Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:
Веб-додаток для створення, організації та відстеження індивідуальних й командних задач і проектів

Виконав студент 2 курсу, групи 1КІ-21м
спеціальності 123 — Комп'ютерна інженерія
Льопа Б. В.
Керівник д.т.н., проф. каф.ОТ
Азаров О. Д.
"15" 12 2023 р.
Опонент к.т.н., доц. каф. МБІС
Грицак А. В.
"17" 12 2023 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.

"19" 12 2023 р.

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Галузь знань — Інформаційні технології
Освітній рівень — магістр
Спеціальність — 123 Комп'ютерна інженерія
Освітня програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри обчислювальної техніки

О.Д. Азаров
" 26 " вересня 2023 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ
студенту **Льопі Богдану Вікторовичу**

1 Тема роботи «Веб-додаток для створення, організації та відстеження індивідуальних й командних задач і проектів» керівник роботи Азаров О.Д. д.т.н., проф. каф.ОТ, затверджено наказом вищого навчального закладу від 18.09.2023 року № **247**

2 Строк подання студентом роботи **17**.12.2023

3 Вихідні дані до роботи: розробка веб-додатку для створення, організації та відстеження індивідуальних й командних задач і проектів; використовувані технології — фреймворк Next.js, база даних PostgreSQL, платформа Supabase, інструмент Prisma, бібліотеки React Query, Tailwind; інтеграція з ChatGPT для написання опису та рекомендацій для задач.

4 Зміст розрахунково-пояснювальної записки: вступ, Аналітичний огляд сучасного стану управління завданнями та проектами, теоретичні дослідження, розробка поставленої задачі, тестування експериментальних досліджень, економічна частина.

5 Перелік графічного матеріалу: технічне завдання, діаграма класів, діаграма послідовностей, скріни роботи програми, лістинг програми.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Кадук Олександр Володимирович к.т.н., доцент		
4	Небава Микола Іванович проф. к.е.н.,		

7 Дата видачі завдання 19.09.2023р.

8 Календарний план виконання МКР приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів МКР	Строк виконання	Підпис
1	Постановка задачі	19.09.2023	
2	Огляд існуючих рішень	24.09.2023	
3	Теоретичні дослідження	29.09.2023	
4	Розробка структурної схеми	03.10.2023	
5	Вибір технологій для розробки	07.10.2023	
6	Моделювання роботи додатку	09.10.2023	
7	Розробка поставленої задачі	11.10.2023	
8	Розрахунок економічної частини	01.11.2023	
9	Оформлення пояснювальної записки та ілюстративного матеріалу	20.10.2023	
10	Виконання магістерської кваліфікаційної роботи	25.10.2023	
11	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	14.12.2023	
12	Підписи супроводжувальних документів у керівника, опонента, нормоконтролера	15.12.2023	
13	Перевірка «антиплагіат»	14.12.2023	
14	Попередній захист	07.11.2023	

Студент Льопа Б.В.

Керівник д.т.н., проф. каф.ОТ Азаров О.Д.

АНОТАЦІЯ

УДК 004.774.6

Льопа Б. В. Веб-додаток для створення, організації та відстеження індивідуальних й командних задач і проектів. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп'ютерна Інженерія, Вінниця: ВНТУ, 2023 — 137 с. На укр. мові. Бібліогр.: 23 назв; рис. 24; табл.: 8.

Робота присвячена розробці веб-додатку, який дозволяє користувачам створювати, організовувати та відстежувати свої індивідуальні та командні задачі та проекти. Додаток базується на концепції дошки зі списками карток, які містять інформацію про задачі. Додаток реалізовано з використанням сучасних технологій, таких як Next.js, PostgreSQL, Supabase, Prisma, React Query, Tailwind. Додаток також інтегровано з ChatGPT для генерації опису та рекомендацій для задач. Робота містить аналіз сучасного стану проблеми, концепцію та архітектуру додатку, реалізацію додатку, експериментальну перевірку та оцінку ефективності додатку. Результати роботи можуть бути використані для підвищення продуктивності та співпраці в різних сферах діяльності, веб-додаток може бути використаний як особистий або командний інструмент для планування та виконання різноманітних задач і проектів.

Ключові слова: веб-додаток, проекти, задачі, ChatGPT, Next.js, управління.

ABSTRACT

УДК 004.774.6

Lopa B. V. A web application for creating, organizing and tracking individual and team tasks and projects. Master's thesis in specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2023 — 148 p. In Ukrainian. Bibliogr.: 42 titles; fig.: 24; tabl. 8.

The work is devoted to the development of a web application that allows users to create, organize and track their individual and team tasks and projects. The application is based on the concept of a board with lists of cards that contain information about tasks. The application is implemented using modern technologies such as Next.js, PostgreSQL, Supabase, Prisma, React Query, Tailwind. The application is also integrated with ChatGPT for generating description and recommendations for tasks. The work contains an analysis of the current state of the problem, the concept and architecture of the application, the implementation of the application, the experimental verification and evaluation of the effectiveness of the application. The results of the work can be used to improve productivity and collaboration in various fields of activity, the web application can be used as a personal or team tool for planning and executing various tasks and projects.

Keywords: web application, projects, tasks, ChatGPT, Next.js, management.

ЗМІСТ

ВСТУП

У сучасному світі інформаційні технології відіграють важливу роль у різних сферах людської діяльності. Одним з найпоширеніших застосувань інформаційних технологій є управління проектами, яке передбачає планування, організацію, виконання та контроль різних задач і процесів, пов’язаних з досягненням певної мети. У середовищі інформаційних технологій, що постійно розвивається, потреба в спрощеному управлінні завданнями та проектами стала першочерговою. У міру того як організації та окремі особи стикаються зі зростаючим обсягом обов’язків, ефективність управління завданнями та проектами відіграє ключову роль в успіху. Управління завданнями є критично важливим процесом для окремих людей і команд будь-якого розміру. Це допомагає забезпечити виконання завдань вчасно та в межах бюджету, а також ефективне використання ресурсів. Управління проектами вимагає ефективних інструментів для співпраці, комунікації, документування, моніторингу та аналізу даних. Особливо актуальними є веб-додатки для управління проектами, які дозволяють користувачам доступатися до своїх проектів з будь-якого пристрою, що має підключення до Інтернету, та співпрацювати з іншими учасниками проекту в режимі реального часу.

Управління проектами — це методологія планування та використання ресурсів протягом його життєвого циклу. Для вдалого завершення проекту необхідна не тільки команда кваліфікованих співробітників, але і інструменти, за допомогою яких можна фіксувати виконані задачі та вдало управляти ресурсами. Серед чітко орієнтованих є рішення, які підходять для управління проектами різних видів. Проте, зазвичай, вони є дорого вартісними і для компаній, які тільки знаходяться в стані становлення, це суттєві витрати. Традиційні методи керування завданнями, такі як списки справ і електронні таблиці, можуть бути ефективними, але вони також можуть бути громіздкими та трудомісткими для підтримки.

Веб-додатки для керування завданнями пропонують низку переваг перед традиційними методами. Як правило, вони прості у використанні, більш

масштабовані та зручніші для співпраці. Крім того, веб-програми можуть надавати низку функцій, які недоступні з традиційними методами.

Попит на веб-додатки для керування завданнями стрімко зростає. Це пов’язано з низкою факторів, зокрема зростаючою популярністю віддаленої роботи, зростаючою складністю проектів і потребою в більш ефективній співпраці команд.

Актуальність обраної теми підкреслена тим, що сучасні технології не тільки полегшують, але і змінюють спосіб, яким люди взаємодіють з завданнями і проектами. Виникає потреба в інтегрованих, інтуїтивно зрозумілих та ефективних інструментах для управління робочим процесом. Спостерігається ріст командної роботи, що ставить перед собою завдання вдосконалити інструменти для колективного виконання завдань, сприяючи координації та співпраці. Існуючі веб-додатки для управління проектами не завжди задовольняють потреби користувачів, що стосується зручності, функціональності, продуктивності та інтелектуальної підтримки. Тому необхідно розробити новий веб-додаток для створення, організації та відстеження індивідуальних й командних задач і проектів, який би враховував сучасні тенденції розвитку веб-технологій та вимоги користувачів. Такий додаток має велике практичне значення для різних сфер діяльності, де застосовуються методи управління проектами.

Метою даної магістерської кваліфікаційної роботи є глибокий аналіз поточних підходів до управління завданнями та проектами, а також розробка високопродуктивного веб-додатку, який враховує потреби як індивідуальних користувачів, так і робочих груп. Це включає в себе багатогранний підхід, що охоплює дизайн взаємодії з користувачем, архітектуру програмного забезпечення та практичну реалізацію. Завдання включають комплексний аналіз існуючих інструментів, незалежне дослідження та використання передових комп’ютерних технологій для ефективного управління завданнями та проектами. Також буде розглянуто використання штучного інтелекту (ШІ) для покращення функціональності та зручності використання програми.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

— проаналізувати існуючі веб-додатки для управління проектами та визначити їх переваги та недоліки;

— визначити вимоги та функціональні можливості до розроблюваного веб-додатку;

— обрати та обґрунтувати технології та інструменти для розробки веб-додатку;

— розробити архітектуру та дизайн веб-додатку;

— реалізувати веб-додаток з використанням обраних технологій та інструментів;

— провести тестування та оцінку якості та продуктивності веб-додатку;

— дослідити можливості застосування штучного інтелекту для підтримки користувачів веб-додатку;

— проаналізувати отримані результати та зробити висновки.

Об’єктом дослідження є процес управління проектами за допомогою веб-додатків.

Предметом дослідження є методи та засоби створення, організації та відстеження індивідуальних й командних задач і проектів у веб-додатку.

Новизна отриманих результатів полягатиме в інноваційному підході до вирішення проблеми управління завданнями та в створенні ефективного інструменту для роботи як індивідів, так і команд, вона буде досягнута завдяки інтеграції штучного інтелекту та впровадженню передових сучасних технологій, які роблять акцент на забезпеченні максимальної зручності, швидкості та ефективності в користуванні.

Практичне значення одержаних результатів полягає в тому, що розроблений веб-додаток може бути використаний для управління різними видами проектів, як для командних так і для індивідуальних потреб.

Отже, дана робота спрямована на заповнення прогалини в сфері управління завданнями, а її результати матимуть практичне значення для

широкого кола користувачів, прагнучи підвищити продуктивність та організованість в їхній роботі.

Отримані результати не лише розширять та удосконалять ринок інструментів для управління завданнями, але і забезпечать ключовий внесок у розвиток області інформаційного управління проектами. Такий підхід передбачає не лише технічну реалізацію, але й комплексний погляд на проблему, що забезпечить глибше розуміння факторів, які визначають ефективність управління завданнями та проектами в сучасному світі.

1 АНАЛІТИЧНИЙ ОГЛЯД СУЧАСНОГО СТАНУ УПРАВЛІННЯ ЗАВДАННЯМИ ТА ПРОЕКТАМИ

1.1 Огляд предметної області

Управління проектом відкриває можливість передбачити й розв'язати можливі труднощі, завдяки збору необхідної інформації та знань. Завдяки цьому, можна ефективно готувати, організовувати, планувати та координувати всі аспекти діяльності для успішної реалізації проекту. Ризики та зворотній зв'язок залишаються непередбачуваними факторами, особливо для великих та складних проектів, але можна вжити заходів для їх запобігання через докладне планування та використання передового досвіду та інструментів управління проектами.

Управління проектами дозволяє організаціям ставити проекти на конкретні етапи, щоб відповідати термінам, очікуванням та бюджету з мінімальним ризиком. Це забезпечує систематичний підхід до керування проектами, що дозволяє виконувати їх ефективно та послідовно в межах відведеного часу та бюджету. Управління проектами приносить численні переваги, такі як визначення дій, підвищення продуктивності, встановлення бюджету, покращення комунікації та зменшення ризиків [1].

З урахуванням росту індустрії програмного забезпечення, багато організацій використовують автоматизовані системи управління проектами. Мета управління проектами полягає у використанні перевірених методів та передових практик для ефективного керування людьми, діяльністю та ресурсами, що дозволяє виконувати проекти ефективно та зменшує ймовірність проблем.

Визначивши мету управління проектами тепер можна розглянути його переваги. Менеджмент проектів надає перевірені методи та передові практики, а також повторювані процеси, які допоможуть організації керувати людьми, діяльністю та ресурсами, задіяними її проектах. Таким чином, це дає змогу виконувати проекти з ефективністю та послідовністю в межах відведеного часу та бюджету. Коректно підібране управління проектами дає такі переваги:

— визначення дій, що відповідатимуть цілям;

— підвищення продуктивності та якості роботи;

— встановлення бюджету та масштабів з самого початку;

— налагодження комунікацій між командами, замовниками та

— клієнтами;

— дотримання розкладу та витрат;

— підвищення шансів успішного завершення проекту;

— зменшення ризиків;

— забезпечення пріоритетного та ефективного використання ресурсів та багато іншого.

Крім того, користувачі можуть використовувати управління проектами для досягнення особистих цілей, поліпшення повсякденного життя, навчання та роботи. Хоча на ринку є різні рішення, популярні серед великих організацій, вони можуть бути занадто складними та неінтуїтивними для окремих користувачів.

1.2 Визначення та класифікація веб-додатків

Веб-додаток — це комп'ютерна програма, яка використовує веб-браузер для виконання певної функції.

Веб-додаток — це програма клієнт-сервер. Це означає, що він має клієнтську і серверну сторону. Термін "клієнт" тут відноситься до програми, яку особа використовує для запуску програми. Це частина клієнт–серверного середовища, де багато комп’ютерів обмінюються інформацією. Наприклад, у випадку з базою даних клієнтом є програма, за допомогою якої користувач вводить дані. Сервер — це програма, яка зберігає інформацію. Схема роботи такої програми зображена на рисунку 1.2.

Веб-додаток доступний за допомогою будь–якого веб-браузера. Його інтерфейс зазвичай створюється за допомогою таких мов, як HTML, CSS, Javascript, які підтримуються основними браузерами [2].

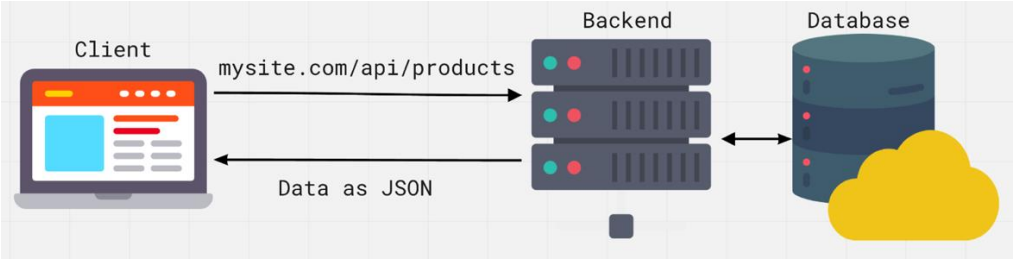

Рисунок 1.1 — Схема роботи програми клієнт-сервер

Типовий робочий процес веб-додатків виглядає так:

— користувач надсилає запит на веб-сервер через Інтернет, використовуючи веб-браузер або інтерфейс користувача програми;

— запит пересилається веб-сервером на відповідний сервер веб-додатків;

— після цього сервер веб-додатків виконує запитане завдання (наприклад, обробка нових даних), а потім генерує відповідні результати;

— потім результати надсилаються із запитаною інформацією із сервера веб-додатків на веб-сервер;

— потім веб-сервер відповідає клієнту, а запитувана інформація відображається користувачеві.

Веб-додаток має багато переваг, зокрема:

— вам не потрібно встановлювати його на жорсткий диск, тому це не спричиняє обмеження простору;

— він вимагає менше підтримки та обслуговування з боку бізнесу та нижчих технічних вимог до комп’ютера користувача;

— веб-додаток знижує витрати як для кінцевого користувача, так і для бізнесу;

— веб-програми завжди оновлюються, оскільки оновлення застосовуються централізовано;

— усі користувачі мають доступ до однієї версії, тому це усуває будь–які проблеми з сумісністю;

— ви можете отримати доступ до веб-програм будь–де за допомогою веб-браузера;

— поки браузер сумісний, веб-додатки можуть працювати на кількох платформах незалежно від операційної системи чи пристрою;

— веб-додатки знімають з розробника відповідальність за створення клієнта, сумісного з певним типом комп'ютера або конкретною операційною системою;

— веб-програми зменшують програмне піратство у веб-додатках на основі підписки.

Веб-додатки мають деякі переваги перед традиційними настільними або мобільними додатками.

Веб-додатки можуть працювати на будь-якій платформі, яка підтримує веб-браузер, без необхідності встановлення додаткового програмного забезпечення.

Веб-додатки можуть бути доступні з будь-якого місця та будь-якого пристрою, який має підключення до Інтернету, без необхідності синхронізації даних.

Веб-додатки можуть легко підтримувати велику кількість користувачів та запитів, розподіляючи навантаження між веб-серверами та використовуючи хмарні сервіси [3].

Веб-додатки можуть захищати дані користувачів та передавати їх захищеними протоколами, такими як HTTPS, а також використовувати різні методи аутентифікації та авторизації.

Веб-додатки можна класифікувати за різними критеріями.

За архітектурою веб-додатки можна поділити на три основні типи.

Клієнт-серверні — це традиційні веб-додатки, які складаються з двох частин: клієнтської, яка виконується в браузері користувача, і серверної, яка виконується на віддаленому комп’ютері. Клієнт і сервер обмінюються даними за допомогою запитів і відповідей через протокол HTTP. Прикладами таких веб-додатків є онлайн-магазини, соціальні мережі, електронна пошта тощо.

Односторінкові — це веб-додатки, які використовують технологію AJAX (Asynchronous JavaScript and XML) для динамічного оновлення вмісту сторінки

без перезавантаження браузера. Такі веб-додатки забезпечують швидку та зручну взаємодію з користувачем, схожу на роботу з настільними додатками. Прикладами таких веб-додатків є Google Maps, Gmail, Google Docs тощо.

