команд 14
2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА БАЗИ ДАНИХ СИСТЕМИ . 23
2.1 Опис функціональних та нефункціональних вимог 23
2.2 Побудова діаграм варіантів використання та послідовності 31
2.3 Проєктування архітектури системи 33
2.4 Розробка інфологічної та даталогічної моделей бази даних 35
2.5 Проєктування інтерфейсу користувача 41
3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ 44
3.1 Обґрунтування вибору технологічного стеку 44
3.2 Реалізація серверної частини 49
3.3 Розробка клієнтської частини 52
3.4 Огляд розробленої системи 54
ВИСНОВКИ 63
Перелік посилань 65
Додаток А Текст програми 68

ВСТУП

Сучасні університети активно переходять до цифрових форматів навчання та, зокрема, організації проєктної роботи. Групова проєктна робота в університетах завжди стикалась з певним хаосом. Завдання розкидані по повідомлення в системах управління навчанням, месенджерах, прогрес непрозорий, а відповідальність розмита.

В той же час методологія Agile, зокрема Scrum і Kanban, вийшла за межі IT-галузі й успішно застосовується у проєктно-орієнтованому навчанні. Ітеративний підхід, спринти та ретроспективи природньо вписуються в семестровий ритм, а розподіл ролей у команді формує реальні компетентності командної роботи. Крім того Agile-компетентності входять до переліку базових вимог роботодавців у IT-секторі. Система, що занурює студентів у Agile-практики ще під час навчання, підвищить їхню готовність до реального робочого середовища.

Попри широкий вибір загальних систем управління проєктами (Jira, Trello, Asana), вони не адаптовані під специфіку освітнього процесу: навчальні плани та графіки, оцінювання, ролі "студент/викладач" тощо. Розробка єдиного інструменту, що поєднує управління беклогом, розподіл задач і комунікацію, стала б чудовим рішенням для сучасних університетів.

Актуальність розробки спеціалізованої інформаційної системи для закладів вищої освіти зумовлена тим, що сучасні промислові інструменти, такі як Jira або Trello, орієнтовані на комерційний результат і стабільні професійні команди, а не на специфіку освітнього процесу. В освітньому середовищі структура взаємодії є значно складнішою, оскільки викладач виконує роль не просто стейкхолдера, а ментора, якому необхідний доступ до аналітики прогресу всіх груп одночасно, а студенти в межах Agile-методології часто змінюють свої ролі, що вимагає гнучкого налаштування прав доступу. Стандартні системи керування проєктами зазвичай позбавлені вбудованих механізмів для оцінювання індивідуального внеску кожного студента та не

передбачають інтеграції з університетською інфраструктурою, зокрема з розкладом занять для автоматичного планування дедлайнів спринтів.

Крім того, професійні інструменти мають високий поріг входження, через що час на їхнє вивчення може перевищувати тривалість самого навчального проєкту, тоді як специфіка ЗВО вимагає спрощеного інтерфейсу, зосередженого на навчальних цілях, а не на адмініструванні.

Важливо також враховувати методологічний аспект, адже використання Agile в університеті спрямоване не лише на управління задачами, а й на навчання гнучкому мисленню, проте стандартне програмне забезпечення не містить необхідних навчальних підказок чи шаблонів для правильного проведення Scrum-ритуалів. Розробка власного рішення дозволяє не лише адаптувати інструментарій під ці потреби, а й впровадити функції прогнозування термінів на основі попередньої успішності, що є критично важливим для ефективного академічного планування.

Мета роботи полягає у розробці та дослідженні інформаційної системи управління проєктною діяльністю студентів, яка базується на методології Agile та адаптована до специфічних потреб освітнього процесу у закладах вищої освіти.

Завдання роботи:

– проаналізувати особливості управління проєктами в освітньому середовищі та провести порівняльний аналіз існуючих рішень;

– сформувати функціональні та нефункціональні вимоги до системи;

– спроєктувати архітектуру системи та структуру бази даних із використанням UML-моделювання;

– обґрунтувати вибір технологічного стеку та реалізувати програмний продукт;

– розробити специфічний функціонал для студентів та викладачів, включаючи інструменти оцінювання внеску учасників;

– провести тестування системи та оцінити її ефективність.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Специфіка управління проєктами в освітньому середовищі

Впровадження проєктного навчання у вищій освіті є фундаментальним кроком для підготовки інженерів програмного забезпечення, які спроможні задовольнити стрімко зростаючий попит індустрії. Проте пряме перенесення практик комерційного управління проєктами в освітній простір є неможливим через глибокі системні розбіжності між цими середовищами. Університети прагнуть зробити внесок у теорію та сформувати у студентів фундаментальні навички високого рівня, тоді як комерційний сектор обмежений рамками прибутковості, швидкістю виходу на ринок та задоволенням бізнес-запитів клієнтів. Специфіка управління проєктами в освітньому середовищі, зокрема в закладах вищої освіти (ЗВО), суттєво відрізняється від комерційного сектору через пріоритетність навчальних цілей над продуктовими. У комерційному середовищі головною метою є створення працюючого продукту (MVP) у визначені терміни. В освітньому процесі проект є лише інструментом для здобуття компетенцій. Це означає, що навіть якщо проект не був завершений технічно, але студент продемонстрував глибоке засвоєння технологій та методології, освітня мета вважається досягнутою.

Ці розбіжності безпосередньо впливають на чотири ключові виміри управління проєктами: обсяг (цінність), технічну складність, заплутаність взаємовідносин та ступінь невизначеності. В академічному середовищі технічна складність часто є фіксованою для забезпечення рівних умов оцінювання, тоді як заплутаність взаємовідносин (так звана «мала політика» взаємодії зі стейкхолдерами) та невизначеність можуть створювати нелінійні виклики через відсутність досвіду у студентів. Більше того, комерційне управління проєктами фокусується на ефективній доставці конкретного результату, тоді як в освітньому середовищі воно набуває ознак управління програмами, де координується кілька пов'язаних проєктів задля досягнення спільної довгострокової мети – формування професійної компетентності.

В Agile-командах ЗВО ролі адаптуються під навчальні потреби:

– викладач поєднує ролі замовника, власника продукту та ментора. Йому потрібен інструментарій для одночасного моніторингу прогресу десятків команд, що не передбачено стандартними ліцензіями Jira/Trello без складних налаштувань;

– студенти часто змінюють ролі (наприклад, роль Scrum-майстра може переходити від одного студента до іншого кожного спринту), щоб кожен учасник отримав управлінський досвід.

Життєвий цикл студентського проекту жорстко прив’язаний до семестру (15–17 тижнів). Спринти мають узгоджуватися з графіком занять, сесіями та дедлайнами з інших дисциплін. Стандартні системи не враховують "академічне навантаження" та не інтегруються з університетським розкладом.

У комерційних Agile-командах оцінюється успіх всієї команди. В освіті викладач зобов'язаний виставити індивідуальну оцінку кожному. Це створює потребу в інструментах, які автоматично аналізують активність у Git-репозиторіях та кількість закритих тасків саме в контексті конкретної особи, а не лише команди.

На відміну від професіоналів, студенти лише вчаться методології. Інформаційна система для ЗВО повинна містити навчальні підказки, шаблони для ретроспектив та вбудовані гайди з артефактів Scrum, що допомагає уникнути формального підходу до управління.

1.2 Огляд існуючих систем управління проєктами та їх порівняльний аналіз

Ринок програмного забезпечення у 2026 році сегментується на три чіткі категорії: спрощені інструменти для управління завданнями, важкі корпоративні рішення з глибокою кастомізацією та нове покоління збалансованих платформ, які пропонують потужну функціональність без надмірної адміністративної складності. Вибір оптимального інструменту визначається балансом між потребою організації у контролі та швидкості

адаптації системи користувачами. Ключова дилема, з якою стикаються покупці програмного забезпечення, полягає у виборі між простотою швидкого налаштування з мінімальним навчанням та глибоким контролем над ресурсами й портфелями проєктів.

Архітектура платформи є фундаментальним фактором її довгострокового успіху в корпоративному середовищі. monday.com базується на архітектурі Work OS, де робочі процеси будуються з модульних блоків у середовищі з низьким використанням коду. Така модель забезпечує високу гнучкість, спрощуючи крос-департаментську координацію та знижуючи навантаження на IT-департамент. На противагу цьому, Asana використовує класичну проєктно-центричну модель із жорсткою ієрархічною структурою «Команда > Проєкт > Завдання». Хоча це полегшує швидку адаптацію нетехнічними командами, подібна структура ускладнює моделювання багаторівневих бізнес-процесів при масштабуванні. Більше того, для розширеного аналізу Asana вимагає залучення сторонніх систем, наприклад Tableau чи PowerBI, що створює ризик фрагментації даних.

Платформа ClickUp реалізує концепцію консолідації, об'єднуючи текстові документи, чати, цілі та облік часу в одному вікні, щоб зменшити витрати на ліцензування супутнього софту. Проте широта функціоналу ClickUp супроводжується зростанням складності налаштування, а її вкладена структура просторів та папок демонструє уповільнення інтерфейсу при роботі з великими обсягами інформації. Системи типу Jira орієнтовані на методології гнучкої розробки програмного забезпечення, пропонуючи глибоке управління беклогами та релізами, що робить їх незамінними для інженерів, але занадто складними для широкого корпоративного використання. Для компаній, що потребують комплексного управління ресурсами та інтегрованого проєктного обліку, Celoxis пропонує потужні інструменти аналізу витрат, бюджетування та детального розподілу навантаження виконавців. Впровадження інструментів із вбудованими процесами затвердження та чіткими шаблонами

дозволяє мінімізувати операційні ризики та уникнути хаосу під час масштабування бізнесу.

Для здійснення всебічного аналізу доцільно розглянути ключові інструменти крізь призму їхніх технічних параметрів, колаборативних властивостей та відповідності сучасним вимогам до управління ресурсами й процесами узгодження.

В таблиці 1.1 представлено основні технічні та операційні параметри систем управління проєктами.

Таблиця 1.1 – Основні технічні та операційні параметри систем управління проєктами.

Платформа	**Сфера найкращого застосування**	**Підтримувані платформи**	**Доступ-ність API**	**Рівень навчання та адаптації**	**Режими технічної підтримки**
Jira	Розробка програмного забезпечення, Agile-команди	SaaS/Web, iOS, Android, Windows, Mac	Доступно	Високий (потребує навчання)	24/7, робочі години, онлайн
Asana	Загальний бізнес, маркетинг, креативні агенції	SaaS/Web, iOS, Android, Windows, Mac	Доступно	Середній (швидке впровадження)	24/7, робочі години, онлайн
monday.com	Крос-функціональні команди, візуальні процеси	SaaS/Web, iOS, Android, Windows, Mac	Доступно	Низький (простий drag-and-drop)	24/7, робочі години, онлайн
ClickUp	Універсальне управління, консолідація інструментів	SaaS/Web, iOS, Android, Windows, Mac	Доступно	Середній/Високий (складна логіка)	24/7, робочі години, онлайн
Stackfield	Регульовані сектори, безпечна командна робота	SaaS/Web, iOS, Android, Windows, Mac	Доступно	Низький (інтуїтивний інтерфейс)	Телефонна підтримка, онлайн-чат
Celoxis	Великі проєкти, інтегрований фінансовий облік	SaaS/Web, iOS, Android, Windows, Mac	Доступно	Середній	Робочі години, онлайн

В таблиці 1.2 зроблено порівняння функціональних можливостей для колаборації в популярних системах управління проєктами.

Таблиця 1.2 – Порівняння функціональних можливостей для колаборації

Платформа	**Робота з коментарями**	**Управління файлами**	**Система сповіщень**	**Зовнішній доступ**	**Вбудований облік часу**
Jira	Гілки на рівні завдань	Базове лінкування файлів	Налаштовувані алерти	Обмежений зовнішній доступ	Потребує інтеграцій/плагінів
Asana	Коментарі до проєктів/завдань	Інтегрований прев'ю-перегляд	Розумне фільтрування сповіщень	Розширений клієнтський доступ	Потребує інтеграцій/плагінів
monday.com	Обговорення елементів	Візуальний менеджмент файлів	Персоналізовані стрічки оновлень	Гнучкий доступ стейкхолдерів	Потребує інтеграцій/плагінів
ClickUp	Контекстний чат, коментарі	Завантаження в хмару	Стрічки активності в просторах	Гнучкі гостьові права	Нативний вбудований трекер
Stackfield	Інтегрований командний чат	Захищене хмарне сховище	Налаштовувані сповіщення	Контрольовані гостьові кімнати	Вбудований облік часу
Celoxis	Коментарі та дискусії	Прикріплення файлів до задач	Автоматичні сповіщення	Клієнтський портал	Вбудований тайм-трекінг

В таблиці 1.3 представлено порівняльний аналіз платформ управління проєктами за складністю адміністрування.

Таблиця 1.3 – Порівняльний аналіз платформ управління проєктами

Платформа	**Готові шаблони процесів**	**Вбудовані процеси затвердження**	**Контроль завантаженості персоналу**	**Складність адміністрування**	**Оцінка Capterra**
Workzone	Наявні з коробки	Вбудовані, багатокрокові	Повноцінний контроль лімітів	Низька	4.8 / 5
monday.com	Наявні, висока кастомізація	Слабкі, потребують ручного налаштування	Помірний моніторинг	Середня	4.6 / 5
Adobe Workfront	Висока складність шаблонів	Вбудовані, автоматизовані	Глибоке стратегічне планування	Висока (потрібен адмін)	4.4 / 5
Wrike	Наявні, потребують адаптації	Вбудовані, багатокрокові	Розвинене ресурсне планування	Висока	4.4 / 5

Smartsheet	Базові табличні моделі	Обмежені вбудовані затвердження	Помірний рівень контролю	Висока (через формули)	4.5 / 5
Asana	Базові легковагі шаблони	Лише у вигляді типу задач	Базовий аналіз завантаження	Низька	4.5 / 5
ClickUp	Наявні, висока деталізація	Обмежені вбудовані затвердження	Помірний моніторинг	Висока	4.6 / 5
Jira	Орієнтовані на Agile-спринти	Обмежені, технічні	Базове ресурсне планування	Висока	4.4 / 5

Окремої уваги заслуговує аналіз специфіки управління проєктами в освітньому середовищі, зокрема у вищих навчальних закладах. Стандартні комерційні платформи, такі як Jira чи Trello, часто є неоптимальними для координації студентської командної роботи над кваліфікаційними чи курсовими проєктами. Це зумовлено відсутністю специфічних академічних функцій, таких як пряма інтеграція з навчальним розкладом, автоматичне оцінювання індивідуального внеску кожного студента у загальний результат та невідповідність стандартних комерційних ролей педагогічним процесам. З цієї причини обґрунтованою та актуальною є розробка власних спеціалізованих інформаційних систем управління проєктною діяльністю студентів на основі методології Agile.

1.3 Принципи Agile та доцільність їх застосування для студентських команд

Гнучка парадигма управління проектами, відома як Agile, виникла як альтернатива традиційним лінійним та каскадним моделям життєвого циклу розробки (Waterfall), що вирізнялися жорсткістю та вразливістю перед динамічними змінами середовища. В основі Agile лежить система пріоритетів, зафіксована у Маніфесті гнучкої розробки, яка висуває на перший план людей та взаємодію між ними, створення працездатного продукту, постійну комунікацію із замовником та готовність оперативно реагувати на

непередбачувані виклики. Методологія базується на емпіричному контролі процесів, що спирається на три фундаментальні стовпи: повну прозорість усіх операцій, регулярну інспекцію проміжних результатів та своєчасну адаптацію інструментарію до нових умов. Дванадцять взаємопов'язаних принципів Маніфесту гнучкої розробки орієнтовані на оптимізацію інформаційного обміну, мінімізацію бюрократичних процедур та створення умов для самоорганізації команд.

Серед розмаїття практичних інструментів Agile ключові позиції займають фреймворки Scrum та Kanban. Scrum пропонує структурований ітераційний підхід, де проект розбивається на чіткі часові відрізки — спринти тривалістю від одного до чотирьох тижнів, протягом яких команда фокусується на створенні конкретного інкременту продукту. Цей фреймворк передбачає розподіл ролей (Власник продукту, Скрам-майстер, Команда розробки) та проведення обов'язкових нарад: планування спринту, щоденного стендапу, огляду результатів та ретроспективи. На противагу цьому, Kanban орієнтований на управління безперервним потоком робіт. Він не регламентує специфічних командних ролей або фіксованих часових циклів, натомість пропонуючи гнучке регулювання через візуальні дошки та ліміти на обсяг незавершеної роботи (WIP limits), що дозволяє запобігати перевантаженню учасників та виявляти вузькі місця у процесі розробки.

У разі, коли Scrum виявляється занадто жорстким, а Kanban — занадто аморфним для конкретного проекту, команди застосовують гібридний фреймворк Scrumban, який гармонійно поєднує структуру Скраму із гнучкістю та орієнтацією на потік Канбану. Окрім того, в інженерних проектах використовуються інші гнучкі інструменти, такі як Екстремальне програмування (XP) та Гнучке моделювання (Agile Modeling). Екстремальне програмування передбачає застосування специфічних інженерних практик: парного програмування, колективного володіння кодом, безперервної інтеграції, тестування перед написанням коду (TDD) та підтримки сталого темпу роботи. Для забезпечення технологічної підтримки цих фреймворків

використовується широкий спектр спеціалізованого програмного забезпечення.

Університетське середовище часто стикається з кризою мотивації під час виконання класичних групових завдань. Традиційні підходи створюють умови для прокрастинації, коли реальна діяльність розпочинається безпосередньо перед дедлайном, що суттєво знижує когнітивну цінність групової роботи та призводить до нерівномірного розподілу навантаження. Застосування Agile-підходів у студентських проектах дозволяє вирішити цю проблему шляхом перетворення академічного процесу на серію коротких циклів із видимими та вимірюваними проміжними результатами. Це дозволяє студентам краще керувати обсягом завдань, оцінювати складність роботи за допомогою послідовності Фібоначчі та самостійно визначати пріоритетність кроків для досягнення навчальних цілей.

Впровадження гнучких практик забезпечує інтенсивний розвиток м'яких навичок (soft skills). Результати академічних досліджень підтверджують позитивний вплив Agile на такі компетенції, як комунікація, координація дій, згуртованість команди, лідерські якості та рівень взаємної довіри. Зокрема, емпіричне дослідження процесу навчання на основі викликів (Challenge Based Learning), у якому взяли участь 45 студентів бакалаврату, об’єднаних у 9 Scrum-команд протягом 10 тижнів, продемонструвало, що закріплення конкретних лідерських ролей (як-от Власник продукту чи Скрам-майстер) значно підвищує рівень відповідальності, стимулює залученість та покращує здатність узгоджувати індивідуальні інтереси з колективними цілями. Аналогічно, досвід залучення аспірантів як Скрам-майстрів для координації університетських робочих та сервісних комітетів підтвердив високу ефективність фреймворку для розвитку управлінських навичок у молодих науковців.

Гнучкі методології демонструють життєздатність у різних освітніх галузях. Дослідження думок студентів медичного профілю (зокрема, майбутніх медичних сестер) за допомогою спеціалізованого опитувальника

сприйняття Agile Scrum (ASPT) підтвердили спроможність фреймворку покращувати результати спільного навчання та розвивати здатність до рефлексії. В інженерній освіті підходи Екстремального програмування та Гнучкого моделювання забезпечують адаптивність до мінливих вимог академічних завдань. Практична реалізація гнучких інструментів у форматі «живих лабораторій» (Live-Labs), наприклад, у Дослідницькому інституті розробки продуктів (IPEK) при Технологічному інституті Карлсруе (KIT), дозволяє студентам магістратури працювати над реальними промисловими викликами разом із експертами. Студенти за короткі спринти створюють до чотирьох послідовних прототипів інженерних рішень, що забезпечує миттєвий зворотний зв'язок та глибоке практичне засвоєння методології проектування.

Навіть у чистому науковому пошуку гнучкі інструменти демонструють свою спроможність. Досвід multidisciplinary-дослідницької команди, яка впроваджувала Kanban протягом 10 тижнів без фіксованих спринтів, але з безперервним візуалізованим потоком завдань, підтвердив суттєве покращення координації. Використання простих цифрових рішень (Telegram для швидких обговорень, Mailman для офіційного листування, Google Docs для модульного написання тексту) дозволило збалансувати навантаження та успішно підготувати складний 145-сторінковий звіт у визначений термін, забезпечивши високу якість інтеграції різних дисциплінарних підходів.

На стику педагогіки та гнучкого управління виник спеціалізований фреймворк eduScrum, розроблений Віллі Війнандсом як адаптація класичного Scrum для середньої та вищої школи. В Україні цей освітній інструмент набув поширення завдяки науково-методичній діяльності дослідників, зокрема перекладу та адаптації офіційного посібника під редакцією В. Бурячка та В. Соколова на базі Київського університету імені Бориса Грінченка. Головний принцип eduScrum полягає у переході від директивного навчання до самоорганізованого співробітництва, де викладач виступає у ролі Власника продукту (Product Owner) та формує завдання відповідно до навчальної

програми («Що» та «Чому»), тоді як студентські команди повністю беруть на себе відповідальність за планування та реалізацію проекту («Як»).

Команда в eduScrum складається з викладача (Власника продукту) та малих груп студентів чи учнів (по 4–5 осіб), що гарантує високу динаміку та запобігає розпорошенню відповідальності. Один із членів студентської групи обирається капітаном (eduScrum Master / team captain) та відповідає за дотримання правил взаємодії, згуртованість команди та моніторинг виконання завдань. Життєвий цикл проекту організується через спринти тривалістю до семи тижнів, що включають такі події:

– етап формування команд на основі аналізу індивідуальних навичок, детального роз'яснення викладачем критеріїв успішності та визначення студентами переліку завдань.