Розподілені — це веб-додатки, які використовують веб-сервіси для забезпечення взаємодії між різними компонентами, які можуть бути розташовані на різних серверах, платформах або мережах. Такі веб-додатки дозволяють інтегрувати різноманітні джерела даних та функціональності в єдиний інтерфейс. Прикладами таких веб-додатків є веб-портали, веб-сервіси, веб-машини тощо [2].

За цим мовою програмування веб-додатки можна поділити на дві основні групи.

Скриптові — це веб-додатки, які використовують інтерпретовані мови програмування, такі як PHP, Python, Ruby, JavaScript тощо. Такі мови не потребують компіляції перед виконанням, а виконуються безпосередньо на сервері або в браузері. Скриптові мови зазвичай мають простий синтаксис, велику кількість бібліотек та фреймворків, а також гнучкість та швидкість розробки. Однак вони також мають недоліки, такі як низька продуктивність, високе споживання ресурсів, відсутність строгої типізації тощо.

Компільовані — це веб-додатки, які використовують компільовані мови програмування, такі як Java, C#, C++ тощо. Такі мови потребують компіляції перед виконанням, а виконуються на віртуальній машині або безпосередньо на процесорі. Компільовані мови зазвичай мають високу продуктивність, низьке споживання ресурсів, строгу типізацію та безпеку. Однак вони також мають недоліки, такі як складний синтаксис, мала кількість бібліотек та фреймворків, а також низька гнучкість та швидкість розробки.

За функціональністю веб-додатки можна поділити на безліч різних категорій, залежно від того, які задачі вони вирішують для користувачів. Наприклад, можна виділити наступні категорії веб-додатків.

Інформаційні — це веб-додатки, які надають користувачам доступ до різних джерел інформації, таких як новини, погода, енциклопедії, карти тощо. Прикладами таких веб-додатків є Wikipedia, Google News, Bing Weather тощо.

Освітні — це веб-додатки, які допомагають користувачам навчатися різним предметам, мовам, навичкам тощо. Прикладами таких веб-додатків є Coursera, Duolingo, Khan Academy тощо.

Розважальні — це веб-додатки, які забезпечують користувачам можливість переглядати, слухати, грати, спілкуватися та інші форми розваг. Прикладами таких веб-додатків є YouTube, Spotify, Netflix, Facebook тощо.

1.3 Цільова аудиторія веб-додатку для управління проектами і задачами

Одним з важливих аспектів розробки будь-якого веб-додатку є визначення його цільової аудиторії. Цільова аудиторія — це група людей, які мають спільні потреби, інтереси, проблеми або цілі, які можуть бути задоволені або вирішені за допомогою веб-додатку. Визначення цільової аудиторії допомагає розробникам краще зрозуміти, хто є їх потенційними користувачами, які їх очікування та потреби, як досягти їх та залучити їх до використання веб-додатку [4].

Веб-додаток для створення, організації та відстеження індивідуальних й командних задач і проектів має широку та різноманітну цільову аудиторію. Цей веб-додаток може бути корисним для будь-якої особи або організації, яка хоче підвищити свою продуктивність, ефективність та співпрацю за допомогою сучасних та зручних інструментів для управління задачами та проектами. Цільовою аудиторією веб-додатку є люди, які потребують планувати, організовувати та відстежувати свою індивідуальну або командну роботу, а також отримувати допомогу від штучного інтелекту у вигляді опису та рекомендацій для задач. Такі люди можуть належати до різних категорій. До цільової аудиторії цього веб-додатку можуть належати:

— студенти, викладачі та науковці, які займаються навчальними або науковими проектами, дослідженнями, публікаціями тощо, вони можуть

використовувати веб-додаток для планування свого навчального чи наукового процесу, розподілу завдань між учасниками проекту, контролю за термінами виконання, спілкування з колегами, отримання рекомендацій щодо написання тексту або коду, генерації графіки або іншого контенту тощо;

— професіонали, фрілансери, підприємці, менеджери, які працюють над різноманітними проектами, задачами, клієнтами, продуктами тощо, вони можуть використовувати веб-додаток для організації своєї роботи, встановлення пріоритетів, відстеження прогресу, звітування, співпраці з командою, клієнтами або партнерами, отримання рекомендацій щодо вдосконалення своїх продуктів, послуг, процесів тощо;

— особисті користувачі, які хочуть краще розпоряджатися своїм часом, ресурсами, цілями, інтересами тощо, вони можуть використовувати веб-додаток для планування свого дня, тижня, місяця, року, створення списків справ, побажань, ідей, натхнення тощо, відстеження своїх досягнень, успіхів, проблем, викликів тощо, отримання рекомендацій щодо покращення свого саморозвитку, самовдосконалення, самореалізації тощо;

— хобісти, фрілансери та волонтери, які хочуть реалізувати свої ідеї, захоплення та ініціативи, знаходити та залучати однодумців та співробітників, ділитися своїм досвідом та досягненнями, отримувати відгуки та поради;

— підприємства, стартапи, команди, організації, які хочуть підвищити свою продуктивність, співпрацю, інноваційність тощо, вони можуть використовувати веб-додаток для створення, організації та відстеження своїх командних проектів, задач, цілей, стратегій тощо, вони можуть також використовувати веб-додаток для комунікації, обміну інформацією, документами, ідеями, фідбеком тощо.

Веб-додаток дозволяє створювати різні дошки, списки, картки, які відображають структуру, статус, пріоритети, відповідальність, ресурси тощо. Веб-додаток також надає рекомендації щодо оптимізації робочих процесів, вирішення проблем, виявлення можливостей тощо.

Цільова аудиторія цього веб-додатку може мати різні вікові, освітні, професійні та культурні характеристики, але вони об’єднані спільною метою — покращити своє життя та роботу за допомогою веб-додатку для створення, організації та відстеження індивідуальних й командних задач і проектів.

1.4 Огляд існуючих веб-додатків для управління проектами і задачами

Управління проектами є важливою частиною будь-якої організації. Воно допомагає ефективно і результативно реалізувати проекти, незалежно від їх розміру, складності або галузі.

Існує широкий спектр систем управління проектами, які пропонують різні функції та можливості. Деякі системи призначені для малого бізнесу, інші — для великих підприємств. Деякі системи орієнтовані на конкретні галузі, інші — на загальні застосування.

Веб-додатки — це найпопулярніший тип систем управління проектами. Вони доступні з будь-якого пристрою з доступом до Інтернету.

Однією з найважливіших та найпоширеніших категорій веб-додатків є організаційні, які допомагають користувачам планувати, організовувати, керувати, співпрацювати та контролювати різні процеси, проекти, задачі, ресурси тощо. Такі веб-додатки можуть бути корисні як для індивідуальних, так і для командних цілей, а також для різних сфер діяльності, таких як освіта, бізнес, наука, мистецтво тощо. Такі веб-додатки можуть мати різні функції та можливості.

Створення та редагування задач. Користувачі можуть створювати та редагувати задачі, надавати їм назви, описи, терміни, пріоритети, відповідальних, мітки, чек-листи, коментарі, прикріплення тощо.

Організація та візуалізація задач. Користувачі можуть організовувати задачі за різними критеріями, такими як проекти, дошки, списки, колонки, рядки, карточки, календарі, діаграми Ганта, діаграми Канбан тощо. Користувачі також можуть візуалізувати задачі за допомогою різних типів графіків, таких як секторні, лінійні, стовпчикові, точкові, площинні тощо.

Відстеження та звітування про задачі. Користувачі можуть відстежувати статус, прогрес, результати, проблеми, ризики, зміни, витрати, час тощо, пов’язані з задачами. Користувачі також можуть генерувати та експортувати різні види звітів, таких як зведені, детальні, порівняльні, аналітичні тощо.

Співпраця та комунікація. Користувачі можуть співпрацювати та комунікувати з іншими учасниками задач і проектів, ділитися інформацією, обмінюватися думками, отримувати та надавати зворотний зв’язок, сповіщати та бути сповіщеними про різні події тощо.

Для того, щоб визначити доцільність, актуальність та перспективність розробки власного веб-додатку для створення, організації та відстеження індивідуальних й командних задач і проектів, необхідно провести порівняльний аналіз існуючих веб-додатків, які надають схожі або альтернативні послуги або функції. Такий аналіз дозволить виявити сильні та слабкі сторони, переваги та недоліки, можливості та загрози, сильні та слабкі сторони, ніші та тренди тощо, які мають різні веб-додатки, а також визначити свою унікальну торгову пропозицію, диференціацію, цільову аудиторію, цінову політику, стратегію маркетингу тощо для свого веб-додатку [5].

Сучасні веб-додатки для створення, організації та відстеження індивідуальних та командних задач і проектів мають різні функціональні можливості, дизайн, інтерфейс, цільову аудиторію та вартість. На ринку представлено широкий спектр систем управління проектами. До найпопулярніших та найкращих веб-додатків цього типу належать:

— Trello — проста і зручна система управління задачами, заснована на канбан-методології;

— Asana — потужний і гнучкий веб-додаток для управління проектами, який підтримує різні методи управління проектами;

— Monday.com — веб-додаток для управління проектами, який відрізняється візуальним інтерфейсом і широким набором шаблонів;

— Jira — веб-додаток для управління проектами, який призначений для розробки програмного забезпечення.

1.4.1 Jira

Jira, розроблена компанією Atlassian, є однією з провідних систем управління проектами та задачами в галузі розробки програмного забезпечення. Вона спеціалізується на ефективному трекінгу та управлінні завданнями великих і складних проектів [5].

Ключові Можливості:

— трекінг завдань**,** Jira дозволяє створювати та відстежувати завдання в реальному часі, кожне завдання може мати різні параметри, включаючи статус, призначеного користувача та прикріплені файли;

— гнучкість конфігурації, система легко налаштовується під конкретні потреби команди, можливості конфігурації включають в себе створення власних полів, робочих процесів та доналаштування прав доступу;

— інтеграція з іншими інструментами, Jira інтегрується з багатьма іншими інструментами розробки, такими як Bitbucket, Confluence та ряд інших, це дозволяє забезпечити поток інформації між різними етапами розробки;

— подробні звіти та аналітика, платформа надає засоби для створення різноманітних звітів та аналітики, що допомагає командам отримувати інсайти в продуктивність та прогрес проектів.

Початково створена система Jira була задумана як звичайний інструмент для відстеження задач і помилок. Зараз цей інструмент виросли у потужний засіб для управління проєктами, який включає в себе різноманітні вбудовані системи. Jira користується великою популярністю серед великих компаній і займає топ-3 світових рішень для управління проєктами.

Зате використання цього сервісу досить зручне лише для технічних команд, які вже мають досвід роботи з ним та знають вбудовані інструменти. Інтерфейс та програмне забезпечення Jira абсолютно не призначені для нових користувачів, що не мали досвіду з даним веб-застосунком. Для цієї аудиторії програма видається незрозумілою та надто складною. Інтерфейс програми зображений на рисунку 1.2.

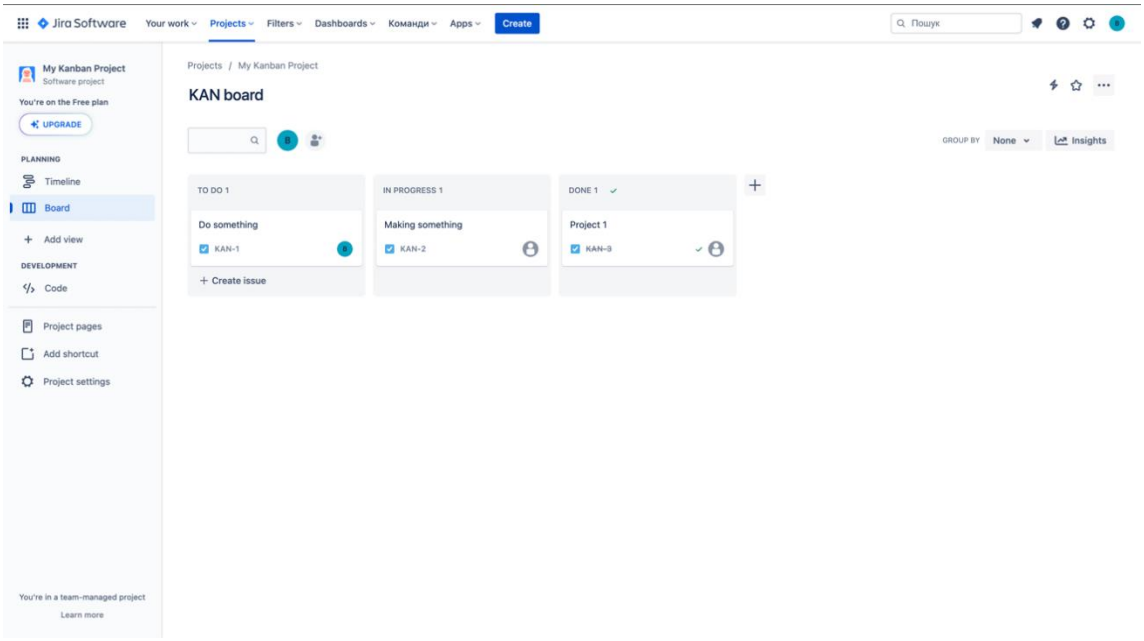

Рисунок 1.2 — Інтерфейс програми Jira

Крім того, користувачі помічають стабільне сповільнення швидкості роботи системи при щоденному використанні Jira. Це стало наслідком перевищеної кількості сторінок, запитів, інструментів, налаштувань, авторизацій, прав та проєктів, що генерують звіти. За цими обставинами розчаровані користувачі входять в систему рідше і, в кінці кінців, припиняють використання програми або переходять на інший аналогічний інструмент.

Також, через велику кількість функцій, важко знаходити необхідну інформацію для користувача. Нові поля форми, які не використовуються, швидко перетворюються на непотрібне сміття, яке ускладнює вже завантажений інтерфейс.

Jira виявляється потужним інструментом для команд, які шукають ефективний спосіб управління завданнями та проектами. Її гнучкість та інтеграційні можливості роблять її важливим інструментом для розробки програмного забезпечення та не тільки. Jira — це потужний і гнучкий веб-додаток для управління проєктами, який може використовуватися для різних цілей. Він пропонує широкий спектр функцій, які допомагають користувачам ефективно управляти проєктами.

Плюси Jira:

— широкий спектр функцій;

— гнучкість;

— комунікація;

— інтеграція.

Мінуси Jira:

— може бути складним у використанні;

— не завжди відповідає конкретним потребам організації.

1.4.2 Asana

Asana є популярною системою управління проектами, призначеною для полегшення співпраці та організації завдань в командах. Розроблена командою Asana Inc., платформа визначається своєю інтуїтивністю та акцентом на зручність використання.

Asana, як система управління, була створена з метою планування та координації задач, їх аналізу та поділу на окремі підзадачі, а також призначення цих завдань учасникам команди. Це рішення формує своєрідну екосистему для кожного проєкту і дозволяє ефективно адмініструвати всі завдання. Описана структура розподілу проєктів входить до однієї з методологій розробки програмного забезпечення, дозволяючи кожному учаснику команди чітко визначити свою зону відповідальності. Користувач може створити власний особистий простір, де він може переглядати всі свої завдання та проєкти. Таким чином, система гарантує ефективну роботу, що виконується чітко, злагоджено та безперебійно.

Вбудована система зв'язку дозволяє членам команди легко обмінюватися інформацією, даними та повідомляти про статус роботи.

Asana визначається як одна з еталонних систем управління проєктами, оскільки вона проста у розумінні, інтуїтивно зрозуміла навіть для новачків у цьому типі систем, і швидка, не завантажена зайвими інструментами. Ця система близька до нашої мети, маючи всі переваги, що були вище перераховані. Проте

варто зауважити, що вона все ще може бути занадто об'ємною для особистого використання. Інтерфейс програми Asana зображено на рисунку 1.3.

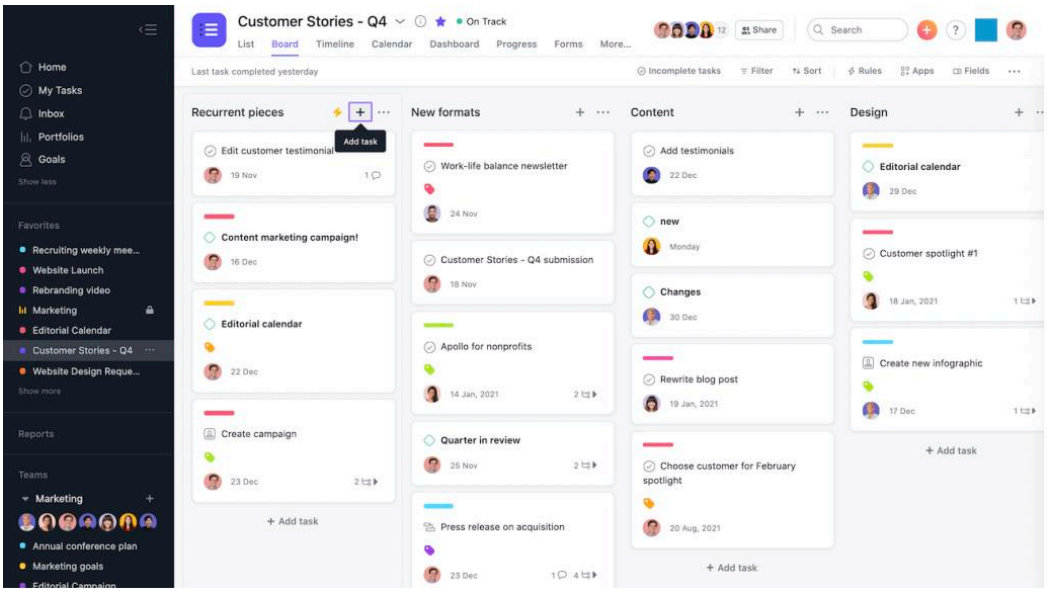

Рисунок 1.3 — Інтерфейс програми Asana

Основні функції Asana:

— планування дозволяє користувачам створювати плани проєктів і визначати їх цілі, завдання та ресурси;

— виконання дозволяє користувачам відстежувати прогрес проєктів, розподіляти завдання та контролювати витрати;

— контроль дозволяє користувачам виявляти і вирішувати проблеми, а також стежити за ризиками;

— звітність дозволяє користувачам створювати звіти про проєкти для відстеження їх прогресу та ефективності.

Особливості Asana:

— простота використання, Asana має простий і інтуїтивно зрозумілий інтерфейс, який легко освоїти;

— гнучкість, Asana є гнучкою системою, яка може бути налаштована відповідно до потреб організаці;

— комунікація, Asana має вбудовані інструменти для комунікації між учасниками проєктів, такі як коментарі, примітки та форуми;

— інтеграція, Asana може бути інтегрована з іншими системами, такими як системи управління базами даних, системи управління контентом та системи управління проектами.

Asana — це простий у використанні і гнучкий веб-додаток для управління проєктами, який може використовуватися для різних цілей. Він пропонує широкий спектр функцій, які допомагають користувачам ефективно управляти проєктами.

Плюси Asana:

— гнучкість;

— простота використання;

— комунікація;

— інтеграція.

Мінуси Asana:

— не може бути достатнім для складних проєктів;

— не завжди відповідає конкретним потребам організації.

1.4.3 Trello

Trello — це веб-додаток для управління задачами, заснований на канбан-методології. Він призначений для широкого спектру цілей, включаючи управління проектами в галузі маркетингу, продажів, виробництва, розробки програмного забезпечення та інших. Trello — це інтерактивна та візуальна платформа для управління завданнями, яка використовує концепцію "дошки", "списків" та "карток" для організації робочих процесів. Розроблена командою Fog Creek Software та випущена у 2011 році, платформа набула великої популярності завдяки своїй простоті та гнучкості [5]. Інтерфейс програми Trello зображено на рисунку 1.4.

Основні функції Trello:

— планування дозволяє користувачам створювати плани проєктів і визначати їх цілі, завдання та ресурси;

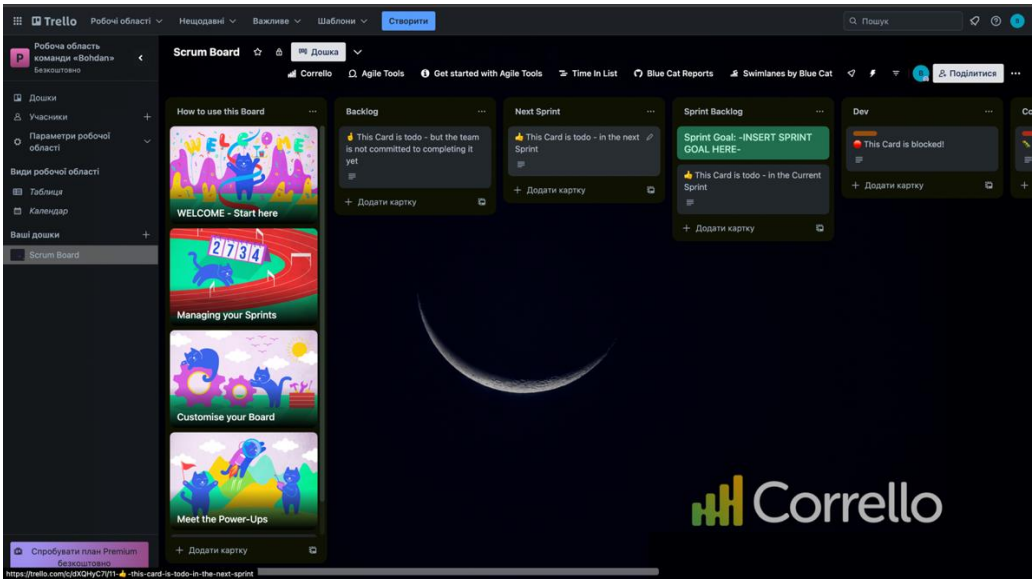

Рисунок 1.4 — Інтерфейс програми Trello

— виконання дозволяє користувачам відстежувати прогрес проєктів, розподіляти завдання та контролювати витрати;

— контроль дозволяє користувачам виявляти і вирішувати проблеми, а також стежити за ризиками;

— звітність дозволяє користувачам створювати звіти про проєкти для відстеження їх прогресу та ефективності.

Особливості Trello:

— простота використання, Trello має простий і інтуїтивно зрозумілий інтерфейс, який легко освоїти;

— гнучкість, Trello є гнучкою системою, яка може бути налаштована відповідно до потреб організації;

— комунікація, Trello має вбудовані інструменти для комунікації між учасниками проєктів, такі як коментарі, примітки та форуми;

— інтеграція, Trello може бути інтегрована з іншими системами, такими як системи управління базами даних, системи управління контентом та системи управління проектами.

Trello став важливим інструментом для тих, хто шукає простий та ефективний спосіб організації завдань та проектів.

Візуальний підхід та простота використання роблять його популярним серед широкого кола користувачів.

Trello це простий у використанні і гнучкий веб-додаток для управління задачами, який може використовуватися для різних цілей. Він пропонує широкий спектр функцій, які допомагають користувачам ефективно управляти задачами.

Плюси Trello:

— простота використання;

— гнучкість;

— комунікація;

— інтеграція.

Мінуси Trello:

— не може бути достатнім для складних проєктів;

— не завжди відповідає конкретним потребам організації;

Trello заснований на канбан-методології, яка є простим і ефективним способом управління задачами. Канбан-дошка складається з трьох колонок: Виконується, Виконується, Виконано. Завдання переміщуються по дошці в міру їх виконання.

Trello пропонує широкий спектр функцій, які допомагають користувачам ефективно управляти задачами. До них відносяться:

— дозволяє створювати дошки для різних проєктів або завдань;

— дозволяє створювати завдання і додавати до них опис, терміни, пріоритети та інші метадані;

— дозволяє додавати до завдань файли та посилання;

— дозволяє коментувати завдання;

— дозволяє налаштовувати права доступу до завдань;

— дозволяє інтегруватися з іншими системами, такими як Google Drive, Dropbox, Slack та іншими.

Trello — це популярний веб-додаток для управління задачами, який використовується компаніями різного розміру і галузі діяльності. Він є хорошим

вибором для команд, які потребують простого і гнучкого рішення для управління задачами.

1.4.4 Monday.com

Monday.com є веб-платформою для управління проектами та завданнями, створеною для полегшення співпраці в командах та покращення організації робочих процесів. Заснована в 2014 році, ця платформа намагається зробити управління проектами більш гнучким та інтуїтивно зрозумілим.

Monday.com — це веб-платформа для управління проєктами, яка пропонує широкий спектр функцій і можливостей. Він призначений для широкого спектру цілей, включаючи управління проектами в галузі маркетингу, продажів, виробництва, розробки програмного забезпечення та інших.

Основні функції Monday.com:

— планування дозволяє користувачам створювати плани проєктів і визначати їх цілі, завдання та ресурси;

— виконання дозволяє користувачам відстежувати прогрес проєктів, розподіляти завдання та контролювати витрати;

— контроль дозволяє користувачам виявляти і вирішувати проблеми, а також стежити за ризиками;

— звітність дозволяє користувачам створювати звіти про проєкти для відстеження їх прогресу та ефективності.

Особливості Monday.com:

— має візуальний інтерфейс, який робить його простим і інтуїтивно зрозумілим у використанні;

— є гнучкою системою, яка може бути налаштована відповідно до потреб організації;

— має вбудовані інструменти для комунікації між учасниками проєктів, такі як коментарі, примітки та форуми;

— може бути інтегрована з іншими системами, такими як системи управління базами даних, системи управління контентом та системи управління проектами.

Monday.com — це потужний і гнучкий веб-додаток для управління проєктами, який може використовуватися для різних цілей. Він пропонує широкий спектр функцій, які допомагають користувачам ефективно управляти проєктами. Інтерфейс програми Monday.com зображено на рисунку 1.5.

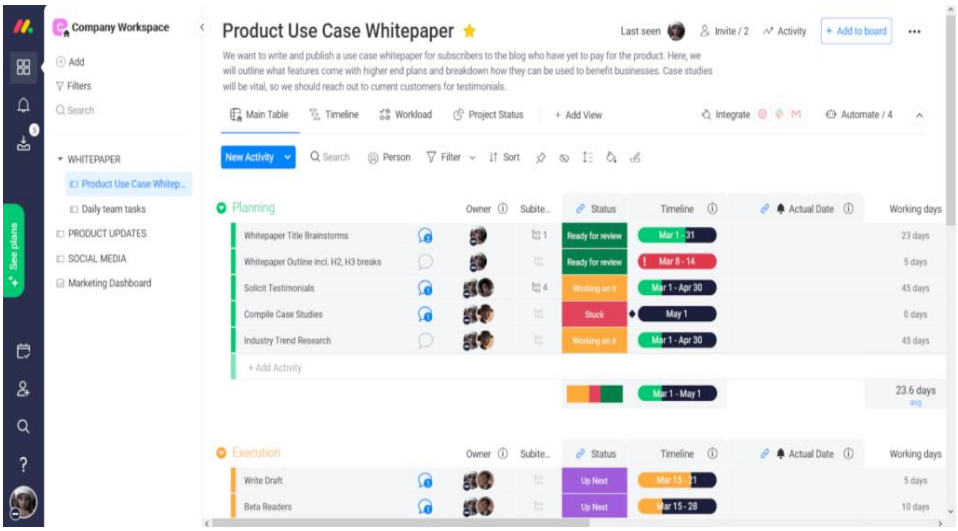

Рисунок 1.5 — Інтерфейс програми Monday.com

Плюси Monday.com:

— візуальний інтерфейс;

— гнучкість;

— комунікація;

— інтеграція.

Мінуси Monday.com:

— може бути складним у використанні;

— не завжди відповідає конкретним потребам організації.

Monday.com пропонує широкий спектр функцій, які допомагають користувачам ефективно управляти проєктами. До них відносяться:

— дозволяє створювати дошки, таблиці, канбани та інші види робочих просторів;

— дозволяє створювати завдання і додавати до них опис, терміни, пріоритети та інші метадані;

— дозволяє додавати до завдань файли та посилання;

— дозволяє коментувати завдання;

— дозволяє налаштовувати права доступу до завдань;

— дозволяє інтегруватися з іншими системами, такими як Google Drive, Dropbox, Slack та іншими.

Monday.com — це популярний веб-додаток для управління проєктами, який використовується компаніями різного розміру і галузі діяльності. Він є хорошим вибором для команд, які потребують потужного і гнучкого рішення для управління проєктами.

Monday.com вирізняється своєю візуальною привабливістю, гнучкістю та рядом функцій, що роблять його привабливим для команд, що шукають зручний та ефективний інструмент для управління проектами та завданнями.

1.4.5 Порівняльна характеристика

Отже, на основі аналізу популярних додатків для управління проектами, порівняєм дані додатки. Порівняльні характеристики зображені в таблиці 1.1.

Таблиця 1.1 — Порівняльна характеристика додатків

Критерій	Jira	Trello	Asana	Monday.com
Інтерфейс	Канбан-стильний інструментарій з елементами перетягування	Налаштовуваний інструментарій з різними видами та фільтрами	Список, дошка, часова шкала та календарні види з елементами перетягування	Дошка, часова шкала, календар, діаграма та картографічні види з елементами перетягування

Продовження таблиці 1.1

Автоматиза-ція	Правила-базовані тригери та дії з Butler	Автоматизація робочих процесів з опціями без коду або з мінімальним кодом	Правила-базовані тригери та дії з попередньо визначеними або користувацькими шаблонами	Автоматизація робочих процесів з понад 250 готовими рецептами або користувацькими опціями
Співпраця	Коментарі, згадування, вкладення, списки перевірки, мітки, терміни виконання та сповіщення	Коментарі, згадування, вкладення, підзадачі, статуси, пріоритети та сповіщення	Коментарі, згадування, вкладення, підзадачі, залежності, етапи та сповіщення	Коментарі, згадування, вкладення, підпункти, залежності, статуси та сповіщення
Інтеграція	Понад 200 підключень з різними додатками та сервісами	Понад 3000 додатків та інтеграцій з різними платформами та інструментами	Понад 100 інтеграцій з різними додатками та сервісами	Понад 40 інтеграцій з різними додатками та сервісами
Звітність	Обмежені можливості звітності та аналітики	Розширені можливості звітності та аналітики з діаграмами	Основні можливості звітності та аналітики з діаграмами	Розширені можливості звітності та аналітики з діаграмами
Цільва аудиторія	Загальне управління проектами	Розробка програмного забезпечення	Загальне управління проектами	Загальне управління проектами
Тип платформи	Візуальна платформа для управління завданнями	Комплексна система управління проектами	Універсальна система для управління проектами	Платформа для управління проектами та завданнями

Закінчення таблиці 1.1

Ціноутво-рення	Безкоштовний план з основними функціями та необмеженими дошками, картками та учасниками; платні плани починаються від 5 доларів США на користувача в місяць	Безкоштовний план з основними функціями та до 10 користувачів; платні плани починаються від 7 доларів США на користувача в місяць	Безкоштовний план з основними функціями та до 15 користувачів; платні плани починаються від 10,99 доларів США на користувача в місяць	Безкоштовний план з основними функціями та до 2 користувачів; платні плани починаються від 8 доларів США на користувача в місяць

2 ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ

2.1 Огляд методологій керування проектами

Проєктні методи роботи стають дедалі популярнішими серед компаній. Будь-яке велике робоче завдання можна розглядати як проєкт, бо воно має мету, строки і ресурси. Проєкт потребує вмілого управління. Управління проєктами, або проджект-менеджмент, широко використовують у своїй роботі IT-компанії, креативні та вебагенції, міжнародні корпорації і навіть державні інституції [6].
Методологія управління будь-яким проєктом — це уніфікація його проведення. На сьогодні існують певні шаблони для виконання проєктів, які допоможуть уникнути багатьох поточних проблем та отримати в результаті готовий продукт. Однак слід пам’ятати, що кожен проєкт — унікальний, тому методологія не є універсальним рішенням, і думати таки доведеться.

Багато організацій зіштовхуються з труднощами у виборі найкращої методології управління проєктами. Вибір найкращого методу з багатьох підходів буває досить складним. Відповідальність за реалізацію проєкту, а потім часто і супроводження подальшого його існування, залежить від декількох осіб — замовників, підрядників, виконавців, але в першу чергу від керівника проєкту, людини, яка на своїх плечах буде нести весь цей тягар. Тому навчання і розвиток для керівника проєкту є обов’язковим і життєво-необхідним [7].

Знання, які необхідні керівнику проєкту, стосуються не тільки методів управління, а й інструментів, за допомогою яких можна вирішувати конкретні нагальні питання і доводити справу до фінішу з очікуваним прогнозованим результатом.
На сьогодні розроблено величезну кількість методологій управління проєктами. Деякі використовуються лише в одній компанії, є й глобальні, які застосовує бізнес різного спрямування в усіх куточках світу.

Методологія управління проєктами — це практика або техніка, яка допоможе вам успішно керувати проєктом і виконувати його. Вона описує, як взятися за проєкт і як виконати покрокові інструкції щодо його завершення. Крім того, вона обмежує життєвий цикл проєкту правильно структурованими

етапами. Займатись веденням проєкту без методології — все одно що їхати на авто по невідомій місцевості без GPS-навігатора або мапи, ймовірність потрапити за адресою ще й вчасно достатньо низька. Методології не лише забезпечують основу для планування та успішного виконання проєктів, а й покращують комунікацію в команді, допомагають керувати ризиками, полегшують вирішення проблем, закривають низку інших важливих питань в проджект-менеджменті. В статті розглянемо які методології найчастіше застосовують PM-и, їх особливості та відмінності між собою.

Методологія в проджект-менеджменті — це набір певних правил, умов, принципів, дій, яких необхідно дотримуватись задля якісного закриття проєкту. Свого роду це стандартизація підходу до роботи. Наприклад методологія може визначати з яких кроків складаються робочі процеси, як контролюється виконання задач та відбувається взаємодія між членами команди, приймаються рішення, включати інші стандарти ведення проєкту [7].

Використання вибраної методологі ї дозволяє:

— мати спільне для усієї команди розуміння процесів, обов’язків, ролей;

— точніше визначати ресурси та прогнозувати кінцевий результат;

— підвищити продуктивність співробітників;

— швидше та краще закривати робочі питання;

— зростати професійно;

— покращити загальну атмосферу в команді та згуртувати співробітників;

— не повторювати помилок, яких команда припускалась в минулих проєктах.

Підходів до керування проектами дуже багато, тому перш ніж зробити вибір, потрібно мати уявлення про найбільш використовувані. При цьому потрібно розуміти переваги та недоліки кожного з них.

Важливо розуміти, кожна методологія керування проектами пропонує різні стратегії, сприяють успішної реалізації проекту. Ретельно підібрана методологія враховує особливості проекту та дає правильні принципи керування,

командної роботи, інструкції з контролю, перевірки та оцінки результату та багато інших переваг.

Як і будь-яка бізнес-модель чи підхід, методології керування проектами мають низку переваг та недоліків. Деякі методології спрямовані на швидкість реалізації проекту. Інші більше орієнтуються на охоплення складових проекту чи керування співробітництвом.

Розглянемо декілька підходів та методологій:

— Waterfall;
— Agile;
— Scrum;
— Kanban.

2.1.1 Waterfall

Водоспадна модель є однією з найдавніших та найбільш класичних методологій керування проектами. Її запропонував Вінстон У. Ройс у 1970 році як відповідь на швидке зростання галузі розробки програмного забезпечення. Модель характеризується послідовним виконанням етапів робіт з попереднім розподілом етапів на завдання. Вона вважається найпростішою методологією для розуміння та оптимальною точкою відліку для вивчення методологій керування проектами.

Водоспадна модель була основним підходом до керування IT-проектами протягом декількох десятиліть, але також застосовувалася в багатьох інших сферах, таких як виробництво або будівництво. Сутність водоспадної моделі полягає в попередньому плануванні етапів робіт з наступним формуванням підходів, рішень та етапів реалізації [6].