– щоурочна п'ятихвилинна нарада на початку заняття, під час якої учасники актуалізують інформацію про хід робіт, планують діяльність на поточне заняття та обговорюють блокери.

– демонстрація створеного продукту або захист проекту, супроводжувана тестуванням або іншими формами контролю знань.

– командний аналіз взаємодії, оцінка ефективності інструментів, обговорення особистісного зростання та визначення кроків для оптимізації процесів.

– самоаналіз індивідуальних досягнень, сильних сторін та областей для подальшого розвитку.

Основними інструментами візуалізації в eduScrum є спеціальна паперова чи цифрова дошка — «флап» (Flap / flipboard), де завдання розподіляються за колонками «Треба зробити», «В роботі» та «Зроблено», а також діаграма згоряння (burndown chart) для моніторингу відповідності часовому графіку. Крім цього, команди формулюють Definition of Doing / Done (критерії готовності завдань), Definition of Communication (правила комунікації) та Definition of Fun (елементи здорового та захопливого командного мікроклімату).

Аналіз впровадження eduScrum у закладах вищої освіти свідчить про глибокі трансформаційні наслідки для академічної культури. Півторарічний експеримент із впровадження цієї методології в інституті вищої освіти в басейні Амазонки показав поступовий відхід від традиційних лекційних занять до активного вирішення прикладних проблем. Студенти продемонстрували здатність самостійно оцінювати складність завдань, планувати навантаження та долати перешкоди без безпосереднього втручання викладача, що наблизило академічне середовище до умов реальної професійної діяльності. В іншому успішному кейсі впровадження eduScrum на магістерському рівні під час вивчення аналізу соціальних мереж студенти працювали малими групами над самостійно обраними науковими проектами та спільно писали дослідницькі статті. Весь процес супроводжувався професором та асистентом, які надавали безперервний зворотний зв’язок відповідно до гнучких канонів.

Хакатони представляють собою унікальну форму короткострокової екстремальної колаборації, під час якої студентські команди зазвичай протягом 24–48 годин інтенсивно розробляють інноваційні рішення для реальних індустріальних чи суспільних викликів. Якщо у великих корпоративних структурах проведення хакатонів іноді викликає занепокоєння менеджменту через тимчасове зупинення регулярної діяльності та зниження поточних операційних показників (що компенсується зростанням лояльності, розвитком компетенцій та перевіркою сміливих гіпотез), то в освітньому середовищі хакатони виступають виключно як каталізатор прискореного навчання.

Дослідження інженерної освіти підтверджують, що хакатони мають колосальний потенціал як платформи для емпіричного навчання (experiential learning). Основними рушійними силами прискореного розвитку навичок проектування під час таких марафонів є конкуренція (86,4% респондентів), жорсткий часовий тиск (68,2%) та нагальна потреба швидко опановувати нові технології для створення продукту (59,1%).

Ключовим фактором успіху студентської команди на хакатоні є позитивний досвід розробника (Developer Experience — DevEx), що охоплює легкість розгортання робочого середовища, наявність шаблонів та чіткої документації, а також використання ефективних каналів зв’язку. Щоб уникнути організаційних та технічних блокерів у стислі терміни, команди мають орієнтуватися на принципи швидкої розробки та створення мінімально життєздатного продукту (MVP) замість намагання створити ідеально відшліфоване рішення. Для цього використовується стандартний стек інструментів швидкої розробки (GitHub, Figma, Slack, AWS).

Прикладом успішної реалізації цієї стратегії є перемоги студентів Варшавського технологічного університету (WUT) на престижному європейському хакатоні HackYeah. Зокрема, студентська команда під промовистою назвою «Agile, Determined, Effective» виборола перше місце у своїй категорії, створивши за 24 години повноцінну інтерактивну освітню гру на React для симуляції життєвих рішень та управління фінансами. Хакатони також мають важливе значення для руйнування професійних бар'єрів між студентами різних спеціальностей, змушуючи розробників виходити за рамки суто технічних завдань та вчитися презентувати цінність продукту через призму орієнтованого на людину дизайну.

Попри доведену ефективність, впровадження гнучких методологій у студентському середовищі пов’язане з ризиками, що виникають через специфіку академічної взаємодії. Найбільш руйнівним явищем для командної роботи є ефект вільного вершника (free-riding), коли окремі студенти ухиляються від виконання завдань, розраховуючи отримати загальну оцінку завдяки зусиллям більш мотивованих колег. Масштаб цієї проблеми безпосередньо залежить від чисельності групи. Емпіричний аналіз командної роботи показав, що групи з 8 осіб демонструють значно вищу варіативність індивідуального внеску та схильність до внутрішньої дезорганізації, порівняно з малими групами з 4 осіб.

Окрім того, студенти часто стикаються з психологічними бар'єрами під час вибору внутрішньокомандних систем розподілу винагороди. У межах експериментального дослідження поведінки самоврядних команд, у якому взяли участь 405 осіб у складі 135 тріад, порівнювалися три механізми розподілу фінансового бонусу: рівний розподіл (even split), колективні торги (bargaining) та неупереджений розподіл (impartial sharing, за якого частку кожного учасника визначають рішення двох інших членів команди на основі спостережень за його зусиллями). Дослідження встановило, що рівний розподіл послаблює стимули до праці та відкриває можливості для соціального ледарства. Натомість торги та неупереджений розподіл здатні дисциплінувати ледарів, забезпечуючи зростання загальної продуктивності команди на 50% (а у пізніших періодах — на понад 100%) та покращуючи суб'єктивну оцінку згуртованості команди. Проте, маючи свободу вибору, абсолютна більшість команд добровільно обирала саме рівний розподіл (55% у випадку порівняння з торгами та 78% у порівнянні з неупередженим розподілом). Це свідчить про нездатність студентів ex-ante передбачити руйнівний вплив навіть незначної меншості недобросовісних учасників на спільну мотивацію.

Додаткові складнощі виникають при взаємодії гнучких студентських команд із ширшим негнучким середовищем (проблема організаційної гнучкості – organizational agility). Навчальні плани, традиційні критерії оцінювання та відсутність прямого доступу до реальних кінцевих користувачів створюють інституційні бар'єри, що обмежують повноцінний ітераційний цикл розробки та вимагають гнучкого балансування між традиційними та гнучкими методами управління. Для подолання зазначених викликів викладачам необхідно застосовувати структурні методи підтримки команд.

Теоретичний аналіз та узагальнення практичного досвіду впровадження Agile-принципів у вищій школі дозволяють сформулювати ключові висновки та рекомендації для оптимізації групової роботи студентів:

– студентські проектні групи мають обмежуватися 4–5 учасниками. Перевищення цієї межі критично підвищує ризики виникнення ефекту вільного вершника та ускладнює координацію завдань.

– вибір методології має залежати від специфіки проекту. Для складних інженерних та дослідницьких робіт із фіксованими дедлайнами доцільно застосовувати Scrum або Scrumban, тоді як для потокових завдань, підготовки звітів чи сервісної діяльності ефективнішим є Kanban із обов'язковим лімітуванням незавершеної роботи (WIP limits).

– застосування паперових чи цифрових дошок із використанням колірного кодування стікерів та діаграм згоряння є критично важливим для створення прозорого робочого середовища та попередження прокрастинації.

– успішне впровадження Agile вимагає попередньої підготовки як викладачів, так і студентів. Освітні установи мають інвестувати у формування гнучкого мислення (Agile Mindset) через серію тренінгів та воркшопів перед початком виконання тривалих командних проектів.

– категорично не рекомендується використовувати систему рівного розподілу балів за умовчанням. Викладачі повинні застосовувати моделі нормалізованого взаємооцінювання та групові контракти для забезпечення справедливого розподілу індивідуальних оцінок на основі реального внеску кожного студента в загальний успіх команди.

2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА БАЗИ ДАНИХ СИСТЕМИ

2.1 Опис функціональних та нефункціональних вимог

Процес розробки інформаційної системи вимагає чітового розмежування її функціональних можливостей, які визначають безпосередні дії системи, та нефункціональних обмежень, що описують якість та поведінку платформи. Для задоволення потреб студентів та викладачів система має підтримувати ієрархічну модель представлення вимог, дозволяючи декомпонувати масштабні навчальні цілі на конкретні вимірювані завдання. Це досягається шляхом створення гнучкого беклогу продукту, де викладач виступає у ролі власника продукту, визначаючи пріоритети та глобальне бачення проєкту.

Студенти, організовані у самокеровані команди чисельністю від 5 до 9 осіб, повинні мати можливість здійснювати планування спринтів, створюючи беклог спринту (Sprint Backlog) та декомпонуючи користувацькі історії (User Stories) на технічні завдання (Tasks) та підзадачі (Subtasks). Інтерфейс управління завданнями має бути адаптований до рівня підготовки студентів та специфіки навчальної дисципліни. Для студентів-початківців або непрофільних спеціальностей система має пропонувати спрощені візуальні інструменти на зразок Trello чи Asana, тоді як для майбутніх IT-фахівців необхідна інтеграція з професійними платформами типу Jira, GitHub Projects або Azure DevOps.

В таблиці 2.1 ми узагальнили вимоги до системи управління проєктами відповідно до особливості потреб цільових груп користувачів.

Важливим аспектом командної роботи є створення єдиного інформаційного простору для накопичення знань та асинхронної комунікації. Система має містити вбудовані модулі за типом Confluence для ведення проєктної документації та підтримувати інтеграцію з інструментами відеофіксації на кшталт Loom для швидкого обміну асинхронними оновленнями. Спільна робота над документами вимагає наявності розвиненої

підсистеми управління файлами, яка підтримує хмарні сховища, різні рівні доступу (читання, редагування, рецензування) та автоматизовані процеси затвердження проміжних результатів.

Таблиця 2.1 – Класифікація вимог відповідно до цільових користувачів.

Категорія студентів	**Рекомендований інструментарій**	**Основні функціональні переваги**	**Рівень складності інтерфейсу**
Початківці (нетехнічні спеціальності)	Trello, Asana, Notion, Microsoft Planner.	Простий Kanban-інтерфейс, мінімальний поріг входження, базові шаблони, коментарі.	Низький (drag-and-drop, інтуїтивно зрозумілі картки).
Групові проєкти середньої складності	ClickUp, Monday.com, ProofHub, Zoho Projects.	Універсальне управління задачами, відстеження часу, розширене налаштування робочих процесів.	Середній (потребує базового ознайомлення з Agile-термінологією).
Професійна IT-розробка та інженерія	Jira Software, GitHub Projects, Azure DevOps.	Інтеграція з системами контролю версій (Git), автоматизація CI/CD, управління дефектами.	Високий (орієнтований на підготовку до реальної індустріальної практики).

Таблиця 2.2 – Вимоги до функціональності та взаємодії користувачів

Функціональна область	**Системна вимога до реалізації**	**Детальний опис механізму та взаємодії користувачів**
Управління беклогом	Формування та пріоритезація вимог.	Створення карток вимог з можливістю додавання критеріїв приймання (Acceptance Criteria) та відносною оцінкою в Story Points.
Візуальна дошка (Board)	Візуалізація робочого процесу в реальному часі.	Підтримка зміни статусів перетягуванням (drag-and-drop), налаштування WIP-лімітів, фільтрація за виконавцями та тегами.
Щоденні зустрічі (Daily Scrum)	Фіксація результатів та перешкод.	Шаблони для щоденного звітування тривалістю до 15 хвилин, ведення журналу виявлених перешкод (impediments log).
Демонстрація та огляд (Review)	Оцінювання працездатного інкременту.	Можливість прикріплення демо-матеріалів, відеозаписів, посилань на репозиторії, збір фідбеку від викладачів та замовників.
Ретроспектива (Retrospective)	Аналіз та покращення командних процесів.	Інтерактивна дошка рефлексії для фіксації позитивних та негативних

		моментів ітерації, анонімне голосування за покращення.
Метрики та аналітика	Візуалізація темпів та прогресу навчання.	Автоматична генерація Burndown-діаграм, звітів про швидкість (Velocity) на основі виконання запланованих навчальних цілей.
Управління файлами	Безпечне хмарне збереження та обмін.	Інтеграція з Google Drive, OneDrive або внутрішнім сховищем, підтримка різних рівнів доступу та версійності документів.
Сповіщення та планування	Інформування про зміни та дедлайни.	Надсилання миттєвих повідомлень через email, Slack або Teams, інтеграція проєктного календаря з календарем Moodle.

Для підготовки майбутніх інженерів програмного забезпечення критично важливим є інтеграція функцій контролю якості коду. Система повинна містити спеціалізовані модулі управління тестуванням (Test Management), які дозволяють студентам створювати вручну та автоматично верифікувати тест-кейси, пов'язуючи їх із конкретними користувацькими історіями. Це дозволяє реалізувати принципи екстремального програмування щодо безперервного тестування (continuous testing), мінімізуючи ризик накопичення технічного боргу.

2.1.1 Функціональні вимоги до модуля взаємного та самооцінювання

Групова робота в академічному середовищі потребує об'єктивних інструментів оцінювання індивідуального внеску, щоб запобігти виникненню міжособистісних конфліктів та забезпечити справедливість виставлення підсумкових балів. Науково обґрунтована модель оцінювання базується на безперервному циклі, що охоплює залучення до виконання завдання, детальний аналіз результатів колег, надання розгорнутого фідбеку та подальше рецензування власної роботи на основі отриманих коментарів. Така практика суттєво підвищує залученість студентів, розвиває навички саморегуляції, критичного мислення та сприйняття конструктивної критики.

Для підвищення точності оцінювання викладач має використовувати чітко структуровані критеріальні рубрики, які виключають неоднозначність трактування рівнів виконання (наприклад, заміну розмитих слів "добре" чи

"відмінно" на конкретні вимірювані показники частоти, обсягу або якості роботи). Система повинна підтримувати різноманітні моделі оцінювання ефективності роботи команд, зокрема модель GRPI (цілі, ролі, процеси, міжособистісні стосунки), яка дозволяє оцінити зрілість команди, та систему CATME, що базується на п'яти ключових вимірах командної взаємодії. Також обов'язковою є реалізація моделі RISE (Reflect, Inquire, Suggest, Elevate), яка структурує процес написання текстових відгуків, спонукаючи студентів до глибокого аналізу та надання конструктивних рекомендацій для покращення роботи колег.

Таблиця 2.3 – Узагальнення вимог відповідно до моделі оцінювання

Модель оцінювання	**Ключові виміри**	**Механізм реалізації в системі**	**Дидактична мета використання**
Модель GRPI	1. Чіткість цілей 2. Розподіл ролей 3. Узгодженість процесів 4. Міжособистісні стосунки.	Анонімне анкетування членів команди на 3-му, 10-му та 15-му тижнях семестру з візуалізацією динаміки у вигляді радарної діаграми.	Діагностика організаційного здоров'я команди та виявлення прихованих конфліктів на ранніх етапах.
Модель CATME	1. Сприяння роботі команди 2. Взаємодія з колегами 3. Контроль якості результатів 4. Наявність релевантних знань 5. Стимулювання активності.	Заповнення деталізованих оціночних форм за кожним виміром з використанням 5-бальної шкали Лайкерта.	Формування всебічного профілю індивідуального внеску та виявлення "пасивних" учасників.

Модель RISE	1. Рефлексія (Reflect) 2. Дослідження (Inquire) 3. Пропозиція (Suggest) 4. Покращення (Elevate).	Текстові поля введення з інтегрованими підказками та валідацією довжини коментаря для кожного етапу моделі.	Розвиток навичок написання високоякісного, конструктивного та дієвого зворотного зв'язку.

Процес підготовки та проведення взаємного оцінювання потребує ретельного планування з боку викладача. Програмне забезпечення має підтримувати гнучке налаштування параметрів оцінювання, включаючи можливість проведення подвійного сліпого рецензування (double-blind), де імена авторів та рецензентів повністю приховані для усунення суб'єктивного впливу особистих взаємин на виставлення балів. Окрім загальних командних метрик, система повинна підтримувати специфічні критерії оцінювання для різних типів завдань. Наприклад, при оцінюванні програмного коду система має автоматично або напівавтоматично аналізувати роботи за чотирма фундаментальними критеріями: коректність функціоналу, підтримуваність (maintainability), читабельність та якість документації коду.

2.1.2 Нефункціональні вимоги, безпека та відповідність регуляторним стандартам

Нефункціональні вимоги визначають обмеження та якісні характеристики інформаційної системи, що гарантують її стабільність, безпеку та зручність використання у високонавантаженому академічному середовищі. Відповідно до визначення інформаційної системи за державними стандартами України, вона є сукупністю технічного, програмного та організаційного забезпечення, а також персоналу, призначеною для своєчасного надання верифікованої інформації належним користувачам.

Основні атрибути якості системи включають зручність використання (Usability), продуктивність (Performance), надійність (Reliability) та безпеку

(Security). Для забезпечення належного рівня інтерфейсу система має фокусуватися на ергономіці: налаштуванні індивідуальних колірних схем, оптимізації шрифтів, використанні білого простору для підвищення сканованості інформації на робочих панелях (dashboards). Також обов'язковою є підтримка адаптивної верстки для коректного відображення на мобільних пристроях під управлінням операційних систем iOS та Android.

Таблиця 2.3 – Узагальнення нефункціональних вимог

Категорія вимог	**Показник (Метрика)**	**Технічні вимоги та системні обмеження**	**Регуляторний або галузевий стандарт**
Продуктивність (Performance)	Час відгуку системи (Response Time).	Завантаження головної сторінки та інтерактивної дошки до 2.0 секунд; синхронізація статусів через WebSockets до 500 мс.	Галузеві стандарти розробки високопродуктивних веб-додатків.
Надійність (Reliability)	Коефіцієнт готовності (Uptime).	Рівень доступності системи не менше 99.9% (три дев'ятки); автоматичне щоденне резервне копіювання даних.	Вимоги до критично важливих освітніх ІС.
Масштабованість (Scalability)	Одночасні підключення (Concurrent Users).	Підтримка до 5000 активних сесій одночасно без деградації продуктивності бази даних.	Вимоги до масштабування хмарних платформ.
Безпека (Security)	Захист даних та шифрування.	Обов'язкове використання протоколу HTTPS (TLS 1.3); автентифікація через OAuth2/OIDC; шифрування персональних даних та оцінок.	Закон України "Про захист персональних даних"; стандарт LTI 1.3 Core Security.
Конфіденційність (Privacy)	Керування доступом.	Рольова модель доступу (RBAC); можливість створення приватних робочих зон для чутливих академічних даних.	Політики безпеки закладів вищої освіти.
Регуляторна сумісність	Стандарти впровадження.	Проходження етапів попередніх	Постанова КМУ щодо створення та

		випробувань, дослідної експлуатації та верифікації працездатності.	адміністрування засобів інформатизації.
Локалізація	Мова інтерфейсу.	Повна локалізація інтерфейсу користувача, документації та системних сповіщень українською мовою.	Закон України "Про забезпечення функціонування української мови як державної".

Реалізація нефункціональних вимог гарантує, що система буде не лише функціонально повною, але й здатною витримувати пікові навантаження під час масових студентських здач проєктів (дедлайнів), забезпечуючи захист персональних даних від несанкціонованого доступу.

Для успішного практичного старту системи в освітньому процесі критично важливим є плавний, поетапний перехід студентів до нових інструментів. Рекомендується починати впровадження з простіших візуальних платформ для молодших курсів або непрофільних спеціальностей, щоб закласти базове розуміння Agile-процесів, і лише після цього переходити до складніших індустріальних рішень на старших курсах профільних програм. Такий підхід допомагає уникнути когнітивного перевантаження та сприяє м'якій адаптації до нових робочих просторів.

Важливою організаційною рекомендацією є впровадження обов'язкового орієнтаційного спринту (Orientation Sprint) на початку кожного навчального курсу. Цей підготовчий період має бути присвячений виключно технічному ознайомленню користувачів з інтерфейсом системи, первинному формуванню команд та спільному налаштуванню каналів комунікації. Проведення такого спринту суттєво знижує тривожність студентів перед початком реальної проєктної роботи та запобігає виникненню технічних помилок у критичні моменти здачі завдань.

З метою підвищення мотивації та залученості здобувачів вищої освіти доцільно залучати їх до процесу спільного проектування (co-creation) оціночних критеріїв.

1. Коли студенти беруть активну участь в обговоренні та розробці дескрипторів для рубрик взаємного оцінювання, вони починають глибше усвідомлювати цілі навчання та відчувати вищий ступінь відповідальності за власні та командні результати.

2. Це також мінімізує майбутні дискусії та конфлікти навколо виставлених одногрупниками балів.

3. Окрему увагу слід приділити забезпеченню психологічної безпеки всередині студентських колективів. Система повинна мати вбудовані технічні механізми для проведення конфіденційних опитувань та подвійного сліпого оцінювання, що повністю виключає тиск авторитетів чи особистих симпатій на підсумкові результати. Можливість анонімно фіксувати проблеми та висловлювати пропозиції під час ретроспектив допомагає командам своєчасно виявляти та вирішувати приховані конфлікти, фокусуючись на безперервному самовдосконаленні.

У довгостроковій перспективі розвиток розробленої системи має бути орієнтований на розширення її інтеграційних можливостей за рахунок використання технологій штучного інтелекту, що взаємодіють через Moodle Analytics API. Автоматизований аналіз поведінкових патернів на Agile-дошках та результатів проміжного оцінювання дозволить завчасно виявляти команди з високим ризиком дезорганізації та надавати викладачам точкові рекомендації щодо коригування навчального процесу. Впровадження системи згідно з викладеними технічними та методичними вимогами забезпечить підготовку висококваліфікованих, конкурентоспроможних фахівців, адаптованих до стандартів сучасної гнучкої розробки.

2.2 Побудова діаграм варіантів використання та послідовності

Наступним етапом проєктування системи управління проєктами в освітньому середовищі стала побудова діаграма варіантів використання (рис. 2.1). Ця діаграма відображає межі системи, акторів (користувачів) та їхню взаємодію з функціоналом платформи.

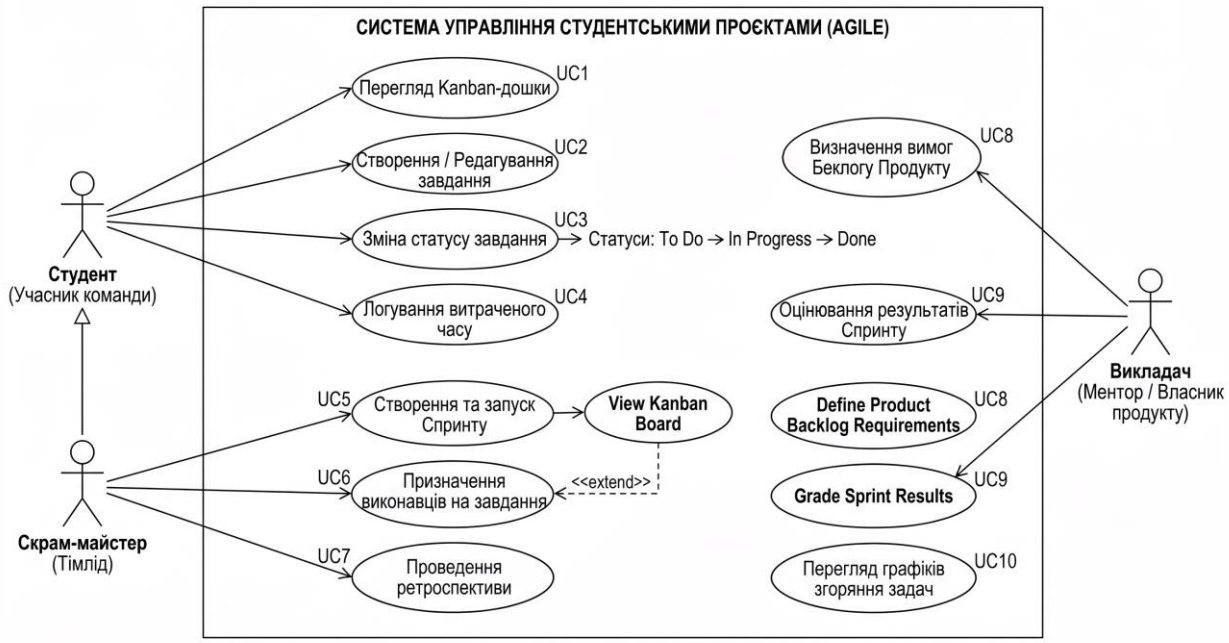

Рисунок 2.1 – Діаграма варіантів використання

Головні актори:

– Студент (Учасник команди). Створює завдання, рухає їх по дошці, бере участь у плануванні.

– Скрам-майстер / Тімлід (Студент-лідер). Керує спринтами, розподіляє ролі в команді.

– Викладач / Ментор (Product Owner). Створює беклог вимог (критерії проєкту), оцінює спринти, переглядає аналітику.

Актори мають виконувати наступні дії:

Студент:

UC1: Перегляд Kanban-дошки.

UC2: Створення / Редагування завдання (Task).

UC3: Зміна статусу завдання (To Do -> In Progress -> Done).

UC4: Логування витраченого часу (Time Tracking).

Скрам-майстер (Тімлід):

UC5: Створення та запуск Спринту (включає UC1).

UC6: Призначення виконавців на завдання.

UC7: Проведення ретроспективи (внесення підсумків у систему).

Викладач (Ментор):

UC8: Створення глобального Беклогу проєкту (User Stories).

UC9: Оцінювання результатів спринту (Sprint Review).

UC10: Перегляд графіків згоряння задач (Burndown Chart).

Діаграма послідовності (рис. 2.2) відображає взаємодію об'єктів у часі для конкретного сценарію. Розглянемо ключовий Agile-процес: «Переведення завдання у статус "Виконано" (Done) та автоматичне оновлення графіку спринту».

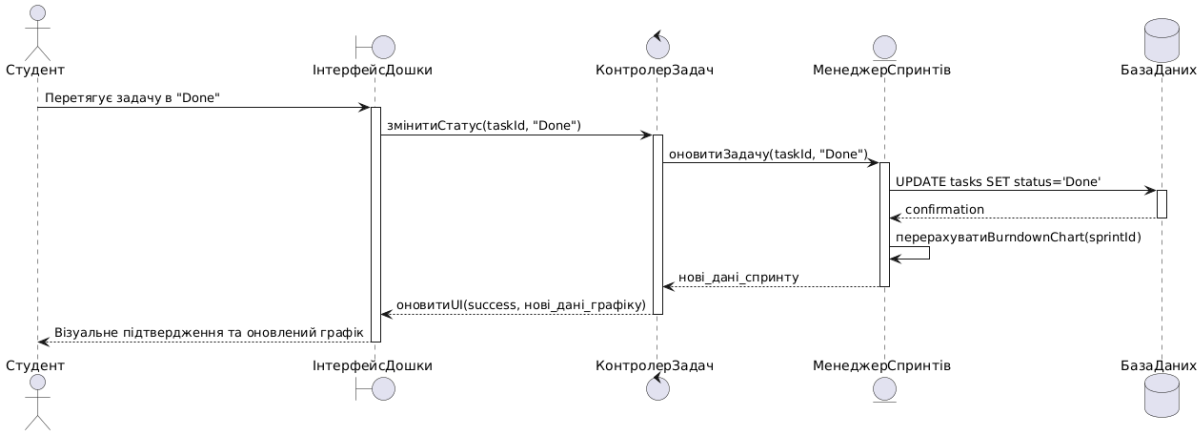

Рисунок 2.2 – Діаграма послідовності

Учасники (Lifelines):

Студент (Актор)

– ІнтерфейсДошки (UI Component)

– КонтролерЗадач (Backend Controller)

– МенеджерСпринтів (Agile Logic Service)

– БазаДаних (Database)

Сценарій взаємодії:

Послідовність дій у системі виглядає наступним чином:

Студент перетягує картку задачі в колонку "Done" в ІнтерфейсДошки.

ІнтерфейсДошки відправляє запит змінитиСтатус(taskId, "Done") до КонтролерЗадач.

КонтролерЗадач викликає метод оновитиЗадачу(taskId, status) у МенеджерСпринтів.

МенеджерСпринтів звертається до БазаДаних для збереження змін UPDATE tasks SET status='Done'.

БазаДаних повертає підтвердження успішно_збережено.

Після цього МенеджерСпринтів викликає внутрішній метод перерахуватиBurndownChart(sprintId).

МенеджерСпринтів надсилає оновлені дані графіку назад до КонтролерЗадач.

КонтролерЗадач передає ІнтерфейсДошки команду на візуальне оновлення показатиУспіх() + оновитиГрафік().

Студент бачить на екрані, що задача закрита, а лінія на графіку згоряння задач (Burndown) опустилася вниз.

2.3 Проєктування архітектури системи

Проєктування архітектури системи управління студентськими Agile-проєктами має враховувати дві критичні особливості: роботу в реальному часі (щоб рух карток на Kanban-дошці миттєво бачила вся команда) та гнучку аналітику (для оцінювання викладачем динаміки спринтів).

Для такої системи найкраще підходить архітектурний шаблон Модульного моноліту. Система розділена на 4 класичні рівні, що забезпечує незалежність інтерфейсу від бізнес-логіки та бази даних.

Клієнтський рівень:

– Single Page Application (SPA). Інтерактивний веб-додаток, який забезпечує швидкий відгук інтерфейсу (Drag-and-Drop для завдань, динамічні графіки).

– Модуль WebSockets Client. Підтримує постійне з'єднання з сервером для отримання миттєвих сповіщень про зміни на дошці.

– Рівень API Gateway та Безпеки

– REST API Endpoints. Точки входу для стандартних операцій (створити проєкт, додати студента).

– WebSocket Server. Менеджер активних сесій, який ретранслює події (наприклад, task_moved) усім учасникам конкретної команди.

– Авторизація (OAuth2 / JWT). Перевіряє токени та визначає права доступу на основі ролей (Студент, Тімлід, Викладач).

Рівень бізнес-логіки (Business Logic Layer) розділений на ізольовані сервіси (модулі):

– Сервіс користувачів та команд. Керує профілями студентів, академічними групами та розподілом по проєктних командах.

– Сервіс Agile-простору (Core). Відповідає за логіку беклогу, створення спринтів, керування тасками, дошками та коментарями.

– Сервіс трекінгу та аналітики. Обчислює витрачений час, швидкість команди (Velocity) та динамічно генерує графіки згоряння задач (Burndown Charts).

– Сервіс оцінювання. Дозволяє викладачу виставляти бали за спринти, залишати критерії оцінювання (Definition of Done) та фідбек.

Рівень даних (Data Layer):

– Реляційна БД. Зберігає структуровані дані з чіткими зв'язками (Команда → Спринт → Задача → Виконавець).

– In-Memory DB (Кеш). Використовується для швидкого зчитування активних WebSocket-сесій та кешування проміжних звітів, щоб не перевантажувати основну базу даних.

Технологічний стек, визначений на стадії проєктування, представлено в таблиці 2.4.

Таблиця 2.4 – Технологічний стек

Рівень	**Технології**	**Обґрунтування**
Frontend	React.js / Vue.js + TailwindCSS	Ідеально для побудови складних інтерактивних інтерфейсів (на кшталт дошки Trello/Jira).
Backend	Node.js (NestJS) або C# (.NET Core)	Мають високу продуктивність для асинхронних операцій та чудову вбудовану підтримку WebSockets.
База даних	PostgreSQL	Надійна open-source РСУБД для збереження складної структури сутностей проєкту.
Кеш / Real-time	Redis + Socket.io	Redis забезпечує кешування та координацію повідомлень, Socket.io — доставку подій на клієнт.
Аналітика	Chart.js / Recharts	Бібліотеки для візуалізації Burndown-графіків на стороні клієнта на основі JSON-даних від API.

2.4 Розробка інфологічної та даталогічної моделей бази даних

Розробка бази даних для системи управління студентськими Agile-проєктами складається з двох етапів: створення інфологічної (концептуальної) моделі, яка описує сутності та зв'язки між ними на рівні бізнес-логіки, та даталогічної (логічної/фізичної) моделі, яка визначає конкретну структуру таблиць, типи даних та обмеження для СКБД.

2.4.1 Інфологічна модель бази даних

Ця модель відображає реальні об'єкти предметної області (сутності) та те, як вони взаємодіють між собою.

Основні сутності та їх атрибути:

- Користувач (User): узагальнена сутність для всіх людей у системі.
    - Атрибути: ID, Ім'я, Прізвище, Email, Пароль, Глобальна роль (Студент / Викладач).
- Команда (Team): група студентів, яка працює над проєктами.
    - Атрибути: ID, Назва, Дата створення.
- Проєкт (Project): об'єкт розробки, який ведеться за методологією Agile.
    - Атрибути: ID, Назва, Опис, Дата початку, Дата дедлайну.
- Спринт (Sprint): фіксована за часом ітерація роботи над проєктом.

- Атрибути: ID, Назва, Дата початку, Дата завершення, Статус (Планується, Активний, Завершений).

- Задача (Task / User Story): атомарна одиниця роботи в межах проєкту чи спринту.
  - Атрибути: ID, Заголовок, Опис, Статус (To Do, In Progress, Review, Done), Пріоритет, Story Points (оцінка складності).
- Лог часу (WorkLog): запис про час, який студент витратив на виконання конкретної задачі.
  - Атрибути: ID, Витрачений час (у годинах), Коментар, Дата логування.
- Оцінка спринту (SprintReview): оцінка та фідбек від викладача (Product Owner) за результатами демо-презентації спринту.
  - Атрибути: ID, Бал, Текстовий фідбек, Дата оцінювання.

Зв'язки між сутностями (Кардинальність):

Користувач ↔ Команда (Багато-до-Багатьох). Студент може брати участь у кількох командах (наприклад, на різних курсах), а команда складається з багатьох студентів. Зв'язок реалізується через проміжну сутність з визначенням ролі всередині команди (Тімлід/Скрам-майстер, Розробник).

Команда ↔ Проєкт (Один-до-Багатьох). Команда може виконувати кілька проєктів, але один конкретний проєкт закріплений за однією командою.

Викладач (Користувач) ↔ Проєкт (Один-до-Багатьох). Один викладач виступає ментором (Product Owner) для багатьох проєктів.

Проєкт ↔ Спринт (Один-до-Багатьох). У межах одного проєкту створюється багато послідовних спринтів.

Проєкт ↔ Задача (Один-до-Багатьох). Усі задачі належать до загального беклогу конкретного проєкту.

Спринт ↔ Задача (Один-до-Багатьох). Спринт містить багато задач. Задача може не належати жодному спринту (тоді вона знаходиться в Беклозі проєкту).

Задача ↔ Виконавець (Користувач) (Багато-до-Одного). На одну задачу призначається один виконавець (студент), але один студент може виконувати багато задач.

Задача ↔ Лог часу (Один-до-Багатьох). До однієї задачі студенти можуть додавати багато записів про витрачений час.

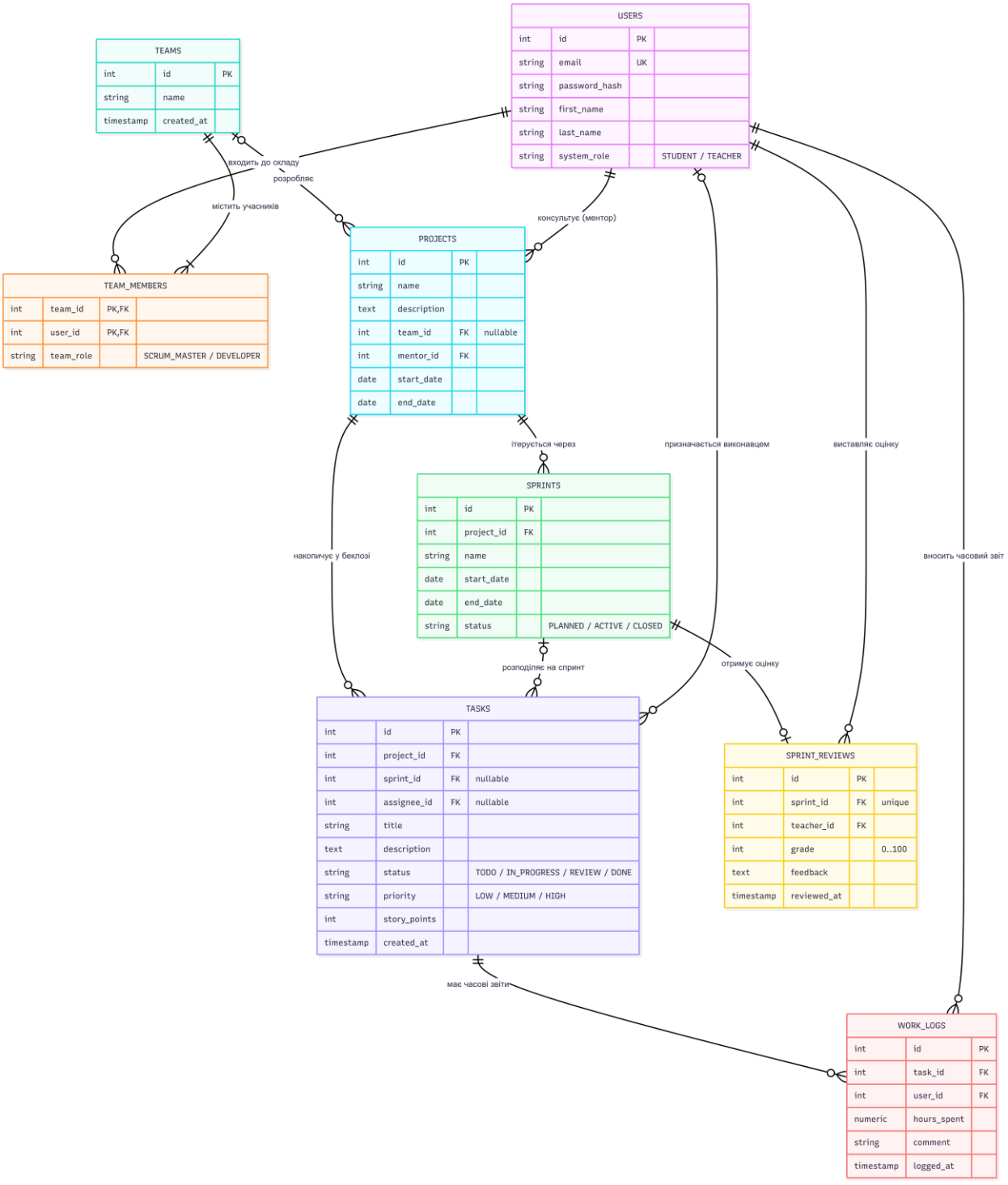

Рисунок 2.3 – ER-діаграма структури бази даних

2.4.2 Даталогічна модель

Переносимо концептуальну модель у реляційну структуру таблиць із зазначенням типів даних, первинних (PK) та зовнішніх (FK) ключів.

Таблиця users (Користувачі)

Назва поля	**Тип даних**	**Обмеження**	**Опис**
id	SERIAL	PK	Унікальний ID користувача
email	VARCHAR(150)	UNIQUE, NOT NULL	Електронна пошта (логін)
password_hash	VARCHAR(255)	NOT NULL	Хеш пароля
first_name	VARCHAR(50)	NOT NULL	Ім'я
last_name	VARCHAR(50)	NOT NULL	Прізвище
system_role	VARCHAR(20)	CHECK (IN ('STUDENT', 'TEACHER'))	Роль у системі

Таблиця teams (Команди)

Назва поля	**Тип даних**	**Обмеження**	**Опис**
id	SERIAL	PK	Унікальний ID команди
name	VARCHAR(100)	NOT NULL	Назва команди
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Дата створення

Таблиця team_members (Зв'язок студентів та команд)

Назва поля	**Тип даних**	**Обмеження**	**Опис**
team_id	INT	FK teams(id) ON DELETE CASCADE	ID команди
user_id	INT	FK users(id) ON DELETE CASCADE	ID студента
team_role	VARCHAR(30)	CHECK (IN ('SCRUM_MASTER', 'DEVELOPER'))	Роль всередині команди

Таблиця projects (Проєкти)

Назва поля	**Тип даних**	**Обмеження**	**Опис**
id	SERIAL	PK	Унікальний ID проєкту
name	VARCHAR(150)	NOT NULL	Назва проєкту

description	TEXT		Опис / Технічне завдання
team_id	INT	FK teams(id) ON DELETE SET NULL	Яка команда виконує
mentor_id	INT	FK users(id) ON DELETE RESTRICT	Викладач-ментор
start_date	DATE		Дата початку
end_date	DATE		Дедлайн проєкту

Таблиця sprints (Спринти)

Назва поля	**Тип даних**	**Обмеження**	**Опис**
id	SERIAL	PK	Унікальний ID спринту
project_id	INT	FK projects(id) ON DELETE CASCADE	До якого проєкту належить
name	VARCHAR(50)	NOT NULL	Назва (напр., "Спринт 1")
start_date	DATE	NOT NULL	Дата початку ітерації
end_date	DATE	NOT NULL	Дата завершення
status	VARCHAR(20)	CHECK (IN ('PLANNED', 'ACTIVE', 'CLOSED'))	Стан спринту

Таблиця tasks (Задачі / Беклог)

Назва поля	**Тип даних**	**Обмеження**	**Опис**
id	SERIAL	PK	Унікальний ID задачі
project_id	INT	FK projects(id) ON DELETE CASCADE	Глобальний беклог проєкту
sprint_id	INT	FK sprints(id) ON DELETE SET NULL, NULLABLE	Поточний спринт (NULL = в беклозі)
assignee_id	INT	FK users(id) ON DELETE SET NULL, NULLABLE	Хто виконує (студент)
title	VARCHAR(200)	NOT NULL	Назва задачі
description	TEXT		Критерії приймання (AC)
status	VARCHAR(20)	CHECK (IN ('TODO', 'IN_PROGRESS', 'REVIEW', 'DONE'))	Статус на дошці

priority	VARCHAR(15)	CHECK (IN ('LOW', 'MEDIUM', 'HIGH'))	Пріоритет
story_points	INT	CHECK (story_points >= 0)	Оцінка складності
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Дата створення

Таблиця work_logs (Тайм-трекінг)

Назва поля	**Тип даних**	**Обмеження**	**Опис**
id	SERIAL	PK	Унікальний ID логу
task_id	INT	FK tasks(id) ON DELETE CASCADE	За яка задачу
user_id	INT	FK users(id) ON DELETE CASCADE	Хто залогував час
hours_spent	NUMERIC(5,2)	CHECK (hours_spent > 0)	Кількість годин
comment	VARCHAR(255)		Що було зроблено
logged_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Коли внесено запис

Таблиця sprint_reviews (Оцінювання спринтів викладачем)

Назва поля	**Тип даних**	**Обмеження**	**Опис**
id	SERIAL	PK	Унікальний ID оцінки
sprint_id	INT	UNIQUE, FK sprints(id) ON DELETE CASCADE	Оцінка за конкретний спринт
teacher_id	INT	FK users(id)	Хто оцінив (викладач)
grade	INT	CHECK (grade BETWEEN 0 AND 100)	Оцінка в балах (100-бальна)
feedback	TEXT		Зауваження та рекомендації
reviewed_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Дата виставлення

2.5 Проєктування інтерфейсу користувача

Інтерфейс базується на двопанельній структурі (Split-screen):

- Ліва бічна панель (Sidebar). Забезпечує навігацію між проєктами, спринтами, беклогом, аналітикою та модулем оцінювання.
- Основна робоча область. Містить контентний блок, який динамічно змінюється залежно від обраного пункту.