Етапи робіт у водоспадній моделі виконуються послідовно. Початок етапу 2 можливий тільки після закінчення етапу 1. Для проекта розробки програмного забезпечення етапи зображені на рисунку 2.1.

Рисунок 2.1 — Етапи робіт у водоспадній моделі

Етапи розробки програмного забезпечення З іншого боку, водоспад вимагає сформувати список цілей та досягти їх. Тому цей метод підходить для сфер, де продукти потребують точних та детальних інструкцій.

Окрім простоти, головна перевага водоспадної моделі полягає в тому, що майже будь-який план з попередніх проектів може бути вдосконалений і використаний знову з невеликими змінами. Крім того, її послідовна природа гарантує, що ви виконуєте всі вимоги з самого початку — етап може бути закінчений лише тоді, коли виконано всі вимоги. І так для кожного етапу проекту.

Ця методологія керування проектами значною мірою залежить від документації та записів у процесі розробки. Таким чином, цей метод також спрямований на вирішення проблем, пов’язаних з унікальними знаннями працівників. Наприклад, якщо старий співробітник пішов, новий може продовжити роботу відразу та без будь-яких проблем завдяки наявній документації. Ще однією перевагою методу є підвищення нагляду та контролю на кожному етапі.

Але як тільки етапи проекту визначені, методологія вимагає, щоб вони не змінювалися. Таким чином, метод виявляється “негнучким” та має серйозні обмеження. Якщо обсяг та масштаб проекту змінюються, план не може адаптуватися до такої зміни.

Деякі критики стверджують, що зменшення рівня гнучкості та креативності робить цю модель застарілою порівняно з сучасними стандартами. Ще один недолік полягає в тому, що метод не враховує зворотний зв’язок від зацікавлених сторін, який можна використовувати для покращення результатів проекту.

Незважаючи на свої слабкі сторони, основною перевагою методології є її підхід до планування проекту. Цей підхід легко пристосовується до будь-якого процесу розробки, що включає великі і складні завдання. Наприклад: промислове виробництво, будівництво, розробка нових продуктів тощо. Також водоспадна модель сприяє майбутній реалізації подібних проектів: проект будівництва стандартної будівлі може бути легко адаптований до нового об’єкту будівництва.

Методологія добре підходить для складних проектів зі суворими термінами реалізації. А також для проектів, які були раніше реалізовані з низьким рівнем ризиків і помилок.

2.1.2 Agile

Одним із способів уникнути слабких місць водоспадної моделі є методологія Agile. Ця методологія керування проектами була сформульована в 2001 році в «Agile Manifesto». Документ, написаний лідерами з розробки програмного забезпечення, пропонував альтернативу, яка б не обтяжувала проекти надмірною документацією та бюрократією.

Методологія зосереджується на проектах, які вимагають швидкості та адаптації. Гнучке керування проектами базується на короткострокових «спринтах», а також високому рівні взаємодії та співпраці.

Методологія Agile ґрунтується на чотирьох ключових цінностях, які зображені на рисунку 2.2.

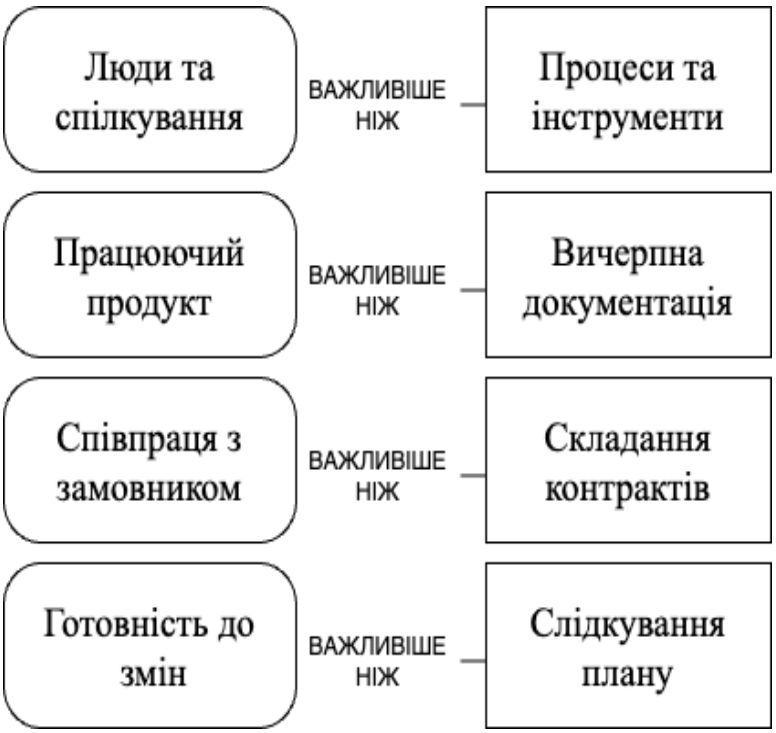

Рисунок 2.2 — Цінності Agile

Причина, чому Agile відкрив нові можливості в керуванні проектами, у її інноваційній філософії. Проекти, що гнучко керувалися, змогли досягти успіху, завдяки адаптивності, співпраці з клієнтами та наданню цінності.

Чесно кажучи, Agile це не методологія. Це ідеологія та підхід, який реалізується у таких методах як: Scrum, Adaptive Project Framework (APF) та Extreme Programming.

Основна ідея полягає в тому, що проект може змінюватися і розвиватися з часом, тому продукт, рішення або результат проекту також можуть змінюватися разом з ним.

У маніфесті Agile визначено дванадцять основних принципів:

— найважливішим для нас є задоволення потреб замовника шляхом регулярної та ранньої доставки цінного програмного забезпечення;

— зміна вимог приймається навіть на пізніх етапах розробки. Agile процеси дозволяють використовувати зміни для надання замовнику конкурентної переваги;

— працюючий продукт слід випускати якомога частіше, з інтервалами від кількох тижнів до кількох місяців;

— протягом всього проекту розробники та представники бізнесу мають щоденно працювати разом;

— над проектом мають працювати мотивовані професіонали. Щоб роботу було зроблено, створіть умови, забезпечте підтримку та повністю довіртеся їм;

— пряме спілкування є найбільш практичним та ефективним способом обміну інформацією;

— діючий продукт — основний показник прогресу;

— процес розробки повинен бути стабільним;

— якість та технічна досконалість у пріоритеті, завдяки чому проект стає гнучким;

— команда не потребує мікро менеджменту та здебільшого само організована;

— всі учасники процесу аналізують варіанти покращення ефективності проекту та змінюють порядок своєї роботи відповідно цьому.

У порівнянні з водопадною моделлю методологія Agile забезпечує більш високу гнучкість. Однак реалізація сильно змінюватиметься залежно від потреб проекту. Основний плюс методології керування проектами полягає в тому, що вона дозволяє здійснювати співпрацю, зміну та переоцінку у процесі розробки. Тому вона дуже популярна у світі інформаційних технологій. Етапи методології Agile зображено на рисунку 2.3.

Методологія підтримує безперервне вдосконалення, залишаючи простір експериментів. Таким чином, вона неймовірно добре працює для проектів, які потребують високого рівня інновацій та креативності. Інші його переваги включають швидкий цикл, постійний зворотний зв’язок та підвищену продуктивність. І навпаки, деякі з недоліків – відсутність фіксованих планів та оцінок, труднощі фінансового керування та збої у плануванні. І хоча деякі віддають перевагу інтерактивності, в інших може не вистачити часу на постійну

співпрацю. Більше того, ця методологія керування проектами не спирається на передбачуваність, що означає, що витрати, зусилля та час не так просто оцінити, порівняно з іншими методологіями [7].

Рисунок 2.3 — Етапи методології Agile

Гнучкість Agile гарантує, що ви зможете впровадити цю методологію у різноманітні проекти, галузі та продукти. Він неймовірно добре підходить для проектів із значним рівнем невизначеності та взаємодії. Agile може сприяти швидкому виробництву при розширеному співробітництві, і він уважно ставиться до зростання попиту та еволюції ринку.

2.1.3 Scrum

Скрам вважається найпростішим і найбільш поширеним методом, пов’язаним з Agile. Він ґрунтується на спринтах, які тривають від двох тижнів до тридцяти днів. Цей підхід допомагає встановити пріоритети завдань, акцентуючи на команді. Під час спринтів учасники команди проводять 15-хвилинні зустрічі, щоб оцінити свої досягнення. Ось як відбуваються спринти:

— команда використовує дошку для відстеження списку завдань;

— завдання поділені на кілька списків;

— люди можуть брати завдання з дошки та виконувати їх;

— кожне завершене завдання перевіряється на якість;

— після того, як власник продукту завдання позначається як готова.

Мета — запобігти перевтомі та виснаженню. Методологія Scrum спрямована на стимулювання гнучкості, креативності та надання високоякісних результатів. На відміну від керівника проекту, цей підхід використовується поняття «scrum master». Ця роль полягає в тому, щоб усунути будь-яку перешкоду і тим самим підвищити продуктивність для самоврядної команди. Scrum запозичує багато процесів і принципів Agile, але використовує особливий набір тактик і цінностей. П’ять основних цінностей Scrum, які фокусуються на людині та команді зображені на рисунку 2.4.

Рисунок 2.4 — Цінності Scrum

Відповідно до цих принципів, методологія Scrum спрямована на сприяння співпраці, успішній реалізації та стійкості складних продуктів. І так само, як інші підходи Agile, вона заохочує ітеративний розвиток.

Орієнтований на спринт робочий процес Scrum полегшує комунікації та керування. Протягом певного періоду спринту команда може поетапно аналізувати свої цілі. Цей підхід фокусується на практичних завданнях, швидкій співпраці та ефективної координації. Це одна з найлегших методик із чітким визначенням обов’язків та ролей.

Підхід ґрунтується на постійному вдосконаленні та оптимізації. Однак гнучкість, що він забезпечує, також є серйозним обмеженням. Наприклад, самоврядна Scrum-команда може бути не в змозі виконати вимоги організацій та галузей, які вимагають фіксованого обсягу, встановлених бюджетів, суворих термінів та інших серйозних обмежень. Ось чому методологія краще працює у складі гібридних підходів. Багато організацій та агентств використовують методи самоврядування та оцінки, що надаються Scrum, у поєднанні з іншими підходами.

Теоретично Scrum був би корисним для невеликих команд (десять чи менше), які вимагають високого рівня гнучкості. Цей підхід найкраще підходить для галузей з високим рівнем невизначеності, що змінюється навколишнім середовищем та тих, які щоразу вимагають абсолютно нових продуктів чи рішень. Деякі приклади, де Scrum може бути корисним поза розробкою програмного забезпечення – це маркетинг, юриспруденція, реалізація стратегії та дизайн продукту. Однак галузі, які покладаються на повторення та передбачуваність, повинні використовувати інший метод.

2.1.4 Kanban

Одна з головних особливостей Kanban — це його велика увага до візуалізації у процесі розробки. Цей термін походить з 1940-х років і перекладається як «сигнальна карта». Спочатку карти Kanban застосовувалися Toyota для оптимізації ресурсів та підвищення ефективності. Зараз Kanban використовує методи віртуальної візуалізації за допомогою систем керування та інших інструментів керування проектами.

Методологія керування проектами Kanban спрямовано візуалізацію робочого процесу виявлення вузьких місць і підвищення продуктивності [6]. Прогрес відображається візуально за допомогою різних сигналів.

Дошка Kanban (цифрова чи фізична) показує етапи розробки чи керування. Модель дошки Kanban показано на рисунку 2.5.

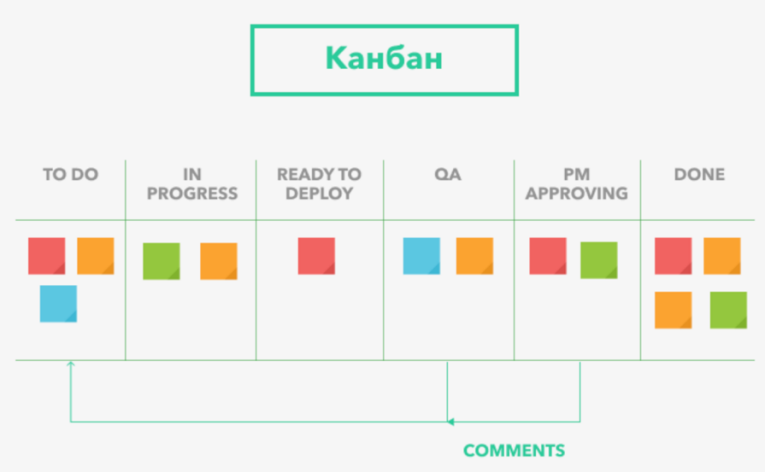

Рисунок 2.5 — Модель дошки Kanban

Карти Kanban видаватися кожному за завдання чи предмета. Вони можуть відслідковувати прогрес і співробітництво та надаючи дані (керування крайніми термінами, статус тощо).

Доріжки мають особливу форму відстеження та класифікації завдань. Зазвичай вони мають горизонтальне планування та потоки.

Окрім візуалізації, Kanban також слідує іншим принципам, таким як керування потоками, цикли зворотного зв'язку, спільне поліпшення тощо. Крім цих принципів, Kanban не має фіксованого набору правил і етапів, або запропонованих ролей.

Методологія керування проектами сприяє візуальній залученості. Це може усунути проблеми, пов'язані з керуванням робочим навантаженням та розміщенням пріоритетів. Метод призводить до підвищення продуктивності, оскільки дозволяє керівникам ефективніше контролювати робоче навантаження. Понад те, методологія та її інструменти прості розуміння. Це також знижує рівень незавершеного виробництва та призводить до оптимізованого та більш швидкого робочого процесу. Команди, що використовують Kanban, також можуть адаптуватися до змін набагато швидше. Що стосується мінусів, то у команди можуть виникнути проблеми з навігацією та обслуговуванням дошки Kanban та інших інструментів. При поганій обробці сигнали візуалізації можуть призвести до надмірного ускладнення та безладного робочого процесу.

Команди та проекти, які забезпечують стабільний результат або працюють у стислий термін, можуть використовувати Kanban для підвищення своєї ефективності. Також методологія зручна для реалізації типових проектів з передбачуваними етапами робіт. Крім цього, метод може принести користь організації, яка значною мірою залежить від виконання обсягу різношерстих завдань. Його можна використовувати для моніторингу та вимірювання технічної роботи, термінів виконання, термінових завдань, а також роботи, що повторюється (наприклад, нарад, звітів тощо). До речі, дошки канбан дуже зручні для керування операційними процесами.

2.2 Представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування, формування діаграми класів.

Статична структура моделі системи описує, які елементи складають систему, як вони пов’язані між собою та які властивості та операції вони мають. В термінології класів об’єктно-орієнтованого програмування, статична структура моделі системи представляється за допомогою діаграми класів.

Діаграма класів — це графічне представлення класів, їх атрибутів, методів та відношень між ними. Діаграма класів допомагає визначити основні концепції, що лежать в основі системи, та спроектувати її архітектуру. Діаграма класів також може відображати інтерфейси, абстрактні класи, перелічування, пакети та інші елементи UML [8].

Для формування діаграми класів необхідно виконати наступні кроки.

Визначити мету та обсяг діаграми. Наприклад, діаграма може бути створена для певного модуля, підсистеми або всієї системи, для показу загальної структури або детального проектування.

Виділити основні класи, що входять до діаграми. Класи можуть бути знайдені за допомогою аналізу вимог, специфікацій, сценаріїв використання або інших джерел інформації про систему. Класи повинні мати чіткі назви, що відображають їх сутність та відповідальність.

Визначити атрибути та методи для кожного класу. Атрибути — це характеристики, що описують стан класу, наприклад, ім’я, розмір, колір тощо. Методи — це дії, які може виконувати клас, наприклад, додати, видалити, змінити тощо. Атрибути та методи повинні бути релевантними для класу та відповідати його функціональності.

Встановити відношення між класами. Відношення — це логічні зв’язки, що існують між класами, наприклад, наслідування, реалізація, асоціація, агрегація, композиція тощо. Відношення допомагають показати, як класи взаємодіють та залежать один від одного.

Діаграма класів складається з наступних елементів.

Клас — це блакитна схема для об’єкта. Клас описує, яким буде об’єкт, але не є самим об’єктом. Класи описують типи об’єктів, а об’єкти — це використовувані екземпляри класів. Кожен клас має свою назву, атрибути та операції, які показуються в різних розділах прямокутника2.

Атрибут — це властивість класу, яка описує його стан. Атрибути відображаються в другому розділі класу після його назви. Кожен атрибут має свій тип, який показується після двокрапки. Атрибути відповідають змінним даних (data members) у коді.

Операція — це поведінка класу, яка описує, що він може робити. Операції відображаються в третьому розділі класу після його атрибутів. Кожна операція має свій підпис, який складається з назви, списку параметрів та типу повернення. Операції відповідають методам класу у коді.

Зв’язок — це відношення між класами, яке показує, як вони пов’язані один з одним. Зв’язки відображаються за допомогою ліній та символів, які вказують на їх тип, напрямок та кардинальність. Деякі типи зв’язків між класами — це асоціація, агрегація, композиція, унаслідування та реалізація1.

Діаграма класів допомагає проектувати архітектуру системи, аналізувати її вимоги та документувати її специфікації. Діаграма класів також може мати різні перспективи, залежно від стадії розробки процесу. Наприклад, концептуальна перспектива фокусується на визначенні основних класів домену,

аналітична перспектива фокусується на визначенні поведінки та відповідальності класів, а дизайнерська перспектива фокусується на визначенні деталей реалізації класів [9].

Діаграма класів допомагає визначити основні сутності та їх взаємозв’язки в системі, а також специфікувати їх атрибути та операції.

Діаграма класів також служить основою для реалізації системи на обраному мові програмування. На її основі можна генерувати код за допомогою різних інструментів, таких як prisma, який дозволяє створювати схеми баз даних та моделі даних для Next.js. Діаграма класів зображена на рисунку 2.6.