Мапа навігації:

1. Dashboard (Головна панель) – загальний стан проєкту, поточний дедлайн, сповіщення.
2. Backlog (Беклог) – керування історіями користувачів (User Stories) та завданнями.
3. Board (Канбан-дошка) – візуалізація поточного спринту.
4. Analytics (Аналітика) – діаграми згоряння (Burndown), вклад кожного студента.
5. Grading Room (Кабінет оцінювання) – інтерфейс для викладача (Product Owner).

Детально опишемо ключові екрани UI.

Екран 1. Студентський Dashboard (Головна панель)

Цей екран є стартовим для студента. Його мета – дати швидкий фокус на завданнях поточного навчального тижня.

- Віджет «Мій спринт». Містить картки завдань, призначені безпосередньо на поточного користувача зі статусом та дедлайном.
- Метрика «Командний пульс». Містить візуальний індикатор (прогрес-бар) виконання цілей спринту.
- Блок «ШІ-Асистент» (AI-Partner). Містить вікно підказок, де штучний інтелект аналізує темп команди й видає рекомендації (наприклад: «Ваш спринт завершується за 2 дні, а 4 завдання досі в Peer Review. Рекомендую перевірити пул-реквести»).

Екран 2. Беклог та Планування Спринту

Інтерфейс має дворівневу структуру для полегшення декомпозиції: Лабораторна робота/Модуль → Історія → Завдання.

- Ліва колонка (Загальний беклог). Містить список усіх вимог до семестрового проєкту. Кожна картка має оцінку в Story Points та пріоритет.
- Права колонка (Поточний Спринт). Це область, куди студенти методом Drag-and-Drop перетягують картки з беклогу на поточний ітераційний цикл (зазвичай 1–2 тижні відповідно до навчального плану).
- Інтерактивна панель Definition of Done (DoD). Містить чек-лист, закріплений у верхній частині екрана, який нагадує, що вважається виконаним завданням в академічному контексті (наприклад: код завантажено в GitHub, тести пройдено, звіт оформлено).

Екран 3. Командна Канбан-дошка

Класична дошка, адаптована під студентські процеси.

To Do (Створити)	**In Progress (В роботі)**	**Peer Review (Рецензування)**	**Done (Завершено)**
Картка завдання (Scrum-роль: Розробник)	Картка завдання (Інтеграція з GitHub API - відображає гілку)	Картка завдання (Призначено на іншого студента для перевірки)	Картка із зеленою позначкою «Перевірено ШІ / Викладачем»

Особливість UI: На картках завдань відображаються аватарки виконавців та теги приналежності до технологічного стеку (Frontend, Backend, QA). При кліку на картку відкривається модальне вікно з детальним описом, коментарями команди та посиланням на комміт у репозиторії.

Екран 4. Панель Викладача та Аналітика

Найважливіший екран для моніторингу та виставлення балів за методологією Agile.

- Діаграма згоряння завдань (Burndown Chart). Показує ідеальну лінію згоряння Story Points та реальний графік команди. Одразу підсвічує «студентський синдром» (коли всі завдання закриваються в останню ніч спринту).

- Матриця контрибуції (Contribution Matrix). Інтерактивна кругова або стовпчаста діаграма, яка показує співвідношення виконаних Story Points, написаних рядків коду (через інтеграцію з Git) та активності на дошці для кожного члена команди.
- Модуль виставлення оцінок: Таблиця, де викладач може оцінити як командний результат (виконання мети спринту), так і індивідуальний внесок кожного студента з можливістю автоматичного експорту в журнал (LMS).

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Обґрунтування вибору технологічного стеку

Для реалізації фронтенду ми обрали фреймворк React.js. Для систем типу проєктного менеджменту критично важливою є концепція реактивного інтерфейсу, де зміна даних в одному модулі (наприклад, оцінка Story Points у беклозі) миттєво відображається в інших (наприклад, на аналітичному графіку викладача).

Ключові переваги для проєкту:

– Компонентно-орієнтована архітектура. Це дозволяє декомпонувати складний інтерфейс на ізольовані блоки багаторазового використання. Це значно спрощує підтримку та масштабування коду (наприклад, силами самих студентів у межах R&D практикумів).

– Ефективне керування станом (State Management). Завдяки механізму Virtual DOM та сучасним інструментам керування станом (нативні React Hooks: useContext, useReducer, або легковажний Zustand), система оновлює лише ті елементи інтерфейсу, які реально змінилися, мінімізуючи перемальовування сторінки (Reflow/Repaint).

– Екосистема та готове інструментарій. Наявність величезної кількості протестованих спільнотою бібліотек для специфічних Agile-завдань (робота з часом, drag-and-drop, рендеринг великих списків).

Для стилізації та UI-дизайн застосунку замість важких UI-фреймворків із фіксованим дизайном (як-от Bootstrap чи Material UI) або класичного написання CSS/SASS, ми обрали утилітарний (utility-first) підхід – Tailwind CSS.

Ключові переваги для проєкту:

– Швидкість прототипування та гнучкість. Стилі описуються безпосередньо в JSX-коді за допомогою атомарних класів. Це дозволяє миттєво адаптувати інтерфейс під специфічні потреби (наприклад, виділити картку завдання червоним бордером, якщо дедлайн менше ніж 2 години).

– Оптимізація розміру бандлу (Bundle Size). Завдяки вбудованому двигуну згортання (Just-in-Time compiler), Tailwind сканує проєкт та включає у фінальний CSS-файл лише ті класи, які реально використовуються. Це гарантує швидке завантаження системи навіть за умов нестабільного університетського Wi-Fi.

– Консистентність дизайну та адаптивність. Легко реалізується адаптивна верстка для планшетів/мобільних телефонів (корисно для студентів, які перевіряють статус завдань «на ходу») та підтримка темної теми (Dark Mode), що є стандартом для IT-спеціалістів.

Канбан-дошка та планувальник спринтів вимагають реалізації функціоналу перетягування (Drag-and-Drop). @hello-pangea/dnd є сучасним, стабільним та активно підтримуваним форком відомої бібліотеки react-beautiful-dnd від Atlassian (творців Jira).

Ключові переваги для проєкту:

– Спеціалізація на списках. На відміну від універсальних низькорівневих бібліотек (як react-dnd), цей інструмент створений саме для переміщення об'єктів між списками (Беклог → Спринт, To Do → In Progress).

Природна фізика та UX: Під час перетягування картки інші елементи плавно розсуваються, забезпечуючи чіткий візуальний відгук. Користувач чітко бачить, куди саме потрапить картка у момент опускання.

Доступність (Accessibility): Бібліотека «з коробки» підтримує керування з клавіатури та сумісна з екранними читанками (Screen Readers), що є важливою вимогою для забезпечення інклюзивності освіти.

Agile-методологія в освіті спирається на наочність результатів. Для побудови діаграм згоряння (Burndown Charts) та матриць контрибуції студентів потрібен надійний графічний рушій Recharts / Chart.js.

Ключові переваги для проєкту:

– Декларативний підхід Recharts. Бібліотека побудована виключно під React. Графіки збираються як звичайні компоненти:

– Продуктивність при рендерингу. Обидві бібліотеки оптимізовані для динамічного оновлення даних. Якщо студент закриває завдання на дошці, графік викладача перераховується і перемальовується без фризів інтерфейсу.

– Адаптивність та інтерактивність. Графіки масштабуються під розмір контейнера та підтримують інтерактивні підказки (Tooltips) при наведенні курсора на конкретну точку спринту.

Узагальнений опис технологій для фронтенд частини представлено в таблиці 3.1.

Таблиця 3.1 – Матриця відповідності вимог системи та технологій

Вимога до системи «EduAgile»	**Технологічне рішення**	**Переваги**
Висока динаміка інтерфейсу (оновлення статусів, ШІ-підказки в реальному часі)	**React.js (State & Hooks)**	Ефективна робота з віртуальним деревом елементів, швидке оновлення UI без перезавантажень.
Сучасний Dashboard-дизайн із підтримкою темної теми та адаптивності	**Tailwind CSS**	Відсутність зайвого CSS-коду, швидка зміна стилів через класи, просте впровадження Dark Mode.
Ергономічна Канбан-дошка з чітким візуальним відгуком	**@hello-pangea/dnd**	Нативна інтеграція з React, плавна анімація переміщення карток «із коробки», висока надійність.
Академічна аналітика (Burndown-діаграми, оцінка вкладу кожного студента)	**Recharts / Chart.js**	Декларативні SVG/Canvas компоненти, які легко пов'язати з реактивним станом системи.

Вибір PostgreSQL як центрального сховища даних є найкращим варіантом для систем академічного обліку та проєктного менеджменту через потребу в суворій консистентності даних.

Ключові переваги для проєкту:

– Надійна реляційна модель та ACID-відповідність. Студентські проєкти мають складну топологію зв'язків (один викладач → багато курсів → багато команд → багато спринтів → безліч завдань та підзавдань). PostgreSQL гарантує стовідсоткову цілісність даних (через Foreign Keys, Unique Constraints та каскадні операції), що виключає появу «загублених» завдань чи помилок у підрахунку оцінок.

Гібридний підхід з підтримкою JSONB: Модуль ШІ-асистента (AI-Partner) та динамічні чек-листи Definition of Done (DoD) вимагають збереження неструктурованих або напівструктурованих даних (наприклад, контекст промптів, історія порад ШІ, специфічні критерії готовності для окремих завдань). Потужна підтримка типу JSONB у PostgreSQL дозволяє поєднувати швидкість реляційних запитів із гнучкістю NoSQL-рішень, забезпечуючи індексацію всередині JSON-документів.

Аналітичні можливості (Window Functions): Для побудови Burndown-діаграм та матриць контрибуції студентів бекенд має виконувати складні агрегаційні обчислення. Віконні функції PostgreSQL дозволяють ефективно вираховувати кумулятивні суми закритих Story Points безпосередньо на рівні бази даних, мінімізуючи навантаження на процесор сервера додатків.

Для забезпечення архітектурної єдності з фронтендом (React/TypeScript) серверна частина проєкту будується на базі Node.js із використанням прогресивного фреймворку NestJS.

Ключові переваги для проєкту:

– Єдиний технологічний простір (Full-stack TypeScript). Використання однієї мови на фронтенді та бекенді дозволяє шерити типи даних, інтерфейси (DTO) та валідаційні схеми між клієнтом і сервером. Це в рази зменшує кількість помилок при інтеграції API та знижує когнітивне навантаження на студентів, які можуть працювати над проєктом як Full-stack розробники.

– Модульна архітектура NestJS. Фреймворк змушує розробників дотримуватися чіткої структури (модулі, контролери, сервіси) та підтримує концепцію Dependency Injection (впровадження залежностей). Це ідеально підходить для академічного середовища, оскільки демонструє студентам найкращі практики інженерії програмного забезпечення (SOLID, Clean Architecture).

– Асинхронність та Non-blocking I/O. Node.js оптимізований для великої кількості легковажних запитів (I/O-bound operations), що критично під час

масового одночасного використання системи студентами під час занять або захисту лабораторних робіт.

Замість написання сирих SQL-запитів або використання застарілих ORM, пропонується інтегрувати Prisma ORM.

Ключові переваги для проєкту:

– Повна типобезпека (Type-safety). Prisma автоматично генерує клієнт бази даних на основі схеми PostgreSQL. Будь-яка помилка в назві поля чи невідповідність типу даних підсвічується ще на етапі написання коду, а не в процесі виконання (Runtime).

– Декларативні міграції. Керування структурою бази даних відбувається через простий файл schema.prisma. Студенти та викладачі можуть легко розгортати локальні копії бази даних однією командою (prisma migrate dev), що мінімізує проблеми з налаштуванням середовища розробки.

Agile-методологія передбачає командну роботу. Якщо один студент перетягує картку на Канбан-дошці з "In Progress" в "Peer Review", решта членів команди мають миттєво побачити цю зміну без перезавантаження сторінки.

Бекенд на NestJS містить вбудовані шлюзи (Gateways) на базі Socket.io, які забезпечують двосторонній зв’язок із низькою затримкою.

Будь-яка мутація даних у PostgreSQL (через REST API або WebSocket-подію) ініціює широкомовне сповіщення (broadcast) для всіх підключених учасників конкретної команди.

Узагальнення технологічних рішень для бекенд частини представлено в таблиці 3.2.

Таблиця 3.2 – Матриця відповідності бекенд-технологій вимогам системи

Вимога до бекенду «EduAgile»	**Технологічне рішення**	**Архітектурна роль**
Суворе збереження структури проєктів, зв'язків «Студент-Команда-Оцінка»	PostgreSQL (Relational)	Гарантія цілісності даних (ACID), каскадні обмеження, складні вибірки.

Збереження гнучких даних (контекст ШІ, кастомні критерії DoD)	PostgreSQL (JSONB)	Ефективне збереження документарних даних без створення десятків додаткових таблиць.
Типобезпека end-to-end від клієнта до бази даних	TypeScript + NestJS + Prisma	Усунення помилок невідповідності типів на всіх етапах проходження даних.
Миттєва синхронізація стану Канбан-дошки між студентами	WebSockets (Socket.io)	Трансляція подій зміни статусів завдань у реальному часі.
Аналітика швидкості команди (Velocity, Burndown розрахунки)	PostgreSQL Window Functions	Проведення складних математичних розрахунків трендів безпосередньо в СУБД.

3.2 Реалізація серверної частини

Для проєкту обрано архітектуру модульного моноліту. Вона ідеально підходить для NestJS, оскільки дозволяє тримати весь код в одному репозиторії (зручно для студентських проєктів), але логічно розділяє його на незалежні домени (Modules).

Основні модулі системи:

– AuthModule, авторизація, генерація JWT-токенів, відновлення паролів.

– UsersModule, керування профілями (Студент, Викладач, Адміністратор).

– ProjectsModule, CRUD для навчальних проєктів, команд та беклогу.

– SprintsModule, логіка ітерацій, розрахунок Burndown-метрик.

– TasksModule, керування завданнями (Story Points, статуси, виконавці).

– EventsModule (WebSockets), шлюз для обміну повідомленнями в реальному часі.

– AIModule, сервіс для взаємодії з LLM (ШІ-асистент).

Моделювання бази даних (Prisma Schema)

Prisma виступає як єдине джерело істини для структури бази даних. Нижче наведено спрощений приклад того, як виглядатиме ядро бази даних у файлі schema.prisma:

Фрагмент коду:

```
// Enum для ролей та статусів
enum Role {
```

```
  STUDENT
  TEACHER
}

enum TaskStatus {
  TODO
  IN_PROGRESS
  PEER_REVIEW
  DONE
}

model User {
  id        String   @id @default(uuid())
  email     String   @unique
  role      Role     @default(STUDENT)
  tasks       Task[]    // Зв'язок: завдання, призначені на
користувача
  projects  Team[]   // Зв'язок: команди, у яких бере участь
}

model Project {
  id          String   @id @default(uuid())
  title       String
  teacherId   String
  backlog     Task[]   // Завдання, що ще не в спринті
  sprints     Sprint[]
}

model Sprint {
  id          String   @id @default(uuid())
  projectId   String
  startDate   DateTime
  endDate     DateTime
  tasks       Task[]
}

model Task {
  id          String     @id @default(uuid())
  title       String
  status      TaskStatus @default(TODO)
  storyPoints Int
  assigneeId  String?
  sprintId    String?
  aiContext   Json?      // JSONB поле для збереження порад
ШІ та DoD
}
```

Реалізація реального часу (WebSockets / Kanban Sync)

Коли студент перетягує картку на Канбан-дошці (через @hello-pangea/dnd на фронтенді), бекенд має не лише зберегти це в БД, а й миттєво оновити екрани інших учасників команди.

Потік виконання (Flow):

Клієнт → Сервер (REST): Фронтенд відправляє PATCH /tasks/:id/status з новим статусом (IN_PROGRESS).

БД (Prisma): TasksService оновлює статус завдання в PostgreSQL.

Сервер → Клієнти (WebSocket): NestJS інжектує EventsGateway у контролер завдань. Після успішного запису в БД, шлюз робить трансляцію (broadcast) події task_updated у кімнату (Room), яка відповідає ID поточного проєкту/команди.

Клієнт (React): Слухає подію task_updated і точково оновлює стан компонента дошки без повного перезавантаження.

Інтеграція ШІ-Асистента (AI-Partner)

Модуль штучного інтелекту працює як бекграунд-сервіс, аналізуючи стан проєкту для надання рекомендацій студентам та аналітики викладачу.

Генерація контексту: AIModule періодично (або за запитом) збирає дані через Prisma (відсоток виконаних завдань, кількість днів до кінця спринту, хто зі студентів давно не робив комітів).

Запит до LLM: Сформований JSON-контекст перетворюється на структурований системний промпт і відправляється через API (наприклад, OpenAI API або Google Gemini).

Кешування відповідей: Щоб не витрачати токени на однакові запити, відповіді ШІ зберігаються у полі aiContext (тип JSONB) конкретного спринту чи завдання в PostgreSQL.

Безпека та Авторизація (Guards & JWT)

Оскільки система містить оцінки, архітектура безпеки має бути безкомпромісною:

Для аутентифікації використовується стратегія Passport-JWT. При логіні користувач отримує Access Token (короткостроковий, для API) та Refresh Token (довгостроковий, зберігається в HttpOnly cookie для захисту від XSS).

Для авторизація (RBAC - Role-Based Access Control) у NestJS створюються кастомні Guards (Охоронці). Наприклад, Ендпоінт POST /projects/grades захищений @Roles(Role.TEACHER). Студент фізично не зможе надіслати запит на зміну оцінки. Ендпоінт PATCH /tasks/:id захищений TeamMemberGuard, який перевіряє, чи належить студент до команди, якій належить це завдання.

3.3 Розробка клієнтської частини

Реалізація клієнтської частини має поєднувати високу продуктивність сучасного Single Page Application (SPA), інтуїтивно зрозумілий інтерфейс для студентів та потужні аналітичні інструменти для викладача. Оскільки фронтенд будується на зв'язці React та Tailwind CSS, архітектура має бути максимально модульною.

Опишемо технічну реалізацію ключових шарів клієнтського застосунку.

Архітектура та організація коду: Feature-Sliced Design (FSD)

Методологія Feature-Sliced Design дозволила нам розділити код на логічні шари:

app/: Ініціалізація застосунку, глобальні стилі, провайдери контексту та налаштування маршрутизації (React Router).

pages/: Компоненти-сторінки (наприклад, DashboardPage, BoardPage, AnalyticsPage). Вони лише компонують віджети та фічі.

widgets/: Самостійні блоки інтерфейсу (наприклад, KanbanBoard, TeacherGradingTable).

features/: Логіка взаємодії користувача (наприклад, MoveTask, AssignStudent, GenerateAIAdvice).

entities/: Бізнес-сутності проєкту (моделі User, Task, Sprint, Project).

shared/: Перевикористовувані UI-компоненти (кнопки, інпути на базі Tailwind), утиліти, конфігурація API.

Управління станом (State Management)

Для уникнення проблеми "Prop Drilling" (передачі пропсів через безліч рівнів компонентів) і забезпечення швидкого відгуку інтерфейсу, стан розділяється на дві категорії.

Серверний стан. Усі запити до NestJS бекенду обробляються через React Query. Список завдань або беклог кешується "з коробки". При переході між вкладками дані не завантажуються повторно, інтерфейс реагує миттєво. Оптимістичні оновлення. Коли студент перетягує картку з To Do в In Progress, React Query миттєво оновлює UI, не чекаючи відповіді від сервера. Якщо бекенд повертає помилку, картка автоматично повертається на попереднє місце.

Клієнтський стан. Бібліотека Zustand використовується для глобальних налаштувань UI, які не залежать від бази даних. Стан бокової панелі (згорнута/розгорнута). Обрана тема (Dark/Light). Стан глобальних модальних вікон (наприклад, вікно перегляду профілю студента).

Інтеграція реального часу (WebSockets на клієнті)

Спільна робота команд вимагає миттєвої синхронізації екранів. На клієнті це реалізується через кастомні React-хуки та бібліотеку socket.io-client. При монтуванні компонента BoardPage клієнт підключається до WebSocket-кімнати поточного проєкту. Слухає події від бекенду (наприклад, TASK_MOVED або NEW_COMMENT). При отриманні події від іншого користувача, хук інвалідує (скидає) відповідний кеш у React Query або точково оновлює масив завдань. Завдяки цьому студенти бачать, як картки переміщуються "самі по собі", як у Google Docs чи Miro.

Концепція "AI-as-a-Partner" в Інтерфейсі

Штучний інтелект у системі виступає не як окремий чат-бот, а як глибоко інтегрований у робочий процес напарник (подібно до інструментів парного програмування). Коли студент пише User Story, ШІ-асистент у

реальному часі аналізує текст і пропонує критерії приймання (Definition of Done) у вигляді чек-листів, які можна додати одним кліком. У модальному вікні картки завдання AI-Partner може підсвічувати вузькі місця у прикріплених посиланнях на GitHub (наприклад, недоліки в алгоритмічних рішеннях або архітектурі бази даних). Наприкінці спринту модуль ШІ генерує коротке резюме для команди прямо на Dashboard: які типи завдань забирали найбільше часу, та дає рекомендації на наступну ітерацію.

UI-компоненти та Аналітичні Дашборди

Для швидкої розробки доступного інтерфейсу використовується комбінація Tailwind CSS та Headless UI (наприклад, Radix UI), що дає повний контроль над стилями без нав'язування чужого дизайну. Kanban Board побудована на @hello-pangea/dnd. Картки підтримують кастомні маркери кольору (наприклад, червоний бордер для завдань, що "горять"). Teacher Analytics використовує бібліотеку Recharts. Будуються інтерактивні Burndown-діаграми. При кліку на точку графіка (конкретний день спринту) викладач бачить модальне вікно з деталізацією: хто саме зі студентів закрив завдання в цей день, що дозволяє швидко оцінити ритмічність роботи та індивідуальний внесок.

3.4 Огляд розробленої системи

Оглянемо детально режими роботи навчальної платформи для управління студентськими проєктами за методологією Agile.

Інтерфейс системи (рис. 3.1) складається з трьох основних частин:

– ліве бічне меню;

– верхня панель;

– головна робоча область.

Ліве бічне меню використовується для навігації між основними екранами системи. Тут також знаходиться кнопка "Налаштування".

Верхня панель відображає назву поточного розділу, залишок часу до кінця спринту та містить важливий перемикач ролей ("Студент" / "Викладач").

Головна робоча область змінюється залежно від обраного пункту меню.

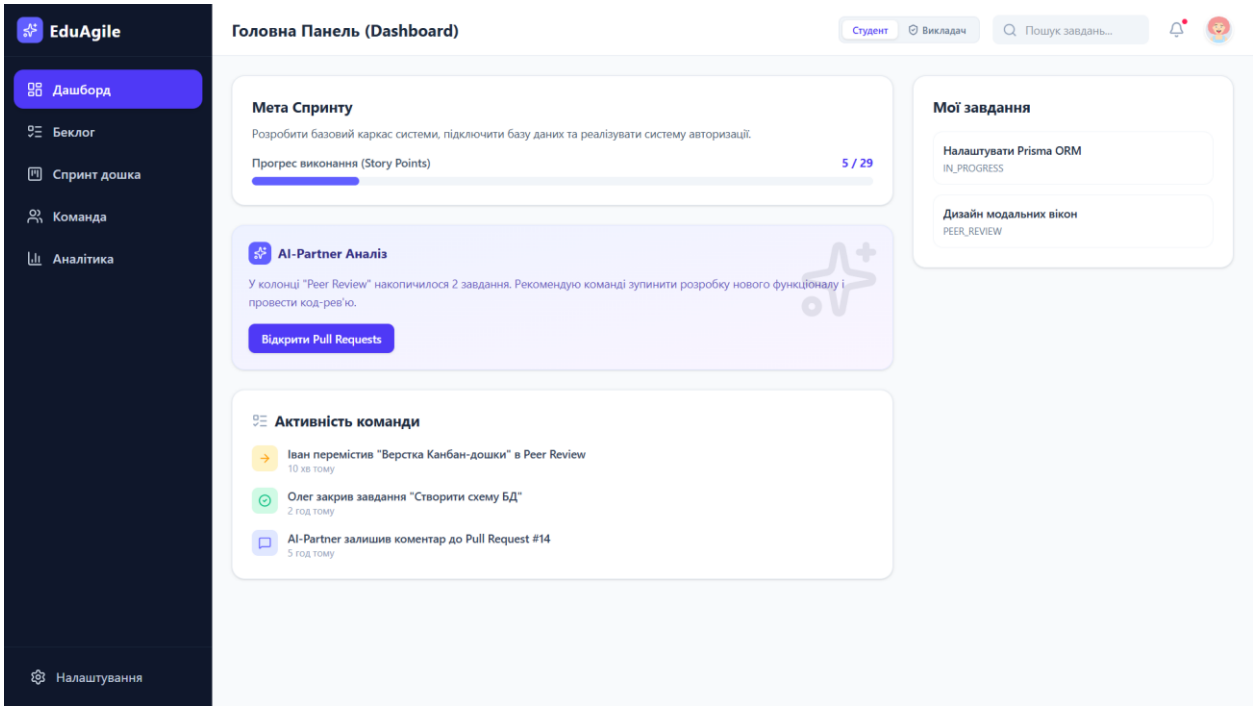

Рисунок 3.1 – Загальний вигляд застосунку

У правому верхньому куті ви можете змінити свою роль. Це впливає на рівень доступу:

– студент;

– викладач;

Студент має базовий доступ, може керувати завданнями, переглядати команду та загальну аналітику. Викладач отримує адміністративний доступ. Має права на формування команди, зміну ролей учасників та доступ до детальної матриці оцінювання кожного студента.

На панелі Дашборд відображається стартовий екран, який дає зріз стану проєкту на дану хвилину. На цій панелі визначається мета спринту, загальний опис того, що потрібно зробити за поточну ітерацію, та прогрес-бар виконання завдань.

В застосунок інтегровано модуль ШІ-помічника: AI-Partner Аналіз. Це блок штучного інтелекту, який аналізує дошку та дає поради (наприклад, якщо накопичилося багато неперевірених завдань).

На головній сторінці ми також можемо бачити стрічку останніх подій (хто яке завдання закрив або перемістив) та швидкий список карток, призначених особисто на вас.

На цій сторінці користувач має виконати наступні дії:

– Читання порад від AI-асистента для коригування роботи команди.

– Швидкий перехід до своїх завдань (натискання на картку у списку праворуч).

Головною робочою областю системи є Спринт дошка (рис. 3.2). Це класична Канбан-дошка з чотирма колонками: До виконання (To Do), В роботі (In Progress), Перевірка (Peer Review), Готово (Done). На картках завдань видно їхню назву, виконавця (аватар), теги та вагу (Story Points).

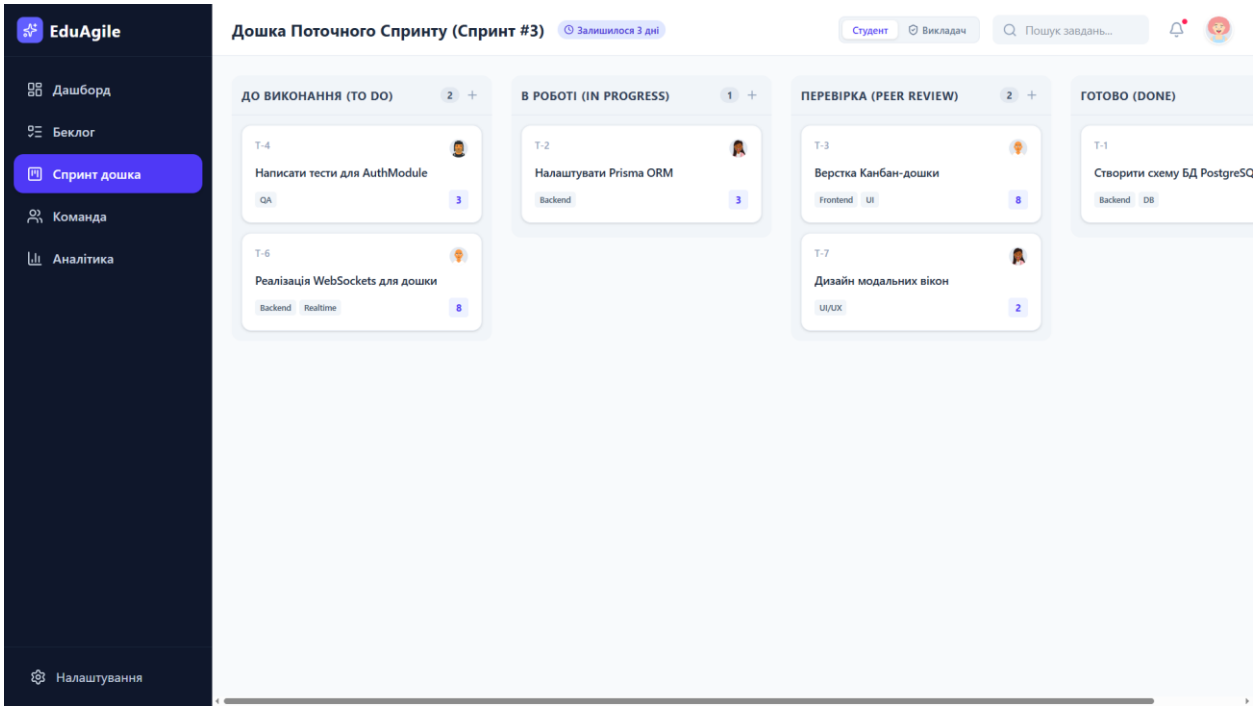

Рисунок 3.2 – Спринт дошка

Користувачу в цьому режимі доступні наступні дії:

– Переміщення завдань (Drag & Drop). Достатньо затиснiти ліву кнопку миші на картці завдання та перетягніть її у потрібну колонку.

– Швидке переміщення. Можна наведіть курсор на картку, у правому верхньому куті з'являться кнопки зі стрілками або галочкою для швидкого перекидання картки на наступний етап.

– Швидке створення завдання. Натисніть іконку + біля назви будь-якої колонки, щоб створити нове завдання, яке одразу з'явиться саме в цій колонці (рис. 3.3).

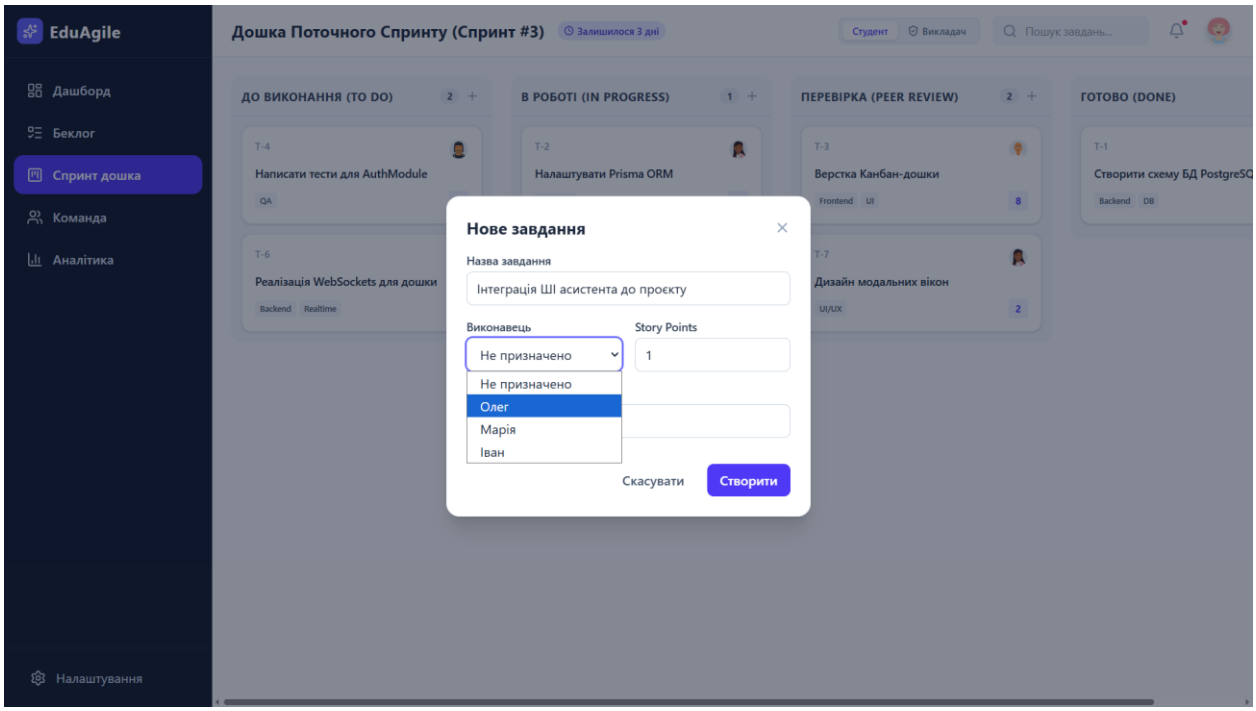

Рисунок 3.3 – Створення нового завдання

На панелі «Команда» (рис. 3.4) відображаються картки всіх учасників проєкту з їхніми іменами, спеціалізацією (наприклад, Backend Розробник) та Agile-роллю (Scrum Master, QA Engineer тощо).

Користувачі можуть виконати на панелі такі дії:

– Перегляд. Студент может лише бачити, хто є в команді та які ролі вони виконують.

– Управління. Викладач може додати учасника, щоб включити нового студента в проєкт.

Також викладач може використовувати випадаючий список на картці студента, щоб змінити його Agile-роль. Або навести курсор на картку та натиснути іконку кошика у верхньому правому куті, щоб видалити учасника.

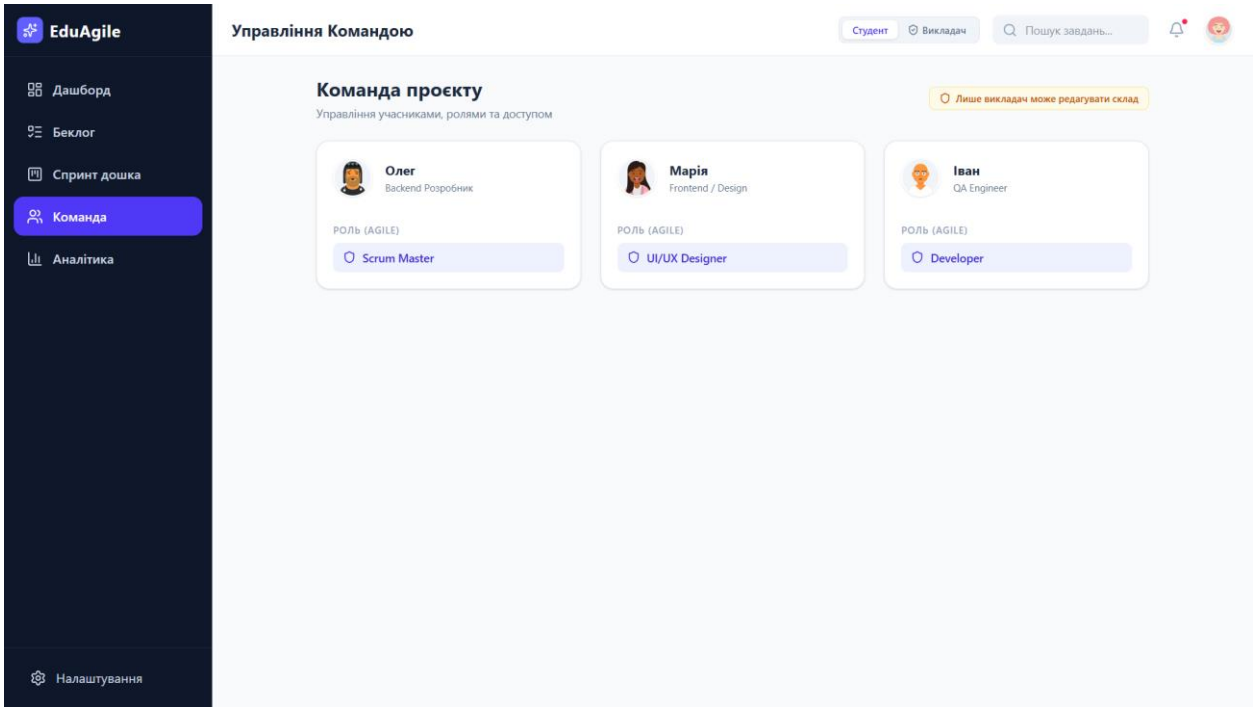

Рисунок 3.4 – Панель «Команда»

Панель «Аналітика» (рис. 3.5) використовується для відстеження ритму роботи команди та індивідуального внеску.

Burndown Діаграма видима всім. На ній відображається графік, який показує ідеальне згоряння завдань (сірим) у порівнянні з реальним темпом команди (фіолетовим). Допомагає виявити "студентський синдром", коли все робиться в останній день.

Матриця контрибуції видима лише Викладачу. Це таблиця з детальними показниками кожного студента.

Користувач на цій сторінці може виконати наступні дії:

– Студент виконує аналіз Burndown-діаграми для розуміння, чи встигає команда закрити спринт вчасно. Наведення курсору на стовпчики показує деталі плану та факту по днях.

– Викладач додатково аналізує таблицю контрибуції. Система автоматично підраховує закриті Story Points, кількість комітів та видає ШІ-оцінку (наприклад, підсвічує червоним "Free-rider" тих, хто не бере участі в розробці).

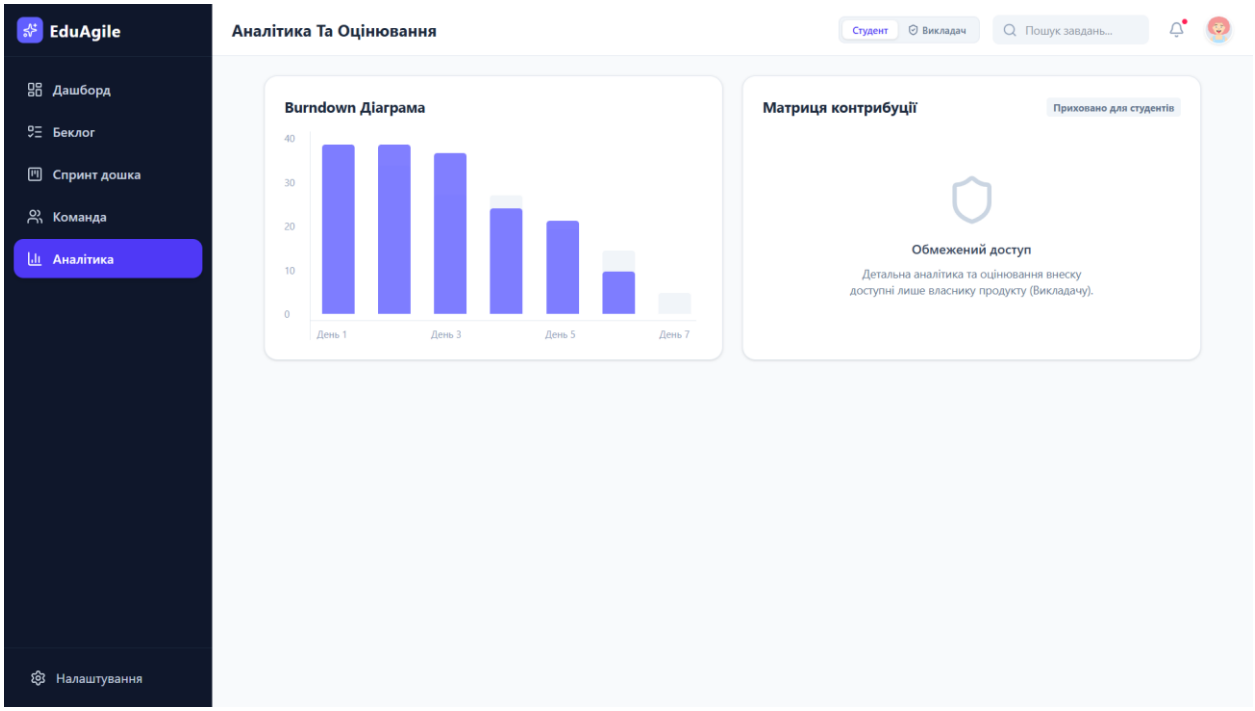

Рисунок 3.5 – Панель «Аналітика»

Також студентам та викладач доступний Беклог. Тут відображається: Список усіх ідей, вимог та завдань проєкту, які ще не взяті в роботу (не додані у поточний спринт).

За створення історії відповідає викладач. Він може натисніть кнопку + Додати історію у верхньому правому куті блоку (рис. 3.6). Для створення історії заповніть назву, призначте виконавця (опціонально), вкажіть складність (Story Points) та додайте теги.

Також цей запис можна перемістити у спринт. Для цього наведіть курсор на завдання у списку та натисніть кнопку У спринт →, яка з'явиться праворуч. Завдання автоматично переміститься на Канбан-дошку в колонку "До виконання".