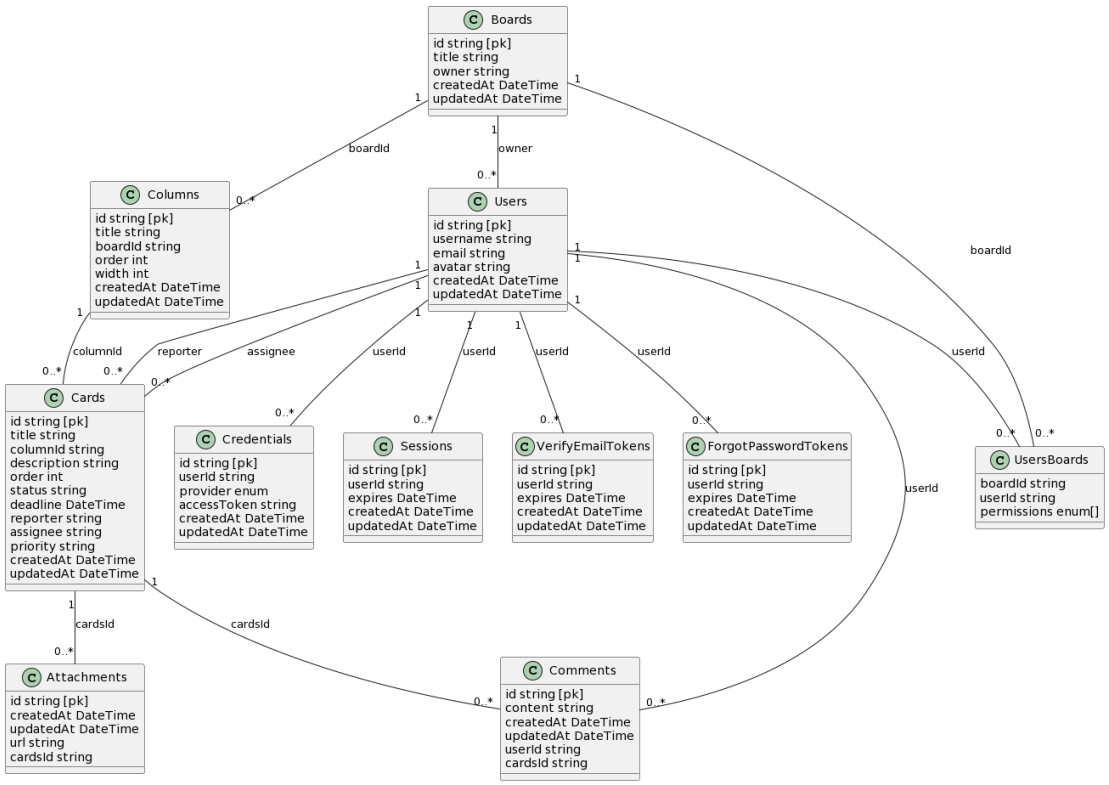

Рисунок 2.6 — Діаграма класів

2.3 Визначення впорядкованої за часом взаємодії об'єктів, формування діаграми послідовності.

Для того, щоб проаналізувати, як об'єкти веб-додатку взаємодіють між собою в процесі виконання задач і проектів, ми використовуємо діаграми послідовності. Діаграма послідовності — це графічне зображення, яке показує,

як об'єкти обмінюються повідомленнями в хронологічному порядку. Діаграма послідовності складається з наступних елементів.

Вертикальні лінії — представляють об'єкти, які беруть участь у взаємодії. Об'єкти можуть бути класами, екземплярами класів, ролями або акторами. Назви об'єктів пишуться над лініями.

Горизонтальні стрілки — представляють повідомлення, які об'єкти надсилають один одному. Повідомлення можуть бути синхронними, асинхронними, відповідями або створенням/знищенням об'єктів. Назви повідомлень пишуться над стрілками.

Прямокутники — представляють час виконання повідомлень або методів. Прямокутники розташовані на вертикальних лініях об'єктів і показують, коли об'єкт активний або пасивний.

Лінії життя — представляють час існування об'єктів. Лінії життя починаються від моменту створення об'єкта і закінчуються від моменту його знищення. Лінії життя можуть бути перерваними, якщо об'єкт тимчасово не бере участі у взаємодії.

Діаграми послідовності допомагають нам визначити, які об'єкти та повідомлення необхідні для реалізації кожного сценарію, а також виявити можливі проблеми або покращення в процесі взаємодії [10]. Наприклад, на рисунку 1 показана діаграма послідовності для сценарію створення нового проекту. На рисунку 2.7 показана діаграма послідовності додатку.

Сценарій такий:

— користувач вводить URL веб-додатку у браузері, який надсилає HTTP GET запит на сервер.

— сервер запитує дані з бази даних, яка повертає дані на сервер.

— сервер надсилає HTTP відповідь з HTML, CSS, JS на браузер, який відображає веб-сторінку для користувача.

— користувач реєструється або входить у свій обліковий запис, вводячи свої облікові дані.

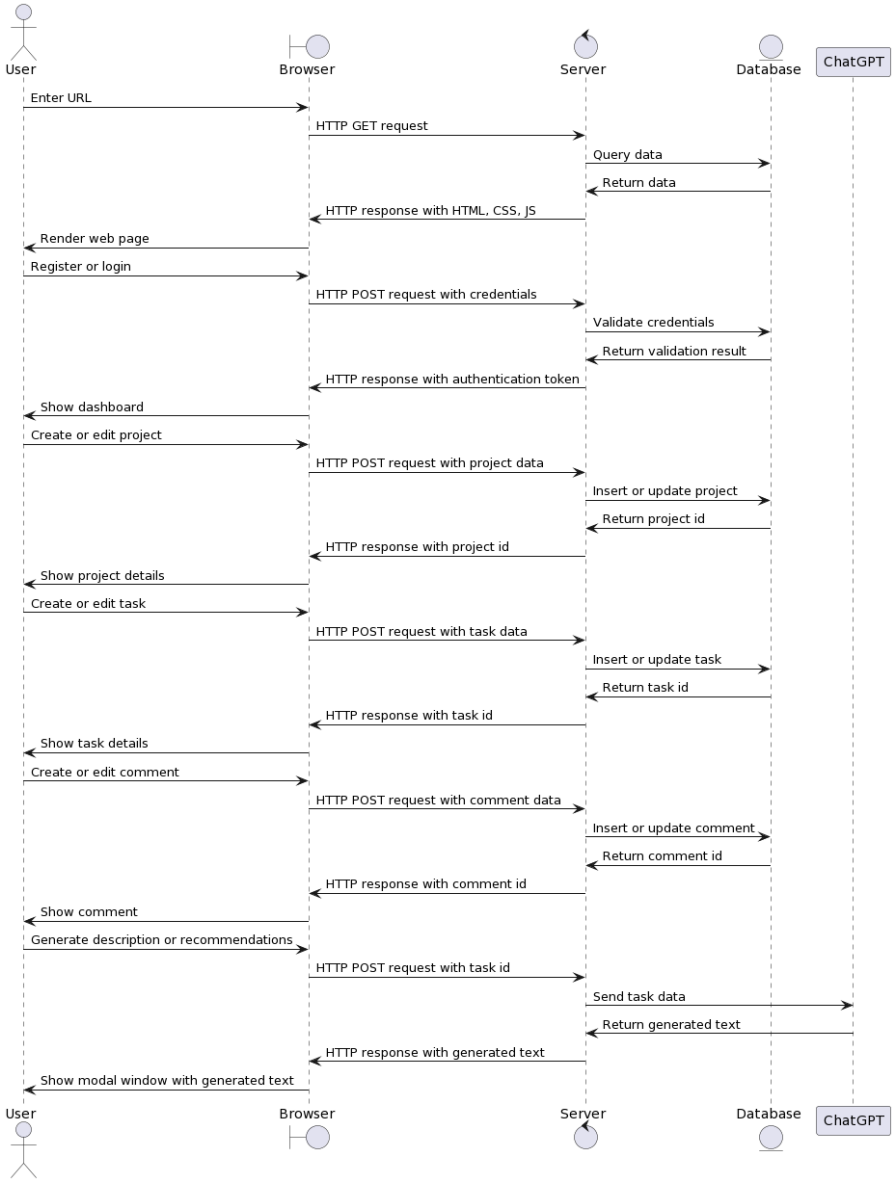

Рисунок 2.7 — Діаграма послідовностей

— браузер надсилає HTTP POST запит з обліковими даними на сервер, який перевіряє їх у базі даних.

— база даних повертає результат перевірки на сервер, який надсилає HTTP відповідь з токеном аутентифікації на браузер;

— браузер показує користувачеві панель керування;

— користувач створює або редагує проект, вводячи дані проекту;

— браузер надсилає HTTP POST запит з даними проекту на сервер, який додає або оновлює проект у базі даних;

— база даних повертає ідентифікатор проекту на сервер, який надсилає HTTP відповідь з ідентифікатором проекту на браузер;

— браузер показує користувачеві деталі проекту;

— користувач створює або редагує задачу, вводячи дані задачі;

— браузер надсилає HTTP POST запит з даними задачі на сервер, який додає або оновлює задачу у базі даних;

— база даних повертає ідентифікатор задачі на сервер, який надсилає HTTP відповідь з ідентифікатором задачі на браузер;

— браузер показує користувачеві деталі задачі;

— користувач створює або редагує коментар, вводячи дані коментаря;

— браузер надсилає HTTP POST запит з даними коментаря на сервер, який додає або оновлює коментар у базі даних;

— база даних повертає ідентифікатор коментаря на сервер, який надсилає HTTP відповідь з ідентифікатором коментаря на браузер;

— браузер показує користувачеві коментар;

— користувач генерує опис або рекомендації для задачі, натискаючи кнопку;

— браузер надсилає HTTP POST запит з ідентифікатором задачі на сервер, який відправляє дані задачі на ChatGPT;

— ChatGPT повертає згенерований текст на сервер, який надсилає HTTP відповідь з згенерованим текстом на браузер;

— браузер показує користувачеві модальне вікно з згенерованим текстом.

2.4 Аналіз та вибір технологій для розробки веб-додатку

2.4.1 Вибір мови програмування

Мова програмування є одним з найважливіших факторів, які впливають на успішність розробки веб-додатку. Мова програмування визначає, яким чином буде написаний код, як він буде виконуватися, як він буде тестуватися, як він

буде сумісний з іншими технологіями, як він буде підтримуватися та розвиватися. Мова програмування також впливає на продуктивність, надійність, безпеку, зручність та якість веб-додатку.

Для розробки веб-додатку було обрано мову програмування TypeScript. TypeScript є надбудовою над JavaScript, яка додає до нього статичну типізацію, що поліпшує читабельність, надійність та підтримку коду. TypeScript також дозволяє використовувати сучасні функції JavaScript, такі як класи, модулі, стрілкові функції, деструктуризація тощо, і компілювати їх у сумісний з більшістю браузерів код. TypeScript є однією з найпопулярніших, найшвидших та найбільш вимогливих мов програмування, яка використовується для розробки веб-додатків [11].

TypeScript також має багатий набір інструментів, бібліотек та фреймворків, які підтримують його, таких як Next.js, React, Angular, Vue, Node.js, Express, MongoDB, Firebase, AWS тощо. TypeScript також сумісний з JavaScript, що дозволяє використовувати будь-який існуючий код JavaScript у своєму проекті. TypeScript є ідеальним вибором для розробки веб-додатку, який потребує високої продуктивності, надійності, безпеки та зручності.

2.4.2 Вибір фреймворка

Фреймворк є набором бібліотек, інструментів та правил, які спрощують та прискорюють розробку веб-додатку. Фреймворк надає готові рішення для загальних задач, таких як рендеринг, маршрутизація, аутентифікація, валідація, кешування тощо. Фреймворк також забезпечує структуру та архітектуру коду, яка сприяє його організації, модульності, тестованості та підтримці, також впливає на швидкість, стабільність, безпеку, масштабованість та якість веб-додатку.

Для розробки веб-додатку було обрано фреймворк **Next.js**. Next.js є фреймворком для розробки веб-додатків на основі React, який дозволяє створювати динамічні, інтерактивні та високопродуктивні веб-додатки. Вибір цієї технології було обґрунтовано наступними перевагами.

Гібридний рендеринг. Next.js підтримує різні способи рендерингу сторінок, такі як статичний, серверний та клієнтський. Це дозволяє оптимізувати швидкість завантаження, відповідальність, доступність та пошукову оптимізацію веб-додатку. Next.js також дозволяє використовувати інкрементальну статичну генерацію, яка динамічно генерує статичні сторінки на основі запитів користувачів, що забезпечує свіжість та актуальність даних.

Автоматична оптимізація коду. Next.js автоматично оптимізує код веб-додатку, використовуючи такі техніки, як бандлінг, мініфікація, транспіляція, розділення коду, ліниве завантаження, префетчинг тощо. Це покращує продуктивність, ефективність, сумісність та якість веб-додатку. Next.js також підтримує TypeScript, CSS модулі, Sass, Less, Stylus та інші технології, які поліпшують розробку та стиль веб-додатку.

Має вбудовану маршрутизацію, що спрощує навігацію та перехід між сторінками веб-додатку. Next.js використовує систему файлів (file system) для визначення маршрутів, що робить їх простими та інтуїтивними. Next.js також підтримує динамічні маршрути (dynamic routes), вкладені маршрути (nested routes), захищені маршрути (protected routes) та перехоплення маршрутів (route interception).

Покращення користувацького досвіду. Next.js забезпечує швидке завантаження та високу продуктивність сторінок, що призводить до чудового користувацького досвіду та збільшення користувацької взаємодії1.

Спрощення масштабування. Next.js спрощує масштабування додатку, дозволяючи йому легко витримувати високі навантаження та адаптуватися до зростаючих потреб користувачів. За допомогою безсерверних варіантів розгортання масштабування стає безшовним процесом.

Прискорення розробки. Next.js надає широкий набір функцій “з коробки”, які покращують цикли розробки. Розробники можуть вибирати вбудовані рішення для маршрутизації, управління станом та стилізації, зменшуючи потребу в ручній конфігурації та залежностях від сторонніх джерел.

Містить API-шар, що дозволяє створювати RESTful або GraphQL API для веб-додатку. Next.js надає API-маршрути (API routes), які можна використовувати для обробки запитів та відповідей на стороні сервера. Next.js також інтегрується з безсерверними функціями (serverless functions), які можна використовувати для реалізації бізнес-логіки та взаємодії з базами даних або іншими сервісами.

Підтримує інтернаціоналізацію (i18n), що дозволяє адаптувати веб-додаток до різних мов та регіонів.

Інтеграція з іншими технологіями: Next.js легко інтегрується з іншими технологіями. Це дозволяє використовувати багатий набір функціональності, бібліотек та інструментів, які надають ці технології, для створення веб-додатку. Next.js також підтримує хуки, компоненти вищого порядку, контекст, редакс та інші концепції React, які полегшують управління станом, даними та логікою веб-додатку [12].

2.4.3 Вибір бази даних

База даних є системою для зберігання, обробки та надання даних, які використовуються веб-додатком. База даних визначає, яким чином будуть організовані, структуровані, запитувані, модифіковані, аналізовані та захищені дані. База даних також впливає на швидкість, надійність, безпеку, масштабованість та якість веб-додатку.

Для розробки веб-додатку було обрано базу даних postgressql. Postgressql є відкритим, об’єктно-реляційним, розподіленим та сумісним з SQL базою даних, яка надає наступні переваги.

Postgressql дозволяє створювати та використовувати різні типи даних, такі як текст, числа, дати, час, JSON, XML, геометричні, масиви, hstore тощо. Postgressql також дозволяє створювати та використовувати власні типи даних, функції, оператори, індекси, тригери, правила, агрегати тощо. Postgressql також підтримує наслідування, поліморфізм, спадкування, перевантаження тощо. Postgressql також дозволяє використовувати різні мови програмування, такі як

PL/pgSQL, PL/Python, PL/Perl, PL/Tcl, PL/Java, PL/R тощо, для написання процедур, тригерів, функцій тощо. Postgressql також дозволяє використовувати різні розширення, такі як PostGIS, pgcrypto, hstore, ltree, citext, pg_trgm тощо, для додавання додаткової функціональності до бази даних. Postgressql є надзвичайно гнучкою базою даних, яка дозволяє адаптувати її до будь-яких потреб та вимог веб-додатку.

Postgressql є стабільною, зрілою та перевіреною часом базою даних, яка гарантує цілісність, консистентність та доступність даних. Postgressql підтримує такі функції, як транзакції, ACID, MVCC, логування, реплікація, кластеризація, бекап, відновлення, балансування навантаження тощо, які забезпечують високий рівень надійності бази даних. Postgressql також має сильну спільноту, яка підтримує, оновлює та вдосконалює базу даних. Postgressql є надійною базою даних, яка зберігає, обробляє та надає дані без втрат, збоїв або проблем.

Postgressql є безпечною базою даних, яка захищає дані від несанкціонованого доступу, зловживання, зміни, витоку або втрати. Postgressql підтримує такі функції, як шифрування, аутентифікація, авторизація, ролі, привілеї, аудит, фільтрація, ROW LEVEL SECURITY, SSL/TLS тощо, які забезпечують високий рівень безпеки бази даних. Postgressql також дотримується стандартів та регулятивів, таких як GDPR, HIPAA, PCI DSS тощо, які вимагають відповідального та законного поводження з даними. Postgressql є безпечною базою даних, яка гарантує конфіденційність, цілісність та доступність даних.

Postgressql є однією з найбільш продуктивних баз даних, яка використовує ефективні алгоритми та структури даних для зберігання, запитування та обробки даних. Postgressql також використовує різні методи оптимізації, такі як кешування, партиціювання, матеріалізацію, паралелізацію, індексацію та кластеризацію, які покращують швидкість та масштабованість бази даних [13].

3 РОЗРОБКА ПОСТАВЛЕНОЇ ЗАДАЧІ

3.1 Опис режимів роботи програмної системи

Програмна система, розроблена в рамках цієї магістерської роботи, є веб-додатком для створення, організації та відстеження індивідуальних й командних задач і проектів. Додаток має наступні режими роботи.