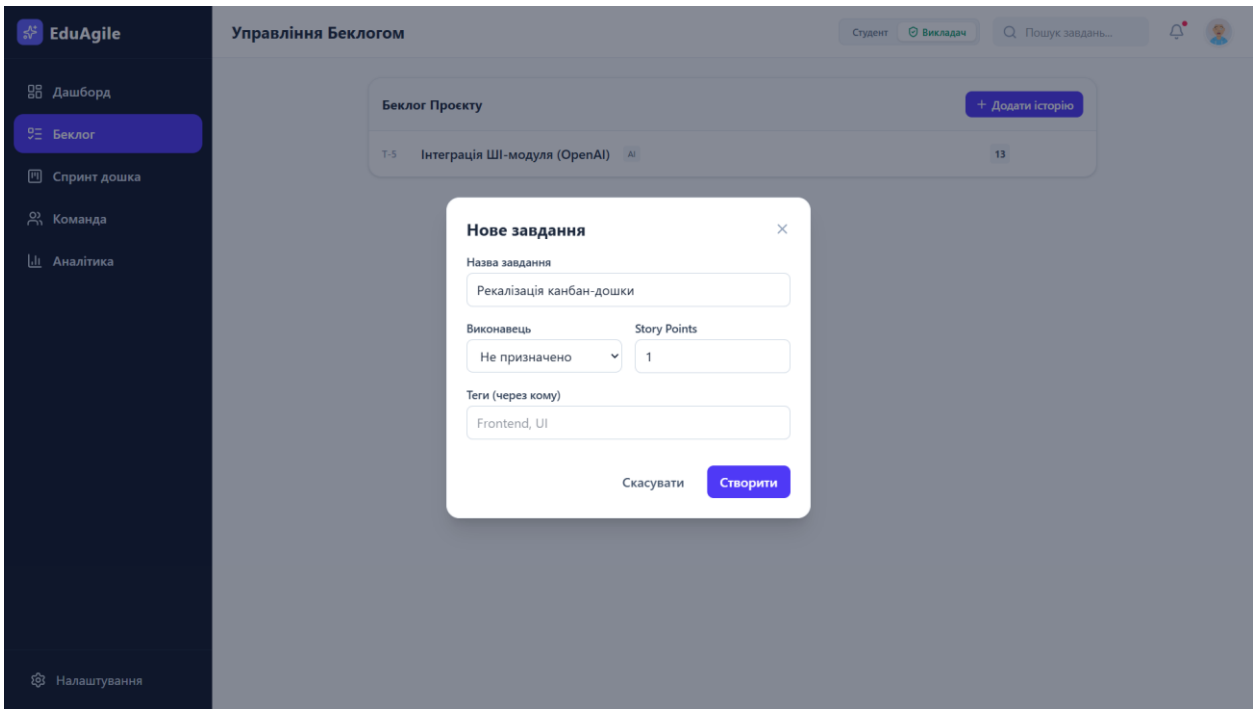

Рисунок 3.6 – Ведення беклогу

У викладача також є можливість редагувати команду (рис. 3.7), додаючи або видаляючи учасників та змінюючи їх ролі.

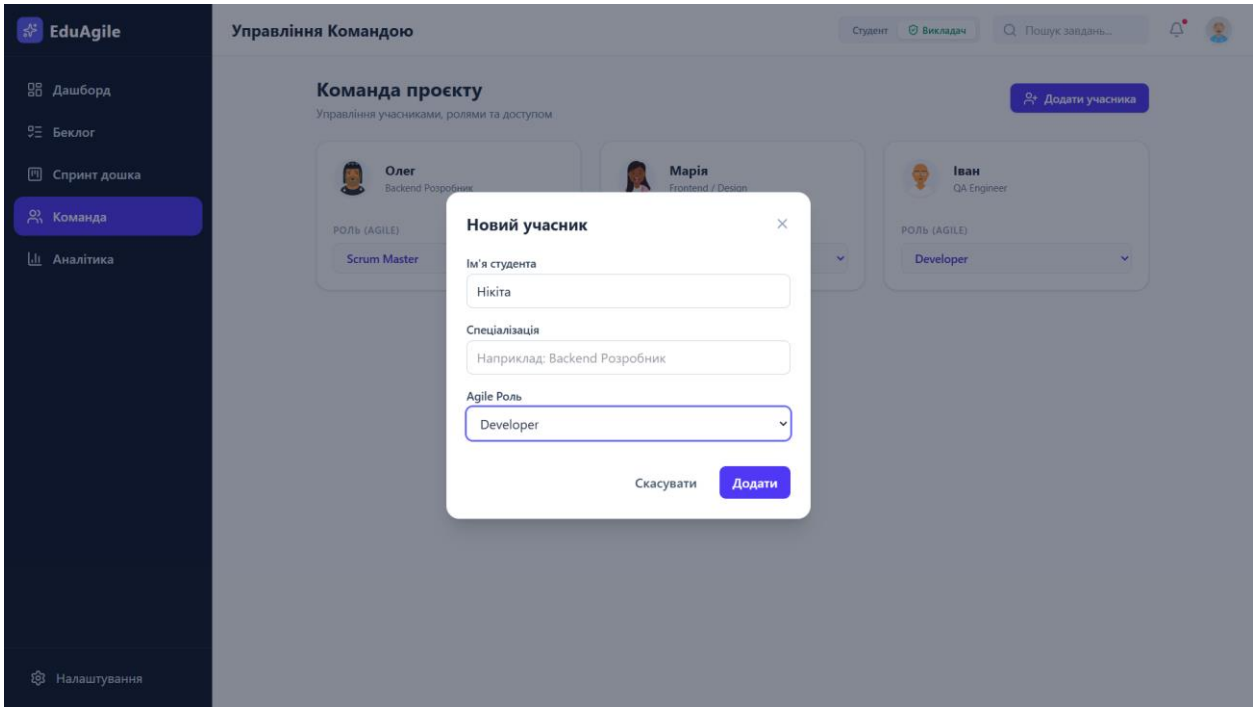

Рисунок 3.7 – Додавання нового учасника в команду

Також викладач має додаткові інструменти аналізу на панелі «Аналітика».

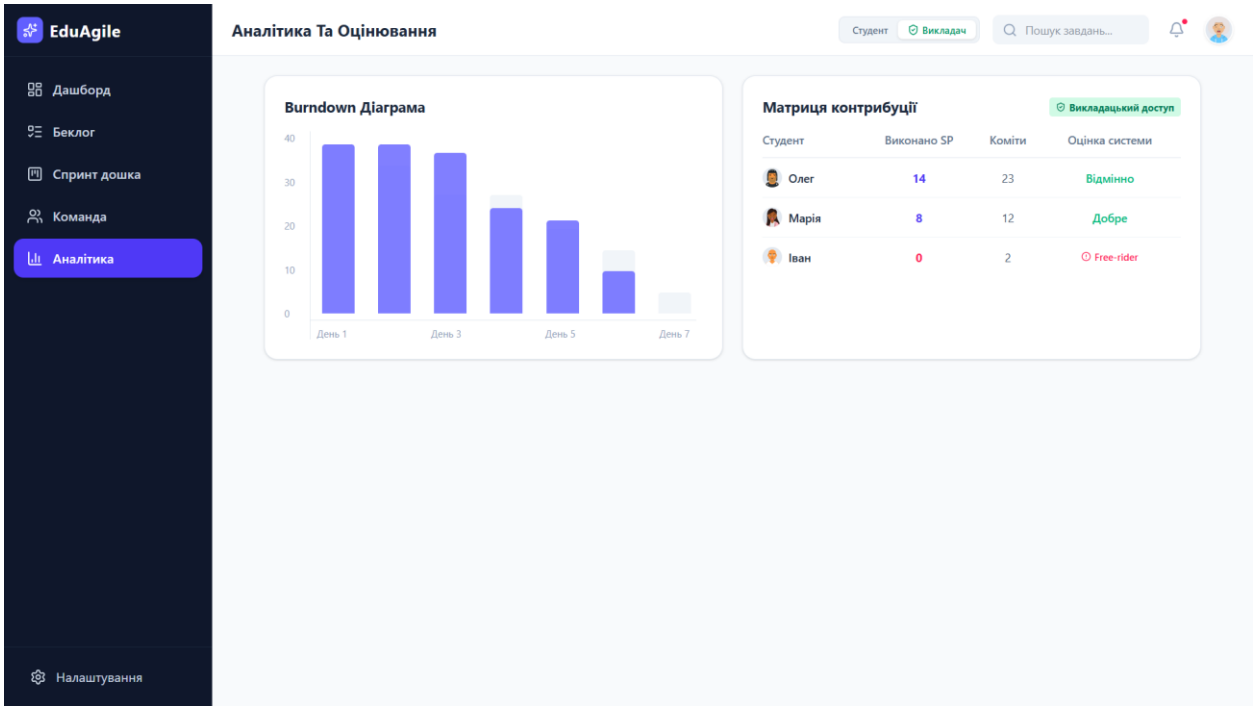

Рисунок 3.8 – Панель Аналітика для викладача

У самому низу лівого бічного меню знаходиться кнопка "Налаштування" (рис 3.9).

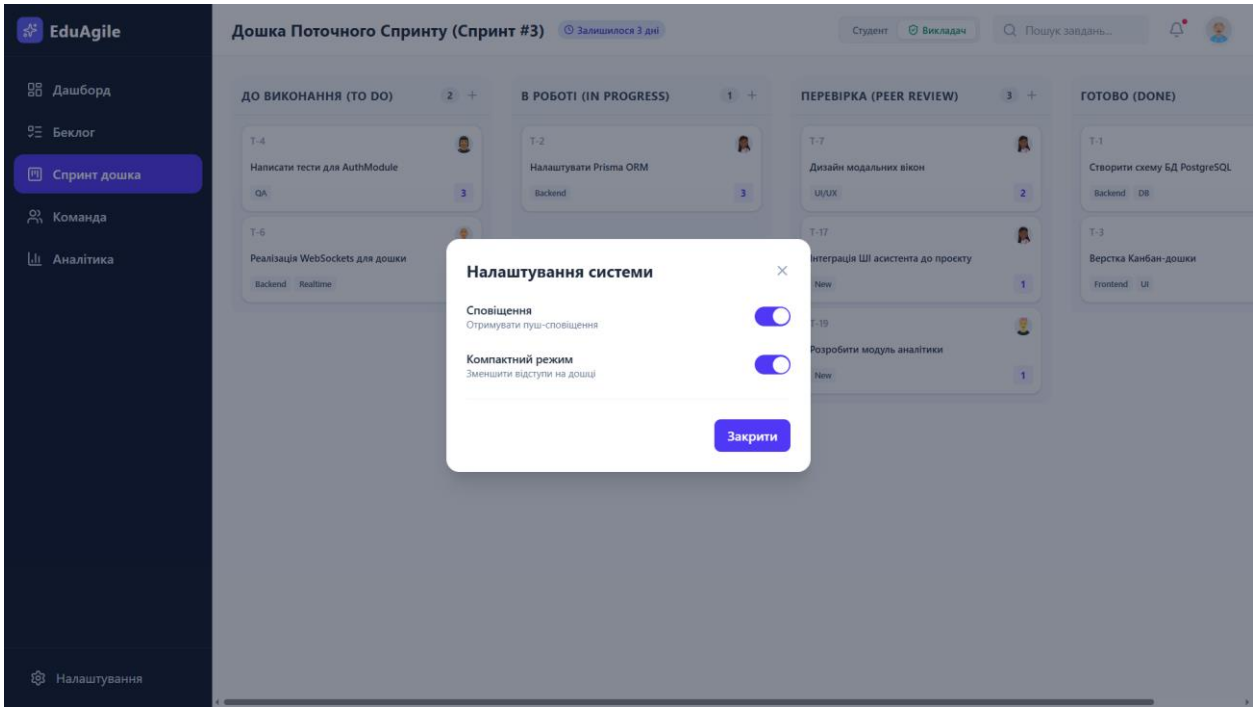

Рисунок 3.9 – Налаштування системи

В налаштуваннях ви можете:

– Увімкнути або вимкнути системні сповіщення.

– Увімкнути "Компактний режим".

Компактний режим зменшує відступи на Канбан-дошці, що дозволяє вмістити більше карток завдань на одному екрані, що особливо зручно для невеликих моніторів або ноутбуків.

ВИСНОВКИ

У ході виконання проєкту було спроєктовано та програмно реалізовано клієнтську частину інтерактивної веб-системи управління студентськими Agile-проєктами «EduAgile». Розробка базується на сучасному технологічному стеку: екосистемі React (із використанням Vite як збирача) та фреймворку Tailwind CSS v4 для утилітарної стилізації.

Вдалося реалізувати повноцінний Feature-Sliced дизайн інтерфейсу, що включає:

– інтерактивну Канбан-дошку з підтримкою Drag-and-Drop операцій для управління станами завдань поточного спринту.

– модуль управління беклогом та командуванням, що дозволяє формувати склад виконавців та призначати Agile-ролі.

– дворольову модель доступу (Студент / Викладач), яка динамічно змінює інтерфейс та обмежує права на модифікацію команди чи перегляд чужих оцінок.

– інноваційний модуль «AI-as-a-Partner», який у режимі реального часу аналізує поточний стан дошки (WIP-ліміти, накопичення завдань на рев'ю) та генерує контекстні підказки.

– аналітичну панель (Dashboard) з імітацією розрахунку Burndown-метрик та матрицею індивідуальної контрибуції, яка ідентифікує ризики, зокрема ефект «безбілетника» (free-rider) у командній роботі. Також було успішно вирішено проблеми конфігурації оточення розробки, що забезпечує легкий локальний запуск проєкту.

Практична цінність «EduAgile» полягає у розв'язанні специфічних проблем академічного середовища під час викладання IT-дисциплін (зокрема, інженерії програмного забезпечення та системного аналізу).

Система надає викладачеві (у ролі Product Owner) інструментарій для об'єктивного оцінювання роботи в групах. Замість єдиної оцінки за проєкт,

викладач бачить індивідуальний внесок кожного студента через закриті Story Points та рівень активності.

Інтерфейс виступає наочним демонстратором методологій Scrum та Kanban. Студенти на практиці засвоюють принципи декомпозиції завдань, планування ітерацій та дотримання Definition of Done.

Інтеграція штучного інтелекту як повноцінного учасника команди (AI-as-a-Partner) формує у студентів навички конструктивної взаємодії з LLM, що є критично важливою компетенцією для сучасних розробників.

Сам вихідний код проєкту, написаний на React та Tailwind CSS, може слугувати навчальним матеріалом для курсів із веброзробки або проєктування інтерфейсів.

Впроваджено серверну частину на базі Node.js/NestJS із використанням PostgreSQL для збереження даних та WebSockets (Socket.io) для миттєвої синхронізації дій на Канбан-дошці між учасниками під час парного чи групового програмування.

В подальшому можливе розширення можливостей ШІ-асистента для автоматичного рев'ю коду через інтеграцію з GitHub/GitLab API, а також генерації тестів до завдань, що закриваються.

Потенційно можлива інтеграція системи з академічним середовищем. Наприклад, можна створити API-шлюзів для експорту зібраних метрик та оцінок до університетських систем управління навчанням (LMS, наприклад, Moodle) чи систем рейтингування.

Перпективним напрямом подальшого розвитку системи є додавання елементів алгоритмічних челенджів або системи досягнень на основі успішно виконаних спринтів для підвищення мотивації студентів.

ПЕРЕЛІК ПОСИЛАНЬ

1. 6 Best Project Management Tools in 2026. Medium. 2026. URL: https://medium.com/@kanerika/6-best-project-management-tools-in-2026-790ad20c59eb
2. Agile Method Development: A Live-Lab Case Study on Product Properties for Process Planning / A. Albers et al. Design Society: website. 2017. URL: https://www.designsociety.org/download-publication/40486/agile_method_development_a_live-lab_case_study_on_product_properties_for_process_planning
3. Agile Methodology in a University Setting. HE Professional. 2025. URL: https://heprofessional.co.uk/agile-methodology-university
4. Agile Project Management with Jira. Atlassian. URL: https://www.atlassian.com/agile/project-management
5. Al-Riyami A. Analysis of computer science education challenges and software engineering pedagogy using agile principles. ResearchGate. URL: https://www.researchgate.net/publication/336090204_Using_Agile_Practice_for_Student_Software_Projects
6. Application of the eduScrum methodology to a higher education institution in the Amazon / J. A. Ferreira et al. Repositorium.uminho.pt: institutional repository. URL: https://repositorium.uminho.pt/bitstreams/5fbb2104-a6ff-45d1-ab60-7bfc9162021a/download
7. Best Agile Tools for Software Engineering Students. University of Europe for Applied Sciences. URL: https://www.ue-germany.com/blog/best-agile-tools-for-software-engineering-students
8. Discover eduScrum: empirical processes in classroom. Letsscrumit: portal. URL: https://letsscrumit.com/discover-eduscrum
9. eduSCRUM як ефективний приклад тайм-менеджменту у класі. На Урок: освітній журнал. URL: https://naurok.com.ua/post/eduscrum-yak-efektivniy-priklad-taym-menedzhmentu-u-klasi

10. Guide to Agile in Education. Agile Alliance. 2024. URL: https://www.agilealliance.org/resources/guide-to-agile-in-education
11. Hoda R., Noble J. Becoming Agile: A Grounded Theory of Agile Transitions in Student Projects. IEEE Transactions on Education. 2021. Vol. 64, no. 2. P. 131–138.
12. How eduScrum works / W. Wijnands, K. Fritsch. eduScrum.org: official website. URL: https://eduscrum.org/how-eduscrum-works/
13. Jira vs Trello for Education: A Comparative Analysis. Atlassian Community. 2024. URL: https://community.atlassian.com/t5/Education-articles/Jira-vs-Trello-for-Education/ba-p/2345678
14. Kropp M., Meier A. Agile Methods in Higher Education: Adapting and Using eduScrum with Real World Projects. Proceedings of the International Conference on Software Engineering & Knowledge Engineering (SEKE). 2020. P. 155–160. DOI: 10.18293/SEKE2020-167.
15. Lazorenko O. Agile learning implementation and eduScrum practices in teaching English for Specific Purposes / O. Lazorenko, O. Krasnenko. International Journal of Learning, Teaching and Educational Research. 2020. URL: https://www.ijlter.org/index.php/ijlter/article/viewFile/2665/pdf
16. Product Backlog in eduScrum. Educational Resources Information Center (ERIC). URL: https://files.eric.ed.gov/fulltext/EJ1426478.pdf
17. Scrum – що це таке. Echometer: блог. URL: https://echometerapp.com/uk/scrum-%D1%89%D0%BE-%D1%86%D0%B5-%D1%82%D0%B0%D0%BA%D0%B5/
18. Scrum in Higher Education: A Literature Review. Aalto University. URL: https://aaltodoc.aalto.fi/bitstreams/7b96d6ec-c52a-4d03-b1f6-362b86d6a699/download
19. Sydorenko R., Vyshnevskyi O. Methodology of Building Agile-Education Processes in Higher Education Institutions. CEUR Workshop Proceedings. 2021. Vol. 2845. URL: https://ceur-ws.org/

20. The eduScrum Guide / eduScrum Team. 2019. URL: https://eduscrum.ru/wp-content/uploads/2020/01/The_eduScrum-guide-English_2.0_update_21-12-2019.pdf
21. Гнучкі методології розробки програмного забезпечення : навч. посіб. / КПІ ім. Ігоря Сікорського. URL: https://ela.kpi.ua/bitstreams/e0f9010b-937a-4859-9536-ba60c8271981/download
22. Делій А., ван Золінген Р., Вейнандс В. Керівництво по eduScrum: навчальне видання / ред. перекл. В. Л. Бурячок, В. Ю. Соколов. Версія 1.2. Київ: КУБГ, 2019. 36 с. URL: https://elibrary.kubg.edu.ua/id/eprint/29432/1/V_Buriachok_V_Sokolov_eduScrum.pdf
23. Колесникова Т. О. та ін. Суть Agile-управління та роль методології Scrum в Agile-команді. Економіка та управління бізнесом. 2025. № 2. URL: https://journals.dut.edu.ua/index.php/emb/article/view/1677
24. Ролі в методології Scrum : навч. посіб. / КПІ ім. Ігоря Сікорського. URL: https://ela.kpi.ua/bitstreams/e0f9010b-937a-4859-9536-ba60c8271981/download
25. Скрам: що це таке та як цим користуватися. Brander: блог. URL: https://brander.ua/blog/skram-shcho-tse-take-ta-yak-tsym-korystuvatysya
26. Технології управління проєктами: Scrum, Kanban, Scrumban. Lemon School. URL: https://lemon.school/blog/tehnologiyi-upravlinnya-proyektamy
27. Технології управління проєктами: Scrumban. Lemon School. URL: https://lemon.school/blog/tehnologiyi-upravlinnya-proyektamy
28. Що таке Agile та як працює гнучка методологія управління проектами? Cityhost. URL: https://cityhost.ua/uk/blog/scho-take-agile-yak-pracyu-gnuchka-metodologiya-upravlinnya-proektami.html