У режимі реєстрації/авторизації користувач може створити свій обліковий запис або увійти в існуючий, вказавши своє ім’я, електронну адресу та пароль. Також користувач може зареєструватись за допомогою GitHub або Gmail. Система перевіряє, чи відповідають ці дані вимогам безпеки та унікальності, та надсилає лист з підтвердженням на електронну адресу користувача. Користувач має перейти за посиланням у листі, щоб активувати свій обліковий запис. Після цього користувач може увійти у систему, використовуючи свою електронну адресу та пароль. Якщо користувач забув свій пароль, він може скористатися функцією відновлення пароля, яка надішле йому посилання на електронну пошту для скидання пароля. Для реалізації цього режиму використовується сервіс **supabase**, який надає аутентифікацію, авторизацію та управління користувачами.

У режимі робочого столу користувач може переглядати свої проекти, створювати нові проекти, видаляти проекти, редагувати проекти. При виборі конкретного проекту користувач переходить до режиму дошки.

Режим дошки дозволяє користувачам переглядати дошки, які представляють проекти або процеси. Кожна дошка складається зі списків — колонок, які представляють категорії або стадії, які відповідають статусам задач, наприклад, “To do”, “In progress”, “Done” тощо, та карток, які представляють задачі або елементи. Користувач може додавати, переміщувати, змінювати, видаляти та перетягувати списки та картки за своїм бажанням , а також змінювати порядок та ширину списків та карток. Дошка допомагає користувачам візуалізувати свій робочий процес та бачити загальну картину своїх задач та проектів. Дошка також дозволяє користувачам фільтрувати, сортувати,

групувати картки. При натисканні на картку задачі користувач переходить до режиму деталей задачі.

У режимі деталей задачі користувач може переглядати та редагувати деталі задачі, такі як назва, опис, термін, приоритет, виконавець, репортер, коментарі, вкладення, статус, дата. Користувач може повернутися до режиму дошки або видалити задачу. Користувач також може отримувати допомогу від штучного інтелекту у вигляді ChatGPT, який може генерувати опис та рекомендації для задач на основі їх назви та інших даних. Користувач може активувати ChatGPT, натиснувши на кнопку "Ask ChatGPT". Користувач може прийняти, відхилити або змінити згенерований текст за своїм бажанням.

Режим співпраці з іншими користувачами дозволяє користувачам співпрацювати з іншими користувачами над спільними проектами або задачами. Користувач може запрошувати або приймати запрошення від інших користувачів для участі в командних проектах. Користувач також може переглядати профілі інших користувачів, де він може бачити їх ім’я, електронну пошту, аватар, кількість проектів, задач, коментарів та файлів, які вони створили або до яких вони долучилися. Користувач також може спілкуватися з іншими користувачами за допомогою коментарів до задач, де він може надсилати текстові повідомлення, емодзі, зображення або файли.

3.2 Архітектура веб-додатку

Архітектура веб-додатку визначає взаємодію між додатками, проміжними системами та базами даних, щоб забезпечити спільну роботу декількох додатків. Архітектура веб-додатку має велике значення для майбутнього зростання, ефективності, надійності, безпеки та масштабованості веб-додатку [14].

Архітектура веб-додатку базується на наступних принципах.

Модульність — веб-додаток складається з різних модулів, які відповідають за окремі функції та компоненти. Кожен модуль може бути розроблений, тестований та оновлюваний незалежно від інших. Це сприяє зручності розробки, підтримки та масштабування веб-додатку.

Гнучкість — веб-додаток адаптується до різних потреб та вимог користувачів. Він дозволяє налаштовувати параметри проектів та задач, вибирати різні режими роботи, змінювати дизайн та інтерфейс, інтегрувати зовнішні сервіси та додатки.

Інтерактивність — веб-додаток надає користувачам можливість взаємодіяти з іншими елементами системи, такими як інші користувачі, проекти, задачі, повідомлення, рекомендації, зворотний зв’язок тощо. Він також використовує різні засоби візуалізації, анімації, звукових та текстових ефектів для підвищення привабливості та зручності використання.

Веб-додаток має клієнт-серверну архітектуру, де клієнтська частина відповідає за відображення графічного інтерфейсу користувача, а серверна частина відповідає за обробку запитів, доступ до бази. Схема архітектури веб-додатку представлена на рис. 3.1.

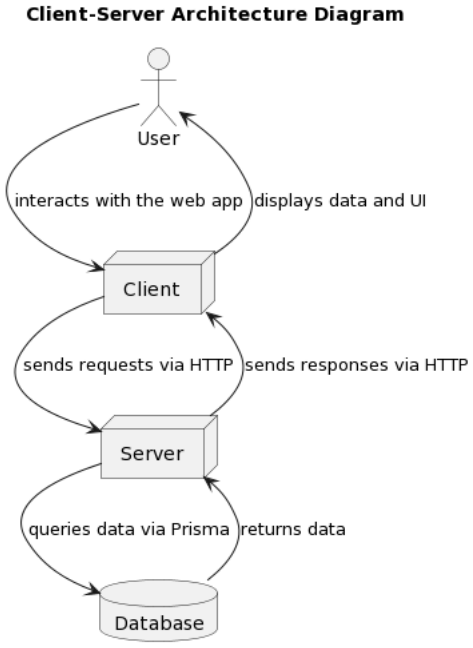

Рисунок 3.1 — Архітектура веб-додатку

Клієнт-серверна архітектура — це архітектура, у якій веб-додаток поділяється на дві основні частини: клієнтську та серверну. Клієнтська частина відповідає за відображення інтерфейсу користувача у веб-браузері за допомогою HTML, CSS та JavaScript. Серверна частина відповідає за обробку запитів

клієнта, виконання бізнес-логіки, зберігання та надання даних за допомогою веб-сервера, бази даних, мови програмування тощо. Клієнт-серверна архітектура дозволяє розділити логіку та презентацію, покращити продуктивність та безпеку, а також спростити масштабування та оновлення окремих частин додатку, але вона також потребує більше ресурсів та складнішої координації між компонентами [15].

Додаток складається з трьох основних компонентів: клієнтської частини, серверної частини та бази даних.Отже, наш веб-додаток має наступну архітектуру:

— клієнтська частина — це та частина веб-додатку, яка виконується на браузері користувача та відповідає за відображення інтерфейсу, обробку взаємодії з користувачем, виконання запитів до сервера та отримання відповідей;

— серверна частина — це та частина веб-додатку, яка виконується на хмарному сервері та відповідає за обробку запитів від клієнтської частини, виконання бізнес-логіки, зберігання та надання даних, інтеграцію з зовнішніми сервісами;

— база даних — це та частина веб-додатку, яка зберігає та надає дані, які використовуються веб-додатком, такі як дані про користувачів, задачі, проекти, коментарі, сповіщення та інші. База даних є реляційною та побудована на основі PostgreSQL, яка є відкритою, надійною, масштабованою та гнучкою.

3.3 Розробка на налаштування бази даних

Для створення веб-додатку, який дозволяє створювати, організовувати та відстежувати індивідуальні й командні задачі і проектів, необхідно розробити та налаштувати базу даних, яка зберігатиме всю необхідну інформацію про користувачів, проекти, задачі, повідомлення та інші сутності. База даних є важливою складовою веб-додатку, оскільки вона забезпечує надійне, безпечне та ефективне зберігання та обробку даних.

Для зберігання і обробки даних про користувачів, проекти, задачі та повідомлення веб-додатку використовується база даних PostgreSQL.

PostgreSQL — це відкрита система керування реляційними базами даних, яка підтримує SQL-стандарт та має багато розширень та можливостей.

Для підключення PostgreSQL до Next.js використовується Prisma. Prisma — це інструмент для роботи з базами даних, який дозволяє моделювати, запитувати та мігрувати дані за допомогою сучасного типобезпеченого API. Prisma генерує клієнтський код на основі схеми бази даних, який можна використовувати для виконання запитів до бази даних з Next.js.

Для розгортання та управління базою даних використовується Supabase. Supabas — це платформа, яка надає хмарні послуги для розробки веб-додатків, такі як бази даних, аутентифікація, зберігання, функції тощо. Supabase дозволяє легко підключити PostgreSQL до Next.js та надає інтерфейс для перегляду та редагування даних.

Тепер у нашому додатку потрібно додати об'єктно-реляційну проєкцію Prisma для роботи з базою даних. Інсталюємо її за допомогою команди — «yarn add prisma -D» та ініціалізуєм проект — «yarn prisma init». Це створює новий prismakаталог із файлом схеми Prisma та налаштовує SQLite як вашу базу даних. Тепер можна моделювати дані та створювати базу даних із таблицями. У нас створився файл .env в який ми записуємо у змінну DATABASE_URL посилання для підключення до бази Superbase. Також створилась папка prisma з файлом schema.prisma. Код у файлі schema.prisma:

```
generator client {
  provider = "prisma-client-js"
}
datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}
```

Схема Prisma забезпечує інтуїтивно зрозумілий спосіб моделювання даних. Додамо основні моделі до schema.prisma файлу:

```
model Board {
  id        String    @id @default(uuid())
  title     String
  owner     User      @relation(fields: [ownerId], references: [id])
  ownerId   String
  createdAt DateTime  @default(now())
  updatedAt DateTime  @updatedAt
  columns   Column[]
}
model Column {
  id        String    @id @default(uuid())
  title     String
  board     Board     @relation(fields: [boardId], references: [id], onDelete: Cascade)
  boardId   String
  order     Int
  width     Int
  createdAt DateTime  @default(now())
  updatedAt DateTime  @updatedAt
  cards     Card[]
}
model Card {
  id          String    @id @default(uuid())
  title       String
  column          Column       @relation(fields: [columnId], references: [id], onDelete:
Cascade)
  columnId    String
  description String?
  order       Int
  status      String
  deadline    DateTime?
```

```
  reporter    User      @relation(fields: [reporterId], references: [id])
  reporterId  String
  assignee    User?     @relation(fields: [assigneeId], references: [id])
  assigneeId  String?
  priority    String
  createdAt   DateTime  @default(now())
  updatedAt   DateTime  @updatedAt
}
model User {
  id          String   @id @default(uuid())
  username  String
  email     String
  avatar    String
  createdAt DateTime  @default(now())
  updatedAt DateTime  @updatedAt
}
```

Моделі в схемі Prisma мають дві основні мети:

— представлення таблиць у базі даних;

— служить основою для створеного API клієнта Prisma.

На даний момент у нас є схема Prisma, але ще немає бази даних. Виконуємо таку команду у вашому терміналі, щоб створити базу даних SQLite і таблиці, представлені нашими моделями: npx prisma migrate dev --name init.

Ця команда виконувала три дії:

— створив новий файл міграції SQL для цієї міграції в prisma/migrationskаталозі;

— виконав файл міграції SQL для бази даних;

— працював prisma generate під капотом (який встановлював @prisma/clientпакет і генерував налаштований клієнтський API Prisma на основі ваших моделей).

Щоб надсилати запити до бази даних, знадобиться файл TypeScript для виконання запитів Prisma Client. Створюєм екземпляр PrismaClient, який представляє інтерфейс запиту до бази даних:

```
import { PrismaClient } from "@prisma/client";
import { config } from "./config";
const prismaClientSingleton = () => {
  return new PrismaClient();
};
type PrismaClientSingleton = ReturnType<typeof prismaClientSingleton>;
const globalForPrisma = globalThis as unknown as {
  prisma: PrismaClientSingleton | undefined;
};
export const prisma = globalForPrisma.prisma || prismaClientSingleton();
if (!config.isProduction) {
  globalForPrisma.prisma = prisma;
}
```

Для розробки і налаштування бази даних були виконані наступні кроки.

Створення облікового запису на Supabase та створення нового проекту з PostgreSQL.

Створення схеми бази даних за допомогою Prisma Schema Language (PSL). PSL — це мова опису даних, яка дозволяє визначати моделі, поля, типи, відносини, індекси та інші параметри бази даних.

Генерація Prisma Client за допомогою команди npx prisma generate. Prisma Client — це автоматично створений бібліотека, яка надає типобезпечене API для запитів до бази даних.

Створення міграцій бази даних за допомогою команди npx prisma migrate dev. Міграції бази даних — це скрипти, які змінюють структуру бази даних відповідно до схеми PSL.

Підключення Prisma Client до Next.js за допомогою функції Prisma.createClient(). Функція Prisma.createClient() створює екземпляр клієнта Prisma, який можна використовувати для виконання запитів до бази даних.

Таким чином, розробка та налаштування бази даних дозволило створити надійне, безпечне та ефективне сховище даних для веб-додатку, яке підтримує всі необхідні функції та можливості для створення, організації та відстеження індивідуальних й командних задач і проектів.

3.4 Розробка серверної частини

Розробка серверної частини веб-додатку полягає у створенні та підтримці логіки, яка забезпечує обробку даних, комунікацію з базами даних та іншими системами, а також надання даних для клієнтської частини за допомогою API. Серверна частина веб-додатку відіграє важливу роль у забезпеченні функціональності, продуктивності, безпеки та масштабованості веб-додатку.

Важливою частиною всієї системи є сховище, яке в даному випадку є REST-сервіс.

REST (REpresentational State Transfer) — розшифровується як передача репрезентативного стану. Це набір принципів проектування підвищення масштабованості і гнучкості мережевих комунікацій. Ці принципи відповідають на низку питань.Які компоненти системи? Як вони спілкуються між собою? Як можна гарантувати, чи можливість поміняти різні частини системи в будь-який час? Як можна масштабувати систему обслуговування великої кількості користувачів?

Система RESTful — це будь-яка мережа, що задовольняє розглянутим обмеженням. Система RESTful повинна бути гнучкою для різних випадків використання, що масштабується для підтримки великої кількості користувачів та компонентів та адаптується з часом.

Серверна частина розроблена за стандартом CRUD:

— Create — створення даних;

— Read — зчитування даних;

— Update — оновлення даних;

— Delete — видалення даних.

У архітектури стилю REST кожна інформаційна одиниця вважається ресурсом, і ці ресурси адресуються за допомогою універсальних ідентифікаторів(URI). Як правило, ідентифікатори представляють собою веб-посилання. Дії над ресурсами виконуються при використанні обмеженого набору чітко визначених операцій. REST як стиль клієнт-серверної архітектури визначає конструкцію для обміну станом цих ресурсів із застосуванням певного інтерфейсу і протоколу. Для розробки веб-служби з передачею стану, необхідно розуміти деталі протоколу передачі гіпертексту (HTTP) і унікальних ідентифікаторів ресурсів (URI), а також дотримуватись певних принципів проектування. По суті, це означає, що кожен URI є поданням певного об’єкта. З даним об’єктом можна взаємодіяти за допомогою запитів HTTP:

— GET — для одержання змісту;

— DELETE — для видалення вмісту;

— POST — для створення вмісту;

— PATCH — часткове оновлення, зміна;

— PUT — для оновлення вмісту.

В даному додатку було використано основні методи GET для отримання інформації, POST для її відправки, PATCH для оновлення інформації та DELETE для видалення.

Для розробки API-маршрутів було використано Next.js API Routes, які є вбудованим функціоналом Next.js для створення API-маршрутів без потреби в додаткових серверах або фреймворках. API-маршрути — це спосіб комунікації між клієнтською та серверною частинами веб-додатку. API-маршрути дозволяють виконувати запити до сервера та отримувати відповіді у форматі JSON.

Next.js API Routes дозволяють створювати API-маршрути за допомогою файлів у папці app/api, які експортують функції, що приймають два параметри: req (request) та res (response). Next.js API Routes підтримують різні HTTP-методи,

такі як GET, POST, PUT, DELETE та інше. Next.js API Routes також дозволяють використовувати middleware, які є функціями, що виконуються перед обробкою запиту, наприклад, для перевірки аутентифікації, валідації даних, логування та інше [16].

Серверна частина веб-додатку складається з наступних API-маршрутів.

/api/auth — це API-маршрут, який відповідає за аутентифікацію користувачів. Він надає можливість реєстрації, входу, виходу, відновлення пароля, перевірки електронної адреси. Цей API-маршрут підтримує наступні HTTP-методи.

POST /api/auth/sign-up — цей метод дозволяє користувачу зареєструватися в веб-додатку, вказавши електронну адресу, пароль та ім’я користувача. Якщо реєстрація успішна, то користувач отримує електронний лист з посиланням для підтвердження електронної адреси.

POST /api/auth/sign-in — цей метод дозволяє користувачу увійти в веб-додатку, вказавши електронну адресу та пароль. Якщо вхід успішний, то користувач отримує токен сесії, який зберігається в куках.

POST /api/auth/forgot — цей метод дозволяє користувачу відновити пароль, вказавши електронну адресу. Якщо електронна адреса існує в базі даних, то користувач отримує електронний лист з посиланням для скидання пароля.

POST /api/auth/verify-email — цей метод дозволяє користувачу підтвердити електронну адресу.

/api/boards — це API-маршрут, який відповідає за роботу з проектами веб-додатку. Він надає можливість зберігати, отримувати, змінювати та видаляти проекти. Цей API-маршрут підтримує наступні HTTP-методи.

GET /api/boards — цей метод дозволяє користувачу отримати список усіх проектів, до яких він має доступ.

POST /api/boards — цей метод дозволяє користувачу створити новий проект в веб-додатку.

GET /api/boards/:id — цей метод дозволяє користувачу отримати детальну інформацію про проект за його ідентифікатором.

PATCH /api/boards/:id — цей метод дозволяє користувачу редагувати існуючий проект в веб-додатку.

/api/columns — це API-маршрут, який відповідає за роботу з колонками проекту. Він надає можливість зберігати, отримувати, змінювати та видаляти колонки. Цей API-маршрут підтримує наступні HTTP-методи.

GET /api/columns — цей метод дозволяє користувачу отримати список усіх колонок одного з проектів.

POST /api/columns — цей метод дозволяє користувачу створити нову колонку в проекті.

GET /api/columns/:id — цей метод дозволяє користувачу отримати колонку за ідентифікатором.

PATCH /api/columns/:id — цей метод дозволяє користувачу редагувати колонку.

DELETE /api/boards/:id — цей метод дозволяє користувачу видалити колонку разом з задачами в проекті за ідентифікатором.