Додаток А

Текст програми

```
import React, { useState, useMemo } from 'react';
import {
  LayoutDashboard,
  ListTodo,
  KanbanSquare,
  BarChart3,
  Settings,
  Bell,
  Search,
  User,
  Plus,
  MessageSquare,
  Sparkles,
  CheckCircle2,
  AlertCircle,
  Clock,
  ArrowRight,
  X,
  Users,
  UserPlus,
  Trash2,
  Shield,
  ShieldCheck
} from 'lucide-react';

const COLUMNS = [
  { id: 'TODO', title: 'До виконання (To Do)' },
  { id: 'IN_PROGRESS', title: 'В роботі (In Progress)' },
  { id: 'PEER_REVIEW', title: 'Перевірка (Peer Review)' },
  { id: 'DONE', title: 'Готово (Done)' },
];

export default function App() {
  // Навігація та основний стан
  const [activeTab, setActiveTab] = useState('board');
  const [tasks, setTasks] = useState(INITIAL_TASKS);
  const [team, setTeam] = useState(INITIAL_TEAM);
  const [userRole, setUserRole] = useState('STUDENT'); //
'STUDENT' або 'TEACHER'

  // Стан для Drag & Drop
  const [draggedTaskId, setDraggedTaskId] = useState(null);

  // Стан для модальних вікон
  const [isAddTaskModalOpen, setIsAddTaskModalOpen] =
useState(false);
  const [newTaskColumn, setNewTaskColumn] = useState('TODO');
```

```
      const [newTaskData, setNewTaskData] = useState({ title: '',
assignee: '', points: 1, tags: '' });

      const  [isAddMemberModalOpen,  setIsAddMemberModalOpen]  =
useState(false);
      const [newMemberData, setNewMemberData] = useState({ name:
'', role: 'Developer', specialty: '' });

      const  [isSettingsModalOpen,  setIsSettingsModalOpen]  =
useState(false);
      const [appSettings, setAppSettings] = useState({
        notifications: true,
        theme: 'light',
        compactMode: false
      });

      const handleDragStart = (e, id) => {
        setDraggedTaskId(id);
        e.dataTransfer.effectAllowed = 'move';
        setTimeout(()  =>  e.target.classList.add('opacity-50'),
0);
      };

      const handleDragEnd = (e) => {
        e.target.classList.remove('opacity-50');
        setDraggedTaskId(null);
      };

      const handleDragOver = (e) => {
        e.preventDefault();
        e.dataTransfer.dropEffect = 'move';
      };

      const handleDrop = (e, status) => {
        e.preventDefault();
        if (draggedTaskId) {
          setTasks(tasks.map(t  =>  t.id  ===  draggedTaskId  ?  {
...t, status } : t));
        }
      };

      const moveTask = (id, newStatus) => {
        setTasks(tasks.map(t => t.id === id ? { ...t, status:
newStatus } : t));
      };

      const handleAddTask = (e) => {
        e.preventDefault();
        if (!newTaskData.title.trim()) return;

        const newTask = {
          id: `T-${tasks.length + 10}`,
          title: newTaskData.title,
```

```
          status: newTaskColumn,
          assignee: newTaskData.assignee || null,
          points: Number(newTaskData.points) || 1,
          tags:                newTaskData.tags                ?
newTaskData.tags.split(',').map(t => t.trim()).filter(Boolean) :
['New']
        };

        setTasks([...tasks, newTask]);
        setIsAddTaskModalOpen(false);
        setNewTaskData({ title: '', assignee: '', points: 1,
tags: '' });
      };

      const handleAddMember = (e) => {
        e.preventDefault();
        if (!newMemberData.name.trim()) return;
        setTeam([...team,      {      ...newMemberData,      id:
`u${Date.now()}` }]);
        setIsAddMemberModalOpen(false);
        setNewMemberData({   name:   '',   role:   'Developer',
specialty: '' });
      };

      const aiAdvice = useMemo(() => {
        const peerReviewTasks = tasks.filter(t => t.status ===
'PEER_REVIEW').length;
        const inProgressTasks = tasks.filter(t => t.status ===
'IN_PROGRESS').length;
        const  doneTasks  =  tasks.filter(t  =>  t.status  ===
'DONE').length;

        if (peerReviewTasks >= 2) {
          return { type: 'warning', text: `У колонці "Peer Review"
накопичилося  ${peerReviewTasks}  завдання.  Рекомендую  команді
зупинити розробку нового функціоналу і провести код-рев'ю.` };
        }
        if (inProgressTasks > 3) {
          return  {  type:  'alert',  text:  'Забагато завдань "В
роботі". Фокусуйтеся на завершенні поточних задач перед тим, як
брати нові (WIP Limit).' };
        }
        if (doneTasks > 0) {
          return { type: 'success', text: 'Чудовий темп! Ви вже
закрили частину завдань спринту. Продовжуйте в тому ж дусі.' };
        }
        return { type: 'info', text: 'Спринт розпочато. Бачу, що
завдання розподілені рівномірно. Не забувайте оновлювати статуси!'
};
      }, [tasks]);

      const Sidebar = () => (
```

```
        <div  className="w-64  bg-slate-900  text-slate-300  flex
flex-col h-screen shrink-0 hidden md:flex">
          <div  className="p-4  mb-4  flex  items-center  space-x-2
border-b border-slate-800">
            <div className="w-8 h-8 bg-indigo-500 rounded-lg flex
items-center justify-center">
              <Sparkles size={18} className="text-white" />
            </div>
            <span   className="text-xl   font-bold   text-white
tracking-wide">EduAgile</span>
          </div>
          <nav className="flex-1 px-3 space-y-1">
            {[
              { id: 'dashboard', icon: LayoutDashboard, label:
'Дашборд' },
              { id: 'backlog', icon: ListTodo, label: 'Беклог' },
              { id: 'board', icon: KanbanSquare, label: 'Спринт
дошка' },
              { id: 'team', icon: Users, label: 'Команда' },
              {  id:  'analytics',  icon:  BarChart3,  label:
'Аналітика' },
            ].map(item => (
              <button
                key={item.id}
                onClick={() => setActiveTab(item.id)}
                className={`w-full  flex  items-center  space-x-3
px-4 py-3 rounded-xl transition-colors ${
                  activeTab === item.id ? 'bg-indigo-600 text-
white shadow-md' : 'hover:bg-slate-800 hover:text-white'
                }`}
              >
                <item.icon size={20} />
                <span                              className="font-
medium">{item.label}</span>
              </button>
            ))}
          </nav>
          <div className="p-4 border-t border-slate-800">
            <button onClick={() => setIsSettingsModalOpen(true)}
className="flex items-center space-x-3 px-4 py-2 w-full rounded-
xl hover:bg-slate-800 transition-colors">
              <Settings size={20} />
              <span>Налаштування</span>
            </button>
          </div>
        </div>
      );

      const Header = () => (
        <header className="bg-white border-b border-slate-200 h-
16 flex items-center justify-between px-6 shrink-0 relative z-10">
          <div className="flex items-center space-x-4">
```

```
            <h1 className="text-xl font-bold text-slate-800
capitalize">
              {activeTab === 'dashboard' && 'Головна панель
(Dashboard)'}
              {activeTab === 'backlog' && 'Управління беклогом'}
              {activeTab === 'board' && 'Дошка поточного спринту
(Спринт #3)'}
              {activeTab === 'team' && 'Управління командою'}
              {activeTab === 'analytics' && 'Аналітика та
Оцінювання'}
            </h1>
            {activeTab === 'board' && (
              <span className="bg-indigo-100 text-indigo-700
text-xs font-semibold px-2 py-1 rounded-full flex items-center">
                <Clock size={12} className="mr-1" /> Залишилося
3 дні
              </span>
            )}
          </div>
          <div className="flex items-center space-x-4">
            {/* Рольовий перемикач */}
            <div className="flex items-center bg-slate-100 p-1
rounded-lg border border-slate-200">
              <button
                onClick={() => setUserRole('STUDENT')}
                className={`px-3 py-1 text-xs font-semibold
rounded-md transition-all ${userRole === 'STUDENT' ? 'bg-white
shadow-sm text-indigo-600' : 'text-slate-500 hover:text-slate-
700'}`}
              >
                Студент
              </button>
              <button
                onClick={() => setUserRole('TEACHER')}
                className={`px-3 py-1 text-xs font-semibold
rounded-md transition-all flex items-center ${userRole ===
'TEACHER' ? 'bg-white shadow-sm text-emerald-600' : 'text-slate-
500 hover:text-slate-700'}`}
              >
                <ShieldCheck size={14} className="mr-1" />
Викладач
              </button>
            </div>

            <div className="relative hidden sm:block">
              <Search size={18} className="absolute left-3 top-
1/2 -translate-y-1/2 text-slate-400" />
              <input type="text" placeholder="Пошук завдань..."
className="pl-10 pr-4 py-2 bg-slate-100 border-none rounded-lg
focus:ring-2 focus:ring-indigo-500 outline-none text-sm w-48" />
            </div>
            <button className="relative p-2 text-slate-400
hover:text-slate-600 transition-colors">
```

```
              <Bell size={20} />
              <span className="absolute top-1 right-1 w-2 h-2 bg-
rose-500 rounded-full border border-white"></span>
            </button>
            <div  className="w-9  h-9  bg-slate-200  rounded-full
flex items-center justify-center border-2 border-white shadow-sm
overflow-hidden">
              <img
src={`https://api.dicebear.com/7.x/avataaars/svg?seed=${userRole
=== 'TEACHER' ? 'Teacher' : 'Student'}`} alt="Avatar" />
            </div>
          </div>
        </header>
      );

      const AIAssistantWidget = () => (
        <div  className="bg-gradient-to-br  from-indigo-50  to-
purple-50  rounded-2xl  p-5  border  border-indigo-100  shadow-sm
relative overflow-hidden">
          <div  className="absolute  top-0  right-0  p-4  opacity-
10"><Sparkles size={100} /></div>
          <div  className="flex  items-center  space-x-2  mb-3
relative z-10">
            <div   className="bg-indigo-500   p-1.5   rounded-
lg"><Sparkles size={16} className="text-white" /></div>
            <h3 className="font-bold text-indigo-900">AI-Partner
Аналіз</h3>
          </div>
          <p className="text-indigo-800/80 text-sm mb-4 relative
z-10 leading-relaxed">{aiAdvice.text}</p>
          <div className="flex space-x-2 relative z-10">
            <button className="bg-indigo-600 hover:bg-indigo-700
text-white px-4 py-2 rounded-lg text-sm font-medium transition-
colors shadow-sm">Відкрити Pull Requests</button>
          </div>
        </div>
      );

      const DashboardView = () => (
        <div  className="p-6  bg-slate-50  flex-1  overflow-y-auto
space-y-6">
          <div className="grid grid-cols-1 lg:grid-cols-3 gap-6">
            <div className="lg:col-span-2 space-y-6">
              <div  className="bg-white  rounded-2xl  p-6  border
border-slate-200 shadow-sm flex flex-col md:flex-row gap-6">
                <div className="flex-1">
                  <h3 className="text-lg font-bold text-slate-
800 mb-2">Мета Спринту</h3>
                  <p  className="text-slate-600  text-sm  mb-
4">Розробити базовий каркас системи, підключити базу даних та
реалізувати систему авторизації.</p>
                  <div className="space-y-2">
```

```
                        <div   className="flex   justify-between
text-sm">
                            <span   className="font-medium   text-
slate-600">Прогрес виконання (Story Points)</span>
                            <span    className="font-bold    text-
indigo-600">
                              {tasks.filter(t   =>   t.status   ===
'DONE').reduce((acc, t) => acc + t.points, 0)} / {tasks.filter(t
=> t.status !== 'BACKLOG').reduce((acc, t) => acc + t.points, 0)}
                            </span>
                        </div>
                        <div className="w-full bg-slate-100 h-2.5
rounded-full overflow-hidden">
                            <div  className="bg-indigo-500  h-full
rounded-full   transition-all   duration-500"   style={{   width:
`${(tasks.filter(t => t.status === 'DONE').reduce((acc, t) => acc
+   t.points,   0)   /   (tasks.filter(t   =>   t.status   !==
'BACKLOG').reduce((acc, t) => acc + t.points, 0) || 1)) * 100}%`
}}></div>
                        </div>
                    </div>
                </div>
            </div>
            <AIAssistantWidget />
            <div  className="bg-white  rounded-2xl  p-6  border
border-slate-200 shadow-sm">
                <h3 className="text-lg font-bold text-slate-800
mb-4 flex items-center"><ListTodo size={20} className="mr-2 text-
slate-400" /> Активність команди</h3>
                <div className="space-y-4">
                    {[
                      { text: 'Іван перемістив "Верстка Канбан-
дошки" в Peer Review', time: '10 хв тому', icon: ArrowRight, color:
'text-amber-500', bg: 'bg-amber-100' },
                      { text: 'Олег закрив завдання "Створити
схему БД"', time: '2 год тому', icon: CheckCircle2, color: 'text-
emerald-500', bg: 'bg-emerald-100' },
                      { text: 'AI-Partner залишив коментар до
Pull Request #14', time: '5 год тому', icon: MessageSquare, color:
'text-indigo-500', bg: 'bg-indigo-100' },
                    ].map((act, i) => (
                      <div  key={i}  className="flex  items-start
space-x-3">
                        <div     className={`p-2     rounded-lg
${act.bg}  shrink-0`}><act.icon  size={16}  className={act.color}
/></div>
                        <div>
                          <p   className="text-sm   font-medium
text-slate-700">{act.text}</p>
                          <p   className="text-xs   text-slate-
400">{act.time}</p>
                        </div>
                    </div>
```

```
                    ))}
                  </div>
                </div>
              </div>
              <div className="space-y-6">
                 <div  className="bg-white  rounded-2xl  p-6  border
border-slate-200 shadow-sm">
                   <h3  className="text-lg  font-bold  text-slate-
800 mb-4">Мої завдання</h3>
                   <div className="space-y-3">
                     {tasks.filter(t => t.assignee === 'Марія' &&
t.status !== 'BACKLOG').map(task => (
                       <div key={task.id} className="p-3 border
border-slate-100  rounded-xl  hover:border-indigo-200  transition-
colors cursor-pointer flex justify-between items-center group">
                         <div>
                           <p  className="text-sm  font-medium
text-slate-800">{task.title}</p>
                           <p  className="text-xs  text-slate-
500 mt-1">{task.status}</p>
                         </div>
                         <button    className="text-indigo-600
opacity-0 group-hover:opacity-100 transition-opacity"><ArrowRight
size={16} /></button>
                       </div>
                     ))}
                     {tasks.filter(t => t.assignee === 'Марія' &&
t.status  !==  'BACKLOG').length  ===  0  &&  <p  className="text-sm
text-slate-500">Немає активних завдань</p>}
                   </div>
                 </div>
              </div>
            </div>
          </div>
        );

        const BacklogView = () => (
          <div className="p-6 bg-slate-50 flex-1 overflow-y-auto">
            <div className="max-w-4xl mx-auto bg-white rounded-2xl
border border-slate-200 shadow-sm overflow-hidden">
              <div  className="p-4  border-b  border-slate-200  bg-
slate-50 flex justify-between items-center">
                <h3  className="font-bold  text-slate-800">Беклог
Проєкту</h3>
                <button          onClick={()          =>          {
setNewTaskColumn('BACKLOG');    setIsAddTaskModalOpen(true);    }}
className="bg-indigo-600 hover:bg-indigo-700 text-white px-3 py-
1.5 rounded-lg text-sm font-medium transition-colors flex items-
center">
                  <Plus  size={16}  className="mr-1"  />  Додати
історію
                </button>
              </div>
```

```
            <div className="divide-y divide-slate-100">
              {tasks.filter(t => t.status === 'BACKLOG').map(task
=> (
                <div key={task.id} className="p-4 flex items-
center justify-between hover:bg-slate-50 transition-colors
group">
                  <div className="flex items-center space-x-4
flex-1">
                    <div className="text-xs font-semibold text-
slate-400 w-8">{task.id}</div>
                    <div  className="font-medium  text-slate-
800">{task.title}</div>
                    <div className="flex space-x-1">
                      {task.tags.map(tag => (
                        <span key={tag} className="text-[10px]
bg-slate-100 text-slate-600 px-1.5 py-0.5 rounded">{tag}</span>
                      ))}
                    </div>
                  </div>
                  <div className="flex items-center space-x-4">
                    <div  className="flex  items-center  justify-
center w-6 h-6 bg-slate-100 text-slate-600 rounded text-xs font-
bold" title="Story Points">{task.points}</div>
                    <button  onClick={()  =>  moveTask(task.id,
'TODO')} className="text-indigo-600 text-sm font-medium opacity-0
group-hover:opacity-100  transition-opacity  hover:underline">У
спринт →</button>
                  </div>
                </div>
              ))}
              {tasks.filter(t => t.status === 'BACKLOG').length
=== 0 && <div className="p-8 text-center text-slate-500">Беклог
порожній. Всі завдання у спринті!</div>}
            </div>
          </div>
        </div>
      );

      const BoardView = () => (
        <div className="flex-1 flex overflow-x-auto p-6 bg-slate-
50 space-x-6 items-start h-full">
          {COLUMNS.map(col => (
            <div
              key={col.id}
              className="flex-shrink-0  w-80  flex  flex-col  bg-
slate-100 rounded-xl max-h-full"
              onDragOver={handleDragOver}
              onDrop={(e) => handleDrop(e, col.id)}
            >
              <div className="p-3 border-b border-slate-200 flex
items-center justify-between">
                <h3 className="font-bold text-slate-700 text-sm
uppercase tracking-wider">{col.title}</h3>
```

```
                <div className="flex items-center space-x-1">
                  <span  className="bg-slate-200  text-slate-600
text-xs font-bold px-2 py-0.5 rounded-full mr-1">{tasks.filter(t
=> t.status === col.id).length}</span>
                  <button        onClick={()        =>        {
setNewTaskColumn(col.id);     setIsAddTaskModalOpen(true);     }}
className="p-1  text-slate-400  hover:text-indigo-600  hover:bg-
indigo-50 rounded transition-colors" title="Додати завдання">
                    <Plus size={16} />
                  </button>
                </div>
              </div>
              <div className={`p-3 flex-1 overflow-y-auto min-h-
[150px]  ${appSettings.compactMode  ?  'space-y-1.5'  :  'space-y-
3'}`}>
                {tasks.filter(t => t.status === col.id).map(task
=> (
                  <div
                    key={task.id}
                    draggable
                    onDragStart={(e)    =>    handleDragStart(e,
task.id)}
                    onDragEnd={handleDragEnd}
                    className={`bg-white  rounded-xl  shadow-sm
border   border-slate-200   cursor-grab   active:cursor-grabbing
hover:border-indigo-300    transition-colors    group    relative
${appSettings.compactMode ? 'p-2.5' : 'p-4'}`}
                  >
                    <div  className="flex  items-start  justify-
between mb-2">
                      <span   className="text-xs   font-semibold
text-slate-400">{task.id}</span>
                      <div className="flex -space-x-1">
                        {task.assignee ? (
                          <div  className="w-6  h-6  rounded-full
bg-slate-200 border border-white flex items-center justify-center
text-[10px] font-bold overflow-hidden" title={task.assignee}>
                            <img
src={`https://api.dicebear.com/7.x/avataaars/svg?seed=${task.ass
ignee}`} alt={task.assignee} />
                          </div>
                        ) : (
                          <div  className="w-6  h-6  rounded-full
bg-slate-50 border-2 border-slate-200 border-dashed flex items-
center   justify-center   text-slate-400   text-xs"   title="Не
призначено">
                            <User size={12} />
                          </div>
                        )}
                      </div>
                    </div>
```

```
                    <h4 className={`${appSettings.compactMode ?
'text-xs' : 'text-sm'} font-semibold text-slate-800 mb-3 leading-
snug`}>{task.title}</h4>
                    <div  className="flex  items-center  justify-
between">
                      <div className="flex flex-wrap gap-1">
                        {task.tags.map(tag => (
                          <span key={tag} className="text-[10px]
font-medium    bg-slate-100    text-slate-600    px-1.5    py-0.5
rounded">{tag}</span>
                        ))}
                      </div>
                      <div className="flex items-center justify-
center w-6 h-6 bg-indigo-50 text-indigo-600 rounded text-xs font-
bold">{task.points}</div>
                    </div>

                    {/* Кнопки переміщення для мобільних */}
                    <div   className="absolute   top-2   right-2
opacity-0   group-hover:opacity-100   transition-opacity   hidden
md:flex space-x-1">
                      {col.id === 'TODO' && <button onClick={()
=>  moveTask(task.id,  'IN_PROGRESS')}  className="p-1  bg-white
border   border-slate-200   shadow-sm   rounded   text-slate-500
hover:text-indigo-600"><ArrowRight size={14}/></button>}
                      {col.id  ===  'IN_PROGRESS'  &&  <button
onClick={() => moveTask(task.id, 'PEER_REVIEW')} className="p-1
bg-white border border-slate-200 shadow-sm rounded text-slate-500
hover:text-indigo-600"><ArrowRight size={14}/></button>}
                      {col.id  ===  'PEER_REVIEW'  &&  <button
onClick={() => moveTask(task.id, 'DONE')} className="p-1 bg-white
border   border-slate-200   shadow-sm   rounded   text-slate-500
hover:text-emerald-600"><CheckCircle2 size={14}/></button>}
                    </div>
                  </div>
                ))}
              </div>
            </div>
          ))}
        </div>
      );

      const TeamView = () => (
        <div  className="p-6  bg-slate-50  flex-1  overflow-y-auto
space-y-6">
          <div className="max-w-5xl mx-auto">
            <div className="flex flex-col sm:flex-row sm:justify-
between sm:items-center mb-6 gap-4">
              <div>
                <h2  className="text-2xl  font-bold  text-slate-
800">Команда проєкту</h2>
                <p    className="text-slate-500    text-sm    mt-
1">Управління учасниками, ролями та доступом</p>
```

```
            </div>
            {userRole === 'TEACHER' ? (
              <button              onClick={()              =>
setIsAddMemberModalOpen(true)} className="bg-indigo-600 hover:bg-
indigo-700 text-white px-4 py-2 rounded-lg text-sm font-medium
transition-colors flex items-center shadow-sm w-fit">
                <UserPlus size={18} className="mr-2" /> Додати
учасника
              </button>
            ) : (
              <div className="bg-amber-50 border border-amber-
200 text-amber-700 px-3 py-1.5 rounded-lg text-xs font-medium flex
items-center">
                <Shield size={14} className="mr-1.5" /> Лише
викладач може редагувати склад
              </div>
            )}
          </div>

          <div  className="grid  grid-cols-1  md:grid-cols-2
lg:grid-cols-3 gap-6">
            {team.map(member => (
              <div    key={member.id}    className="bg-white
rounded-2xl p-5 border border-slate-200 shadow-sm relative group
flex flex-col hover:border-indigo-200 transition-colors">
                {userRole === 'TEACHER' && (
                  <button
                    onClick={()  =>  setTeam(team.filter(m  =>
m.id !== member.id))}
                    className="absolute  top-4  right-4  text-
slate-300 hover:text-rose-500 opacity-0 group-hover:opacity-100
transition-opacity"
                    title="Видалити учасника"
                  >
                    <Trash2 size={18} />
                  </button>
                )}
                <div className="flex items-center space-x-4 mb-
4">
                  <div  className="w-12  h-12  bg-slate-100
rounded-full  flex  items-center  justify-center  overflow-hidden
border-2 border-white shadow-sm shrink-0">
                    <img
src={`https://api.dicebear.com/7.x/avataaars/svg?seed=${member.n
ame}`} alt={member.name} />
                  </div>
                  <div>
                    <h3    className="font-bold    text-slate-
800">{member.name}</h3>
                    <p  className="text-xs  text-slate-500  mt-
0.5">{member.specialty}</p>
                  </div>
                </div>
```

```
                    <div className="mt-auto pt-4 border-t border-
slate-50">
                      <label    className="text-xs    font-semibold
text-slate-400   uppercase   tracking-wider   mb-2   block">Роль
(Agile)</label>
                      {userRole === 'TEACHER' ? (
                        <select
                          value={member.role}
                          onChange={(e) => setTeam(team.map(m =>
m.id === member.id ? { ...m, role: e.target.value } : m))}
                          className="w-full  text-sm  font-medium
border  border-slate-200  rounded-lg  px-3  py-2  focus:ring-2
focus:ring-indigo-500  outline-none  bg-slate-50  hover:bg-white
transition-colors text-indigo-700"
                        >
                          <option
value="Developer">Developer</option>
                          <option    value="Scrum    Master">Scrum
Master</option>
                          <option   value="Product   Owner">Product
Owner</option>
                          <option      value="QA      Engineer">QA
Engineer</option>
                          <option   value="UI/UX   Designer">UI/UX
Designer</option>
                        </select>
                      ) : (
                        <div  className="bg-indigo-50  text-indigo-
700 px-3 py-2 rounded-lg text-sm font-medium inline-flex items-
center w-full">
                          <Shield  size={16}  className="mr-2"  />
{member.role}
                        </div>
                      )}
                    </div>
                  </div>
                ))}
                {team.length === 0 && (
                  <div  className="col-span-full  py-12  text-center
text-slate-500 bg-white rounded-2xl border border-dashed border-
slate-300">
                    <Users size={48} className="mx-auto mb-3 text-
slate-300" />
                    Команда порожня. Додайте першого учасника!
                  </div>
                )}
              </div>
            </div>
          </div>
        );

        const AnalyticsView = () => (
```

```
        <div className="p-6 bg-slate-50 flex-1 overflow-y-auto space-y-6">
          <div className="grid grid-cols-1 lg:grid-cols-2 gap-6 max-w-6xl mx-auto">
            {/* Burndown Chart - доступний всім */}
            <div className="bg-white rounded-2xl p-6 border border-slate-200 shadow-sm lg:col-span-2 xl:col-span-1">
              <h3 className="text-lg font-bold text-slate-800 mb-4">Burndown Діаграма</h3>
              <div className="h-64 flex items-end justify-between space-x-2 pt-4 relative">
                <div className="absolute left-0 top-0 bottom-0 w-8 flex flex-col justify-between text-xs text-slate-400 border-r border-slate-200 pb-6">
                  <span>40</span><span>30</span><span>20</span><span>10</span><span>0</span>
                </div>
                <div className="absolute left-8 right-0 bottom-0 h-6 flex justify-between text-xs text-slate-400 pt-2 border-t border-slate-200 px-2">
                  <span>День 1</span><span>День 3</span><span>День 5</span><span>День 7</span>
                </div>
                <div className="ml-8 flex-1 flex justify-around items-end h-full pb-8">
                  {[40, 35, 28, 28, 20, 15, 5].map((ideal, idx) => {
                    const actual = [40, 40, 38, 25, 22, 10, 0][idx];
                    return (
                      <div key={idx} className="relative w-8 sm:w-10 h-full flex items-end group">
                        <div className="absolute bottom-0 w-full bg-slate-100 rounded-t-sm" style={{ height: `${(ideal/40)*100}%` }}></div>
                        <div className="absolute bottom-0 w-full bg-indigo-500 rounded-t-sm opacity-80" style={{ height: `${(actual/40)*100}%` }}></div>
                        <div className="absolute -top-10 left-1/2 -translate-x-1/2 bg-slate-800 text-white text-xs py-1 px-2 rounded opacity-0 group-hover:opacity-100 transition-opacity z-10 whitespace-nowrap pointer-events-none">
                          План: {ideal} SP<br/>Факт: {actual} SP
                        </div>
                      </div>
                    )
                  })}
                </div>
              </div>
            </div>
```

```
            {/* Матриця контрибуції - залежить від ролі */}
            <div  className="bg-white  rounded-2xl  p-6  border
border-slate-200 shadow-sm lg:col-span-2 xl:col-span-1 flex flex-
col">
                <h3 className="text-lg font-bold text-slate-800
mb-4 flex justify-between items-center">
                  <span>Матриця контрибуції</span>
                  {userRole === 'TEACHER' ? (
                    <span   className="text-xs   bg-emerald-100
text-emerald-700 px-2 py-1 rounded flex items-center"><ShieldCheck
size={12} className="mr-1"/> Викладацький доступ</span>
                  ) : (
                    <span  className="text-xs bg-slate-100 text-
slate-500 px-2 py-1 rounded">Приховано для студентів</span>
                  )}
                </h3>

                {userRole === 'TEACHER' ? (
                  <div className="overflow-x-auto flex-1">
                     <table  className="w-full  text-left  text-
sm">
                        <thead>
                           <tr className="border-b border-slate-
200 text-slate-500">
                              <th     className="pb-3     font-
medium">Студент</th>
                              <th   className="pb-3   font-medium
text-center">Виконано SP</th>
                              <th   className="pb-3   font-medium
text-center">Коміти</th>
                              <th   className="pb-3   font-medium
text-center">Оцінка системи</th>
                           </tr>
                        </thead>
                        <tbody className="divide-y divide-slate-
100">
                           <tr className="hover:bg-slate-50">
                              <td  className="py-3  flex  items-
center space-x-2">
                                 <div     className="w-6     h-6
rounded-full          bg-slate-200          overflow-hidden"><img
src="https://api.dicebear.com/7.x/avataaars/svg?seed=Олег"
alt="Олег"/></div>
                                 <span    className="font-medium
text-slate-800">Олег</span>
                              </td>
                              <td   className="py-3   text-center
font-bold text-indigo-600">14</td>
                              <td   className="py-3   text-center
text-slate-600">23</td>
                              <td     className="py-3     text-
center"><span          className="text-emerald-500          font-
medium">Відмінно</span></td>
```

```
                            </tr>
                            <tr className="hover:bg-slate-50">
                               <td  className="py-3  flex  items-
center space-x-2">
                                  <div     className="w-6     h-6
rounded-full          bg-slate-200          overflow-hidden"><img
src="https://api.dicebear.com/7.x/avataaars/svg?seed=Марія"
alt="Марія"/></div>
                                  <span    className="font-medium
text-slate-800">Марія</span>
                               </td>
                               <td  className="py-3  text-center
font-bold text-indigo-600">8</td>
                               <td  className="py-3  text-center
text-slate-600">12</td>
                               <td     className="py-3     text-
center"><span          className="text-emerald-500          font-
medium">Добре</span></td>
                            </tr>
                            <tr className="hover:bg-slate-50">
                               <td  className="py-3  flex  items-
center space-x-2">
                                  <div     className="w-6     h-6
rounded-full          bg-slate-200          overflow-hidden"><img
src="https://api.dicebear.com/7.x/avataaars/svg?seed=Іван"
alt="Іван"/></div>
                                  <span    className="font-medium
text-slate-800">Іван</span>
                               </td>
                               <td  className="py-3  text-center
font-bold text-rose-500">0</td>
                               <td  className="py-3  text-center
text-slate-600">2</td>
                               <td     className="py-3     text-
center"><span  className="text-rose-500  font-medium  text-xs  flex
items-center        justify-center"><AlertCircle        size={12}
className="mr-1"/>Free-rider</span></td>
                            </tr>
                         </tbody>
                      </table>
                   </div>
                ) : (
                   <div  className="flex-1  flex  flex-col  items-
center justify-center text-center py-8">
                      <Shield  className="text-slate-300  w-16  h-16
mb-4" />
                      <h4  className="text-slate-700  font-semibold
mb-2">Обмежений доступ</h4>
                      <p  className="text-slate-500  text-sm  max-w-
xs">Детальна аналітика та оцінювання внеску доступні лише власнику
продукту (Викладачу).</p>
                   </div>
                )}
```

```
            </div>
          </div>
        </div>
      );

      return (
        <div   className="flex   h-screen   bg-white   font-sans
overflow-hidden text-slate-800">
          <Sidebar />
          <div className="flex-1 flex flex-col min-w-0">
            <Header />
            <main className="flex-1 overflow-hidden flex flex-col
relative bg-white">
              {activeTab === 'dashboard' && <DashboardView />}
              {activeTab === 'backlog' && <BacklogView />}
              {activeTab === 'board' && <BoardView />}
              {activeTab === 'team' && <TeamView />}
              {activeTab === 'analytics' && <AnalyticsView />}
            </main>
          </div>

          {/* Модальне вікно створення завдання */}
          {isAddTaskModalOpen && (
            <div  className="fixed inset-0 bg-slate-900/50 flex
items-center justify-center z-50 p-4">
              <div  className="bg-white  rounded-2xl  p-6  w-full
max-w-md shadow-xl border border-slate-100">
                <div   className="flex   justify-between   items-
center mb-4">
                  <h2  className="text-xl  font-bold  text-slate-
800">Нове завдання</h2>
                  <button             onClick={()             =>
setIsAddTaskModalOpen(false)}          className="text-slate-400
hover:text-slate-600">
                    <X size={20} />
                  </button>
                </div>
                <form onSubmit={handleAddTask} className="space-
y-4">
                  <div>
                    <label  className="block  text-sm  font-medium
text-slate-700 mb-1">Назва завдання</label>
                    <input   type="text"   autoFocus   required
value={newTaskData.title}             onChange={e             =>
setNewTaskData({...newTaskData,     title:     e.target.value})}
className="w-full  border  border-slate-300  rounded-lg  px-3  py-2
focus:outline-none      focus:ring-2      focus:ring-indigo-500"
placeholder="Наприклад: Написати тести" />
                  </div>
                  <div className="grid grid-cols-2 gap-4">
                    <div>
                      <label   className="block   text-sm   font-
medium text-slate-700 mb-1">Виконавець</label>
```

```
                        <select       value={newTaskData.assignee}
onChange={e    =>    setNewTaskData({...newTaskData,    assignee:
e.target.value})}   className="w-full   border   border-slate-300
rounded-lg px-3 py-2 focus:outline-none focus:ring-2 focus:ring-
indigo-500">
                          <option value="">Не призначено</option>
                          {team.map(member       =>       <option
key={member.id} value={member.name}>{member.name}</option>)}
                        </select>
                      </div>
                      <div>
                        <label   className="block   text-sm   font-
medium text-slate-700 mb-1">Story Points</label>
                        <input   type="number"   min="1"   max="21"
value={newTaskData.points}             onChange={e             =>
setNewTaskData({...newTaskData,     points:     e.target.value})}
className="w-full border border-slate-300 rounded-lg px-3 py-2
focus:outline-none focus:ring-2 focus:ring-indigo-500" />
                      </div>
                    </div>
                    <div>
                      <label className="block text-sm font-medium
text-slate-700 mb-1">Теги (через кому)</label>
                      <input type="text" value={newTaskData.tags}
onChange={e     =>     setNewTaskData({...newTaskData,     tags:
e.target.value})}   className="w-full   border   border-slate-300
rounded-lg px-3 py-2 focus:outline-none focus:ring-2 focus:ring-
indigo-500" placeholder="Frontend, UI" />
                    </div>
                    <div className="pt-4 flex justify-end space-x-
3">
                      <button    type="button"    onClick={()    =>
setIsAddTaskModalOpen(false)} className="px-4 py-2 text-slate-600
hover:bg-slate-100     rounded-lg     transition-colors     font-
medium">Скасувати</button>
                      <button type="submit" className="px-4 py-2
bg-indigo-600    hover:bg-indigo-700    text-white    rounded-lg
transition-colors font-medium">Створити</button>
                    </div>
                  </form>
                </div>
              </div>
            )}

            {/* Модальне вікно додавання учасника */}
            {isAddMemberModalOpen && (
              <div  className="fixed  inset-0  bg-slate-900/50  flex
items-center justify-center z-50 p-4">
                <div  className="bg-white  rounded-2xl  p-6  w-full
max-w-md shadow-xl border border-slate-100">
                  <div   className="flex   justify-between   items-
center mb-6">
```

```
                    <h2 className="text-xl font-bold text-slate-
800">Новий учасник</h2>
                    <button onClick={() =>
setIsAddMemberModalOpen(false)} className="text-slate-400
hover:text-slate-600">
                      <X size={20} />
                    </button>
                  </div>
                  <form onSubmit={handleAddMember}
className="space-y-4">
                    <div>
                      <label className="block text-sm font-medium
text-slate-700 mb-1">Ім'я студента</label>
                      <input type="text" autoFocus required
value={newMemberData.name} onChange={e =>
setNewMemberData({...newMemberData, name: e.target.value})}
className="w-full border border-slate-300 rounded-lg px-3 py-2
focus:outline-none focus:ring-2 focus:ring-indigo-500"
placeholder="Анна" />
                    </div>
                    <div>
                      <label className="block text-sm font-medium
text-slate-700 mb-1">Спеціалізація</label>
                      <input type="text"
value={newMemberData.specialty} onChange={e =>
setNewMemberData({...newMemberData, specialty: e.target.value})}
className="w-full border border-slate-300 rounded-lg px-3 py-2
focus:outline-none focus:ring-2 focus:ring-indigo-500"
placeholder="Наприклад: Backend Розробник" />
                    </div>
                    <div>
                      <label className="block text-sm font-medium
text-slate-700 mb-1">Agile Роль</label>
                      <select value={newMemberData.role}
onChange={e => setNewMemberData({...newMemberData, role:
e.target.value})} className="w-full border border-slate-300
rounded-lg px-3 py-2 focus:outline-none focus:ring-2 focus:ring-
indigo-500">
                        <option
value="Developer">Developer</option>
                        <option value="Scrum Master">Scrum
Master</option>
                        <option value="QA Engineer">QA
Engineer</option>
                        <option value="UI/UX Designer">UI/UX
Designer</option>
                      </select>
                    </div>
                    <div className="pt-4 flex justify-end space-x-
3">
                      <button type="button" onClick={() =>
setIsAddMemberModalOpen(false)} className="px-4 py-2 text-slate-
```

```
600 hover:bg-slate-100 rounded-lg transition-colors font-
medium">Скасувати</button>
                    <button type="submit" className="px-4 py-2
bg-indigo-600 hover:bg-indigo-700 text-white rounded-lg
transition-colors font-medium">Додати</button>
                  </div>
                </form>
              </div>
            </div>
          )}

          {/* Модальне вікно налаштувань */}
          {isSettingsModalOpen && (
            <div className="fixed inset-0 bg-slate-900/50 flex
items-center justify-center z-50 p-4">
              <div className="bg-white rounded-2xl p-6 w-full
max-w-md shadow-xl border border-slate-100">
                <div  className="flex  justify-between  items-
center mb-6">
                  <h2 className="text-xl font-bold text-slate-
800">Налаштування системи</h2>
                  <button            onClick={()            =>
setIsSettingsModalOpen(false)}         className="text-slate-400
hover:text-slate-600">
                    <X size={20} />
                  </button>
                </div>
                <div className="space-y-6">
                  <div  className="flex  items-center  justify-
between">
                    <div>
                      <h4 className="text-sm font-medium text-
slate-800">Сповіщення</h4>
                      <p     className="text-xs     text-slate-
500">Отримувати пуш-сповіщення</p>
                    </div>
                    <label    className="relative    inline-flex
items-center cursor-pointer">
                      <input                    type="checkbox"
checked={appSettings.notifications}       onChange={(e)       =>
setAppSettings({...appSettings,                   notifications:
e.target.checked})} className="sr-only peer" />
                      <div className="w-11 h-6 bg-slate-200 peer-
focus:outline-none         rounded-full         peer         peer-
checked:after:translate-x-full    peer-checked:after:border-white
after:content-['']  after:absolute  after:top-[2px]  after:left-
[2px]    after:bg-white    after:border-slate-300    after:border
after:rounded-full after:h-5 after:w-5 after:transition-all peer-
checked:bg-indigo-600"></div>
                    </label>
                  </div>
                  <div  className="flex  items-center  justify-
between">
```

```
                    <div>
                      <h4 className="text-sm font-medium text-
slate-800">Компактний режим</h4>
                      <p     className="text-xs     text-slate-
500">Зменшити відступи на дощці</p>
                    </div>
                    <label    className="relative    inline-flex
items-center cursor-pointer">
                      <input                      type="checkbox"
checked={appSettings.compactMode}        onChange={(e)        =>
setAppSettings({...appSettings, compactMode: e.target.checked})}
className="sr-only peer" />
                      <div classNam="w-11 h-6 bg-slate-200 peer-
focus:outline-none        rounded-full        peer        peer-
checked:after:translate-x-full    peer-checked:after:border-white
after:content-['']  after:absolute  after:top-[2px]  after:left-
[2px]    after:bg-white    after:border-slate-300    after:border
after:rounded-full after:h-5 after:w-5 after:transition-all peer-
checked:bg-indigo-600"></div>
                    </label>
                  </div>
                </div>
                <div className="pt-6 mt-6 border-t border-slate-
100 flex justify-end">
                  <button             onClick={()             =>
setIsSettingsModalOpen(false)} className="px-4 py-2 bg-indigo-600
hover:bg-indigo-700 text-white rounded-lg transition-colors font-
medium">
                    Закрити
                  </button>
                </div>
              </div>
            </div>
          )}
        </div>
      );
    }
```

```
    -- SQL Сценарій створення бази даних студентської Agile-
системи
    -- Сумісно з PostgreSQL 12+
    -- Створено автоматично в Interactive DB Visualizer

    -- Таблиця Користувачі (users)
    CREATE TABLE users (
        id SERIAL PRIMARY KEY,
        email VARCHAR(150) UNIQUE NOT NULL,
        password_hash VARCHAR(255) NOT NULL,
        first_name VARCHAR(50) NOT NULL,
        last_name VARCHAR(50) NOT NULL,
        system_role VARCHAR(20) NOT NULL,
```

```
        CONSTRAINT chk_system_role CHECK (system_role IN
('STUDENT', 'TEACHER'))
    );

    -- Таблиця Команди (teams)
    CREATE TABLE teams (
        id SERIAL PRIMARY KEY,
        name VARCHAR(100) NOT NULL,
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    );

    -- Таблиця Учасники Команд (team_members)
    CREATE TABLE team_members (
        team_id INT REFERENCES teams(id) ON DELETE CASCADE,
        user_id INT REFERENCES users(id) ON DELETE CASCADE,
        team_role VARCHAR(30) NOT NULL,
        PRIMARY KEY (team_id, user_id),
        CONSTRAINT chk_team_role CHECK (team_role IN
('SCRUM_MASTER', 'DEVELOPER'))
    );

    -- Таблиця Проєкти (projects)
    CREATE TABLE projects (
        id SERIAL PRIMARY KEY,
        name VARCHAR(150) NOT NULL,
        description TEXT,
        team_id INT REFERENCES teams(id) ON DELETE SET NULL,
        mentor_id INT REFERENCES users(id) ON DELETE RESTRICT,
        start_date DATE,
        end_date DATE,
        CONSTRAINT chk_project_dates CHECK (end_date >
start_date)
    );

    -- Таблиця Спринти (sprints)
    CREATE TABLE sprints (
        id SERIAL PRIMARY KEY,
        project_id INT REFERENCES projects(id) ON DELETE CASCADE,
        name VARCHAR(50) NOT NULL,
        start_date DATE NOT NULL,
        end_date DATE NOT NULL,
        status VARCHAR(20) NOT NULL,
        CONSTRAINT chk_sprint_dates CHECK (end_date >
start_date),
        CONSTRAINT chk_sprint_status CHECK (status IN ('PLANNED',
'ACTIVE', 'CLOSED'))
    );

    -- Таблиця Завдання (tasks)
    CREATE TABLE tasks (
        id SERIAL PRIMARY KEY,
        project_id INT REFERENCES projects(id) ON DELETE CASCADE,
        sprint_id INT REFERENCES sprints(id) ON DELETE SET NULL,
```

```
        assignee_id INT REFERENCES users(id) ON DELETE SET NULL,
        title VARCHAR(200) NOT NULL,
        description TEXT,
        status VARCHAR(20) NOT NULL,
        priority VARCHAR(15) NOT NULL,
        story_points INT DEFAULT 0,
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        CONSTRAINT chk_task_sp CHECK (story_points >= 0),
        CONSTRAINT chk_task_status CHECK (status IN ('TODO',
'IN_PROGRESS', 'REVIEW', 'DONE')),
        CONSTRAINT chk_task_priority CHECK (priority IN ('LOW',
'MEDIUM', 'HIGH'))
    );

    -- Таблиця Логування Часу (work_logs)
    CREATE TABLE work_logs (
        id SERIAL PRIMARY KEY,
        task_id INT REFERENCES tasks(id) ON DELETE CASCADE,
        user_id INT REFERENCES users(id) ON DELETE CASCADE,
        hours_spent NUMERIC(5,2) NOT NULL,
        comment VARCHAR(255),
        logged_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        CONSTRAINT chk_hours_spent CHECK (hours_spent > 0)
    );

    -- Таблиця Оцінки Спринтів (sprint_reviews)
    CREATE TABLE sprint_reviews (
        id SERIAL PRIMARY KEY,
        sprint_id INT UNIQUE REFERENCES sprints(id) ON DELETE
CASCADE,
        teacher_id INT REFERENCES users(id),
        grade INT NOT NULL,
        feedback TEXT,
        reviewed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        CONSTRAINT chk_grade CHECK (grade BETWEEN 0 AND 100)
    );
```