PATCH /api/columns/order — цей метод дозволяє користувачу змінити порядок відображення колонки.

api/cards — це API-маршрут, який відповідає за роботу з задачами веб-додатку. Цей API-маршрут підтримує наступні HTTP-методи.

GET /api/cards — цей метод дозволяє користувачу отримати список усіх задач у конкретній колонці.

POST /api/cards — цей метод дозволяє користувачу створити нову задачу.

GET /api/cards/:id — цей метод дозволяє користувачу отримати детальну інформацію про задачу за її ідентифікатором.

PATCH /api/cards/:id — цей метод дозволяє користувачу редагувати існуючу задачу.

DELETE /api/cards/:id — цей метод дозволяє користувачу видалити існуючу задачу в веб-додатку за її ідентифікатором.

PATCH /api/cards/order — цей метод дозволяє користувачу змінити порядок відображення задачі в колонці.

/api/comments — це API-маршрут, який відповідає за роботу з коментарями веб-додатку. Цей API-маршрут підтримує наступні HTTP-методи.

GET /api/comments — цей метод дозволяє користувачу отримати список усіх коментарів до задачі.

POST /api/comments — цей метод дозволяє користувачу створити новий коментар до задачі.

DELETE /api/comments/:id — цей метод дозволяє користувачу видалити існуючий коментар в веб-додатку за його ідентифікатором.

/api/attachments — це API-маршрут, який відповідає за роботу з прикріпленнями веб-додатку. Він використовує сервіс **Supabase Storage**, який надає можливість зберігати та отримувати файли, такі як документи, зображення, відео, аудіо та інше. Цей API-маршрут підтримує наступні HTTP-методи.

GET /api/attachments — цей метод дозволяє користувачу отримати список усіх прикріплень до задачі.

POST /api/attachments — цей метод дозволяє користувачу додати нове прикріплення в веб-додатку.

DELETE /api/attachments/:id — цей метод дозволяє користувачу видалити прикріплення в задачі за його ідентифікатором.

Розглянемо розробку API-маршруту /api/columns. Усі API- маршрути знаходяться в папці api та мають префікс api, наприклад api/columns. API-роути приймають запити від клієнта та повертають відповіді у форматі JSON. В папці columns є 3 файли: dto.ts, route.ts та [id]/route.ts. Для початку у файлі dto.ts додамо схеми створення та зміни колонки за допомогою бібліотеки для оголошення схем і перевірки TypeScript—Zod.

```
export const createColumnDto = z.object({
  title: z.string().min(1).max(20),
  boardId: z.string().uuid(),
  width: z.number().min(200).default(200),
});
export type CreateColumnDto = z.infer<typeof createColumnDto>;
```

```
export const updateColumnDto = createColumnDto
  .omit({
    boardId: true,
  })
  .partial();
export type UpdateColumnDto = z.infer<typeof updateColumnDto>;
export const updateColumnsOrderDto = z.array(
  z.object({
    id: z.string().uuid(),
    order: z.number(),
  })
);
```

По маршруту api/columns створимо GET та POST запити для отримання всіх колонок в проекті та додавання колонки в проект:

```
export async function GET(req: Request) {
  const { searchParams } = new URL(req.url);
  const boardId = searchParams.get("boardId");
  if (!boardId) {
    return NextResponse.json(
      [
        {
          code: "missing_query_param",
          field: "boardId",
          message: "Query param boardId is required",
        },
      ],
      { status: 400 }
    );
  }
```

```
  const columns = await prisma.columns.findMany({
    where: {
      boardId,
    },
    orderBy: {
      order: "asc",
    },
  });
  return NextResponse.json(columns);
}
export async function POST(req: Request) {
  const bodyRaw = await req.json();
  const validateBody = createColumnDto.safeParse(bodyRaw);
  if (!validateBody.success) {
    return NextResponse.json(validateBody.error.issues, { status: 400 });
  }
  const { title, boardId, width } = validateBody.data;
  const lastColumn = await prisma.columns.findFirst({
    where: {
      boardId,
    },
    orderBy: {
      order: "desc",
    },
  });
  const newColumn = await prisma.columns.create({
    data: {
      title,
      boardId,
      width,
```

```
    order: lastColumn ? lastColumn.order + 1 : 0,
   },
  });
  return NextResponse.json(newColumn);
}
```

По маршруту api/columns/:id було створено наступні запити.

Запит GET — для отримання колонки по id.

```
export async function GET(req: Request, { params }: ColumnRouteContext) {
  const { id } = params;
  const column = await prisma.columns.findUnique({
   where: {
    id,
   },
   include: {
    cards: true,
   },
  });

  if (!column) {
   return NextResponse.json([
    {
     code: "not_found",
     messages: "Column not found",
    },
   ]);
  }
  return NextResponse.json(column);
}
```

Запит PATCH — для оновлення колонки по id.

```
export async function PATCH(req: Request, { params }: ColumnRouteContext) {
  const { id } = params;
  const bodyRaw = await req.json();
  const validateBody = updateColumnDto.safeParse(bodyRaw);
  if (!validateBody.success) {
    return NextResponse.json(validateBody.error.issues, { status: 400 });
  }
  const findColumn = await prisma.columns.findUnique({
    where: {
      id,
    },
  });
  if (!findColumn) {
    return NextResponse.json([
      {
        code: "not_found",
        messages: "Column not found",
      },
    ]);
  }
  const column = await prisma.columns.update({
    where: {
      id,
    },
    data: validateBody.data,
  });
  return NextResponse.json(column);
}
```

Запит DELETE — для видалення колонки по id.

```
export async function DELETE(req: Request, { params }: ColumnRouteContext) {
  const { id } = params;
  const findColumn = await prisma.columns.findUnique({
    where: {
      id,
    },
  });
  if (!findColumn) {
    return NextResponse.json([
      {
        code: "not_found",
        messages: "Column not found",
      },
    ]);
  }
  await prisma.columns.delete({
    where: {
      id,
    },
  });
  return NextResponse.json({}, { status: 200 });
}
```

Таким чином, розробка серверної частини дозволила створити потужну, безпечну та гнучку логіку для веб-додатку, яка забезпечує обробку даних, комунікацію з базами даних та іншими системами, а також надання даних для клієнтської частини за допомогою API.

3.5 Реалізація клієнтської частини

Клієнтська частина додатку відповідає за відображення інтерфейсу додатку на браузері користувача та за надсилання та отримання даних від

сервера. Для розробки клієнтської частини додатку було використано фреймворк Next.js, який дозволяє створювати веб-додатки з використанням React, який є одним з найпопулярніших бібліотек для створення інтерактивних інтерфейсів.

Для розробки клієнтської частини веб-додатку було вибрано наступні технології.

Next.js — це фреймворк для розробки React-додатків, який надає можливість створювати статичні та динамічні сайти з підтримкою серверного рендерингу, оптимізації продуктивності, маршрутизації, API-маршрутів та інших функцій.

Tailwind CSS — це утилітарний CSS-фреймворк, який дозволяє стилізувати елементи HTML за допомогою класів, які відповідають різним властивостям CSS. Такий підхід спрощує процес розробки та підтримки інтерфейсу, а також забезпечує адаптивність та гнучкість дизайну.

React Query — це бібліотека для управління станом даних, які отримуються з асинхронних запитів до сервера. React Query надає засоби для кешування, синхронізації, мутації, оптимістичного оновлення та інших операцій з даними, а також інтегрується з React-компонентами для автоматичного перерендерингу при зміні даних.

Архітектура клієнтської частини веб-додатку базується на концепції компонентів, які є основними будівельними блоками React-додатків. Компоненти представляють різні елементи інтерфейсу, такі як кнопки, списки, карточки, форми тощо, а також містять логіку, яка визначає їх поведінку та взаємодію з даними. Компоненти можуть бути функціональними або класовими, залежно від того, чи використовують вони хуки — спеціальні функції, які дозволяють працювати зі станом, ефектами, контекстом та іншими можливостями React у функціональних компонентах. У даному проекті було використано функціональні компоненти, оскільки вони є більш сучасними, простими та ефективними. Компоненти можуть бути також поділені на презентаційні та контейнерні, залежно від того, чи відповідають вони лише за відображення даних, чи також за їх отримання та обробку. У даному проекті було

використано такий підхід для розділення відповідальності між компонентами та спрощення їх повторного використання.

Розглянемо структуру додатку яка зображена на рисунку 3.2.

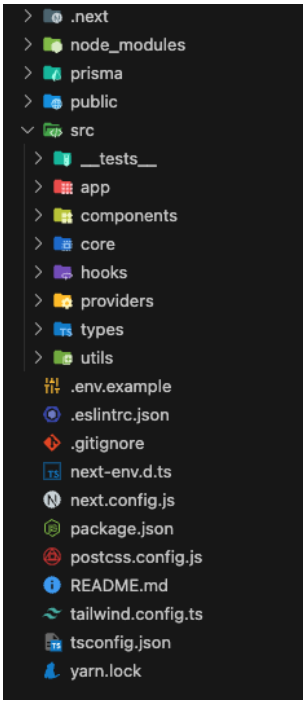

Рисунок 3.2 — Структура проекту

У каталозі components знаходяться основні React компоненти. Дана абстракція надає можливість збирати великі програми з маленьких «шматочків». Ці «шматочки» являють собою придатні до повторного використання об'єкти. Компоненти – основні будівельні блоки React–додатків, за допомогою яких інтерфейс розділяється на незалежні частини.

Створюються невеликі компоненти, які можна поєднувати, щоб сформувати більші або використовувати їх як самостійні елементи інтерфейсу. Найголовніше в цій концепції те, що і великі, і маленькі компоненти можна використовувати повторно і в новому проекті.

React-додаток можна представити як дерево компонентів. На верхньому рівні стоїть кореневий компонент, у якому вкладено довільну кількість інших компонентів. Кожен компонент повинен повернути JSX-розмітку, тим самим

вказуючи який HTML хочемо відрендерити в DOM. Дана схема зображена на рисунку 3.3.

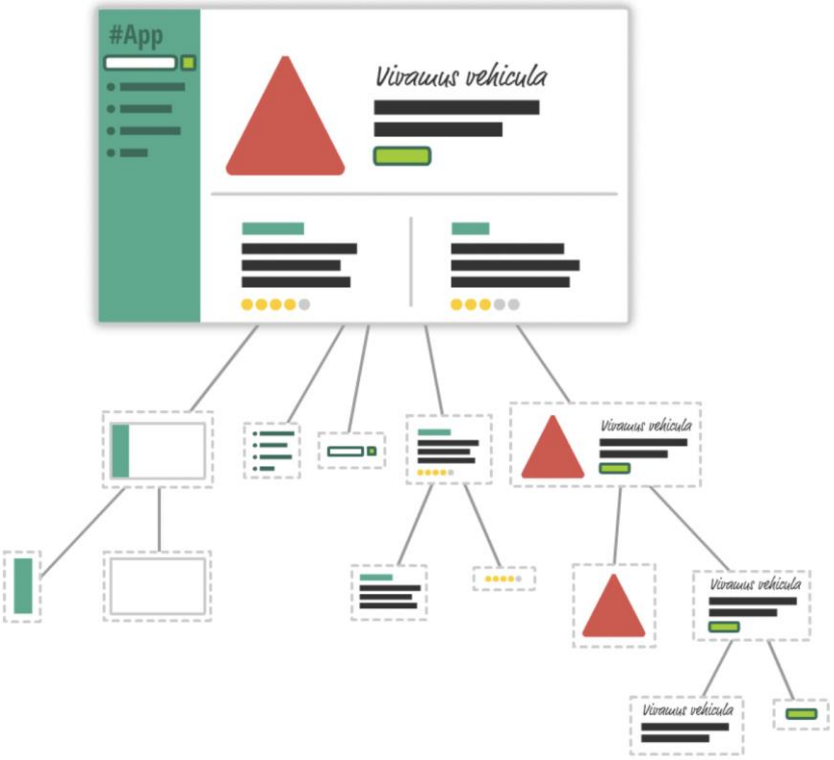

Рисунок 3.3 — Схема компонентів у вигляді дерева

Для стилізації нашого додатку ми використовуємо Tailwind CSS — утилітарний фреймворк для CSS, який дозволяє швидко та легко створювати адаптивні та красиві інтерфейси, використовуючи готові класи та компоненти. Tailwind CSS також надає можливість кастомізувати та розширювати свої стилі за допомогою конфігураційних файлів та плагінів.

Tailwind дозволяє стилізувати елементи інтерфейсу за допомогою утилітарних класів, які визначають різні властивості CSS, такі як колір, розмір, відступ, тінь, анімація тощо.

Tailwind надає такі переваги:

— швидкість розробки, оскільки не потрібно писати власні стилі CSS, а лише використовувати готові класи;

— гнучкість, оскільки можна комбінувати різні класи для досягнення бажаного ефекту;

— кастомізація, оскільки можна налаштувати кольорову палітру, розміри, шрифти, анімації тощо за своїми потребами;

— адаптивність, оскільки можна використовувати префікси для різних розмірів екрану, наприклад sm:, md:, lg: тощо.

Для отримання, кешування та синхронізації даних з сервера використано React-query, який дозволяє ефективно управляти станом даних в React. React-query надає можливість використовувати хуки, такі як useQuery, useMutation, useInfiniteQuery та інші, які дозволяють отримувати, мутувати, пагінувати та іншим чином працювати з даними.

React-query є бібліотекою, яка дозволяє керувати станом даних, які отримуються з асинхронних запитів до сервера. React-query надає такі переваги:

— кешування даних, що зменшує кількість запитів до сервера та покращує швидкість відгуку додатку;

— оновлення даних на основі залежностей, що забезпечує консистентність даних між різними компонентами додатку;

— повторення запитів у разі помилок, що підвищує надійність додатку;

— створення мутацій, що дозволяє виконувати операції зі зміною даних на сервері, такі як додавання, видалення, редагування тощо.

Ми використовуємо react-query для управління даними, які пов’язані з користувачами, дошками, списками, задачами та коментарями. Ми використовуємо наступні хуки react-query.

— useQuery — для отримання даних з сервера за допомогою GET запитів, наприклад, ми використовуємо цей хук для отримання списку дошок користувача, деталей конкретної дошки, списку задач на дошці тощо.

— useMutation — для зміни даних на сервері за допомогою POST, PUT, PATCH або DELETE запитів, наприклад, ми використовуємо цей хук для створення нової дошки, додавання нової задачі, редагування опису задачі, видалення коментаря тощо.

— useQueryClient — для доступу до кешованих даних та їх оновлення, наприклад, ми використовуємо цей хук для оновлення списку дошок після

створення нової дошки, оновлення списку задач після зміни статусу задачі, оновлення деталей задачі після додавання коментаря тощо.

Ось приклад використання хука useQuery для отримання колонок проекту.

```
import { api } from "@/core/api";
import { Prisma } from "@prisma/client";
import { useQuery } from "@tanstack/react-query";
export type ColumnPayload = Prisma.ColumnsGetPayload<{
  include: { cards: true };
}>;
const getColumnFn = async (columnId: string) => {
  const { data } = await api.get<ColumnPayload>(`/api/columns/${columnId}`);
  return data;
};
interface UseColumnQueryOptions {
  initialData: ColumnPayload;
}
export const useColumnQuery = ({ initialData }: UseColumnQueryOptions) => {
  const query = useQuery<ColumnPayload>(["column", initialData.id], {
    queryFn: () => getColumnFn(initialData.id),
    initialData,
    refetchOnMount: false,
    refetchOnWindowFocus: false,
    refetchOnReconnect: false,
  });
  return query;
};
```

Клієнтська частина веб-додатку складається з наступних сторінок.

Логін — сторінка, яка дозволяє користувачеві увійти в свій обліковий запис в веб-додатку. Сторінка містить форму для введення електронної пошти та

пароля, також посилання для реєстрації нового облікового запису і посилання на відновлення паролю. Форма логіну зображена на рисунку 3.4.

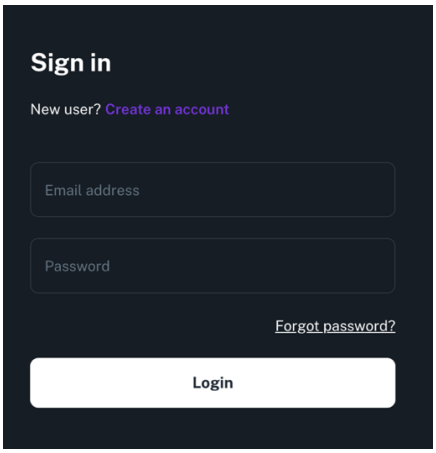

Рисунок 3.4 — Форма логіна

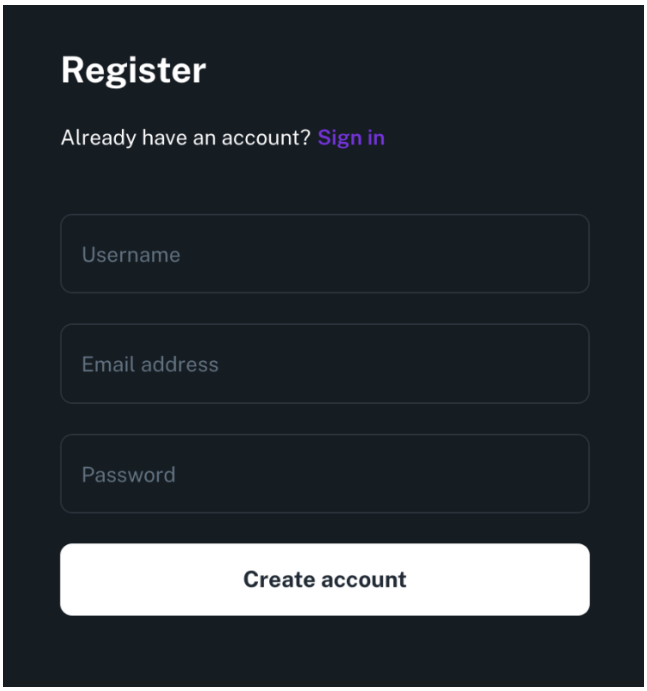

Рисунок 3.5 — Форма реєстрації

Сторінка проектів — сторінка, яка відображає список проектів, до яких належить користувач, а також можливість створити новий проект.

Сторінка проекту — сторінка, яка відображає деталі про один проект, до якого належить користувач, проектом виступає дошка з колонками зі списком

задач. За допомогою бібліотеки react-beautiful-dnd було реалізовано можливість перетягувати колонки та задачі. Сторінка проекту зображена на рисунку 3.6.

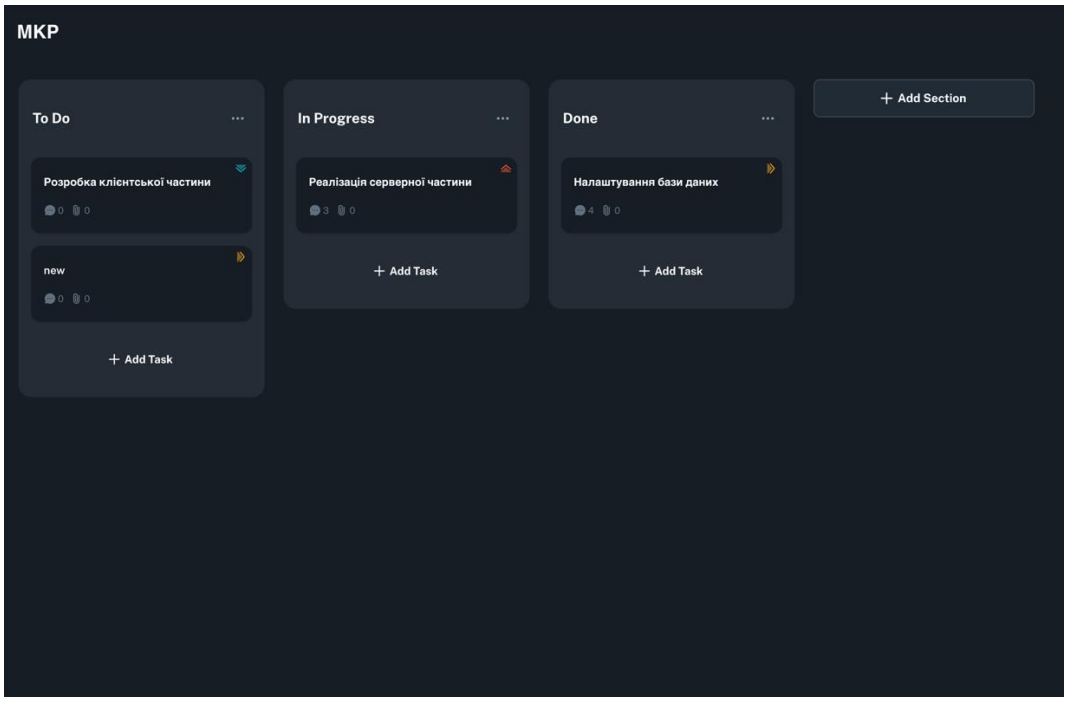

Рисунок 3.6 — Сторінка проекту

Сторінка — це компонент React, який експортується з файлу з розширенням .js, .jsx, .tsабо .tsx, що знаходиться в директорії app. Кожна директорія асоціюється із маршрутом (роутом) за назвою. Наприклад, сторінка app/login/page.ts буде доступна за адресою /login. Маршрут для сторінки app/boards/[id]/page.ts буде динамічним, тобто така сторінка буде доступна за адресами boards/1, boards/2і т.д. Отже, маршрутизація додатку Next.js побудована у вигляді дерева. Приклад зображений на рисунку 3.7.

За замовчуванням усі сторінки рендеруються заздалегідь (pre-rendering). Це призводить до кращої продуктивності та SEO.

Next.js використовує маршрутизатор на основі файлової системи, де папки використовуються для визначення маршрутів [17]. Маршрут — це один шлях із вкладених папок, що слідує за ієрархією файлової системи від кореневої папки до кінцевої кінцевої папки , яка містить page.jsфайл.

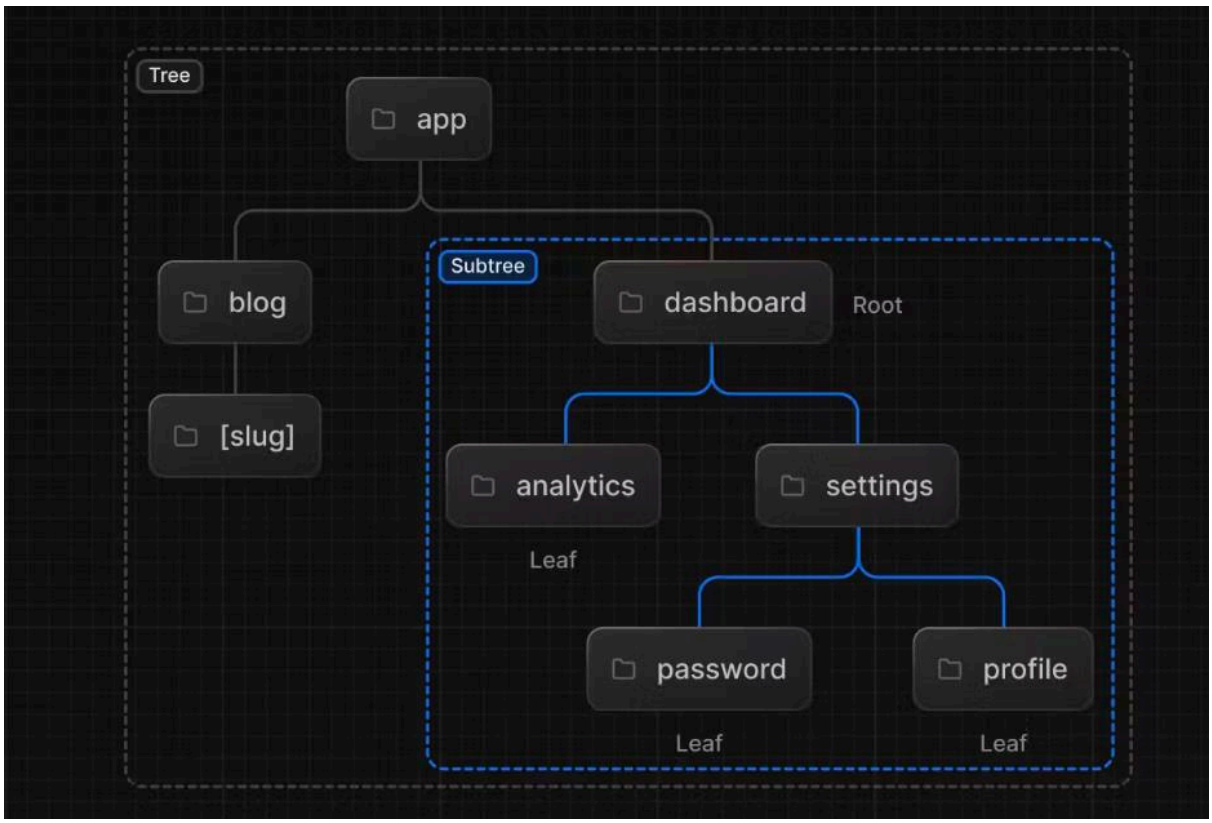

Рисунок 3.7 — Маршрутизація додатку Next.js

Розглянемо код сторінки проекту який знаходиться в файлі app/boards/[id].pages.ts.

```
export default async function BoardPage(props: PageProps) {
  const board = await prisma.boards.findUnique({
    where: {
      id: props.params.id,
    },
    include: {
      columns: {
        orderBy: {
          order: "asc",
        },
        include: {
          cards: {
            orderBy: {
```

```
            order: "asc",
          },
        },
      },
    },
  },
 });
 if (!board) {
  return notFound();
 }
 return (
  <BoardProvider>
   <ColumnsList board={board} />
   <CardDialog />
  </BoardProvider>
 );
}
```

Так як це серверний компонент, ми можемо звертатись до бази даних напряму із даної сторінки використовуючи prisma — це забезпечує швидке завантаження сторінки. SSR Рендеринг допомагає знизити навантаження на пристрій, який використовує програму, адже більшість операцій, що здійснюються в додатку, що стосуються його відображення, відбувається на сервері, а не на пристрої користувача (телефоні, планшеті, комп'ютері тощо) [17].

3.6 Інтеграція ChatGPT

ChatGPT — штучний інтелект, здатний генерувати людськоподібний текст на основі контексту та попередніх розмов . ChatGpt є сестринською моделлю до InstructGpt, яка навчена виконувати інструкцію в запиті та надавати детальну відповідь.

ChatGPT використовує архітектуру трансформера, яка дозволяє моделі ефективно вчитися від великих обсягів текстових даних та адаптуватися до різних сценаріїв спілкування. ChatGPT може виконувати різні завдання, пов’язані з мовою, такі як відповідати на запитання, надавати творчий натхнення, навчати щось нового тощо [18].

Ми інтегрували ChatGpt в наш веб-додаток з метою поліпшити якість та продуктивність роботи наших користувачів з задачами та проектами.

ChatGPT може допомогти користувачам нашого додатку написати описи для своїх задач, а також отримати рекомендації щодо їх виконання, планування, пріоритизації та делегування.

ChatGPT зможе допомогти користувачам:

— сформулювати чіткі та зрозумілі описи задач, які відповідають SMART-критеріям (Specific, Measurable, Achievable, Relevant, Time-bound);

— отримати корисні рекомендації щодо виконання задач, такі як кращі практики, поради, ресурси, приклади тощо;

— знайти альтернативні або креативні способи розв’язання задач, якщо вони застрягли або потребують нових ідей.

Інтеграція ChatGPT в веб-додаток має такі переваги:

— полегшує процес створення та виконання задач, надаючи користувачам готові шаблони опису та рекомендацій;

— стимулює творчість та навчання користувачів, показуючи їм різні можливості та способи розв'язання задач;

— збільшує продуктивність та ефективність роботи, зменшуючи час та зусилля, потрібні для формулювання та реалізації задач.

ChatGPT використовували як зовнішній сервіс, доступний через API. Для інтеграції ChatGPT у наш додаток виконано наступні кроки:

— створено обліковий запис на платформі OpenAI, яка надає доступ до ChatGPT та інших моделей мови;

— отримано ключ API, який дозволяє нам викликати ChatGPT з нашого додатку;

— додали функціонал у нашому додатку, який дозволяє користувачам вибирати опцію "Згенерувати опис" або "Отримати рекомендації" для будь-якої задачі;

— створили функції, які формують запити до ChatGPT на основі даних про задачу, таких як назва, тип, категорія, термін, відповідальний тощо;

— відправляли запити до ChatGPT через API, використовуючи ключ API та параметри, які визначають довжину, формат, стиль, рівень деталізації та мову відповіді;

— отримували відповіді від ChatGPT у форматі JSON, які містять згенерований текст для опису або рекомендацій для задачі;

— передавали відповіді до нашого додатку, який відображає їх у відповідних полях для опису або рекомендацій для задачі;

— давали можливість користувачам редагувати, зберігати або відхилити згенерований текст, а також надавати зворотний зв'язок про його якість та корисність.

Для роботи з ChatGPT API для чіткого формулювання опису і рекомендацій до задачі, ви можете використати наступні налаштування, запит, та промт.

Для налаштування виберіть модель gpt-3.5-turbo, яка є оптимізованою для чату версією GPT-3. Встановіть температуру 0.8, щоб отримати збалансовані та різноманітні відповіді. Встановіть максимальну кількість токенів 200, щоб обмежити довжину відповіді. Встановіть частоту 1, щоб отримати одну відповідь на кожен запит.

Створіть запит у форматі JSON, який містить масив повідомлень, які представляють історію розмови між користувачем та системою. Кожне повідомлення має два поля: role та content. Роль може бути user або system, а зміст — це текст повідомлення. Додайте до масиву останнє повідомлення з роллю user і змістом, який містить ваше запитання до системи. Наприклад, якщо ви хочете попросити систему допомогти вам сформулювати опис задачі, ви можете написати: "content": "Допоможи мені написати опис задачі."

Створіть промт, який дасть системі інструкції, як відповідати на ваш запит. Промт повинен бути включений до масиву повідомлень як перший елемент з роллю system. Промт повинен містити наступні елементи:

— роздільник між інструкцією та вхідним текстом;

— умови, яким повинен відповідати вхідний текст;

— структурований вихідний формат, який полегшує парсинг відповіді;

— приклади вхідно-вихідного відношення, які демонструють бажаний результат.

Інтеграція ChatGpt в наш веб-додаток не була простою та безпроблемною задачею. Ми зіткнулися з деякими викликами, які вимагали від нас технічних, організаційних та етичних рішень.

Сумісність з нашим стеком технологій. Наш веб-додаток був написаний на Next.js, в якості бази даних використано postgressql, supabase, prisma. Також використовував react-query, tailwind. Ці технології не були повністю сумісні з ChatGpt, яка була розроблена на Python та використовувала Azure AI для навчання та запуску моделі. Ми мусили створити місток між нашим додатком та ChatGpt, використовуючи REST API та GraphQL. Ми також мусили оптимізувати наш додаток для забезпечення швидкої та надійної взаємодії з ChatGpt.

Обмеження ресурсів та вартості. ChatGpt — це дуже потужна та складна модель, яка вимагає великої кількості обчислювальних ресурсів для своєї роботи. Ми мусили враховувати обмеження ресурсів та вартості, які накладалися на нас Azure AI. Ми мусили вибирати оптимальний рівень якості та кількості генерації тексту, щоб не перевищити наш бюджет та не знизити задоволення користувачів. Ми також мусили розробити механізми кешування та повторного використання тексту, щоб зменшити навантаження на ChatGpt та зекономити ресурси.

Якість та зміст генерації тексту. ChatGpt — це модель, яка навчена на великому обсязі текстових даних, але не гарантує, що весь згенерований текст буде правильним, змістовним та безпечним. Ми мусили перевіряти та фільтрувати згенерований текст, щоб видалити або виправити помилки, незв’язність, невідповідність, непристойність, образи, неправдивість та інші

проблеми, які могли б зашкодити якості нашого додатку та репутації наших користувачів. Ми також мусили враховувати контекст та потреби наших користувачів, щоб згенерований текст був релевантним, корисним та цікавим для них.

4 ТЕСТУВАННЯ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

4.1 Інструкція користувача програмної частини

У цьому підпункті наведено інструкцію користувача програмної частини розробленого веб-додатку для створення, організації та відстеження індивідуальних й командних задач і проектів. Для коректної роботи програмного додатку необхідно дотримуватися основних рекомендацій для її використання. Через те, що розроблена інформаційна система є веб-додатком, її не треба встановлювати користувачам. Для роботи з системою необхідно мати лише доступ в інтернет та браузер. Веб-додаток надає інтуїтивно зрозумілий інтерфейс та функціонал для користувачів системи.

Для початку роботи з веб-додатком користувач повинен зареєструватися або увійти за допомогою свого електронного листа та пароля. Для цього потрібно перейти на головну сторінку sign-in, а для реєстрації на сторінку register. На екрані з’явиться вікно, де можна ввести свої дані. Сторінка авторизації та реєстрації зображені на рисунках 4.1 і 4.2.

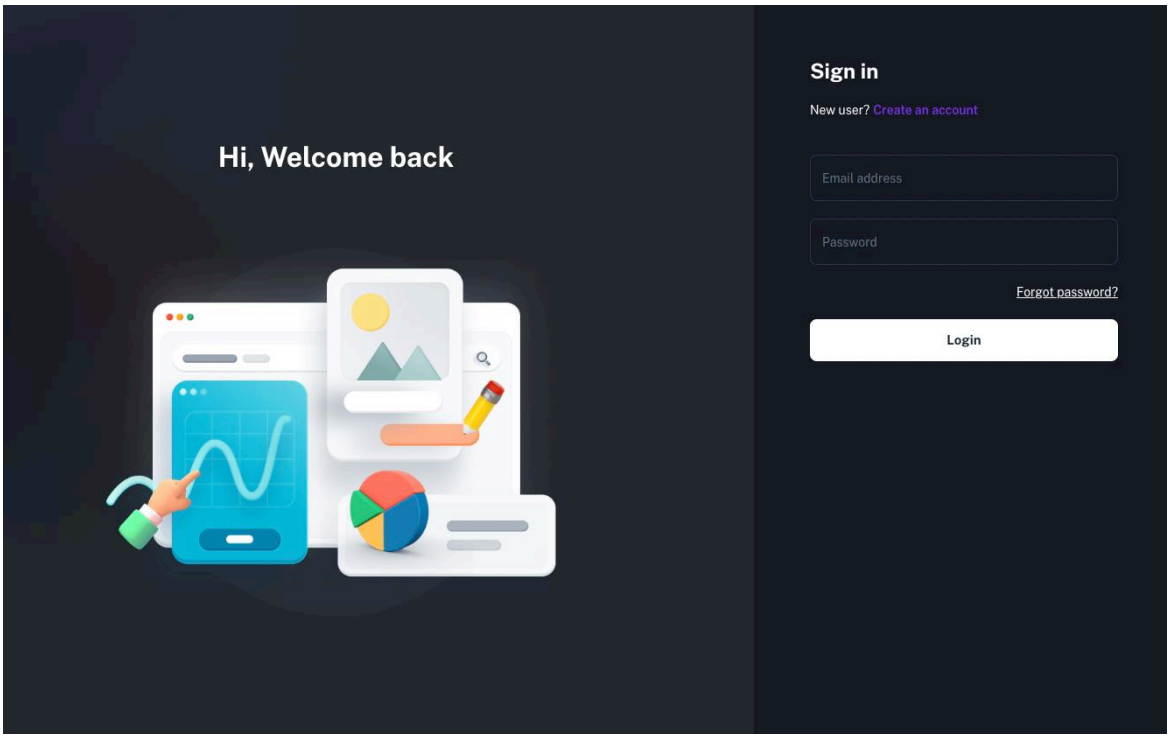

Рисунок 4.1 — Сторінка авторизації

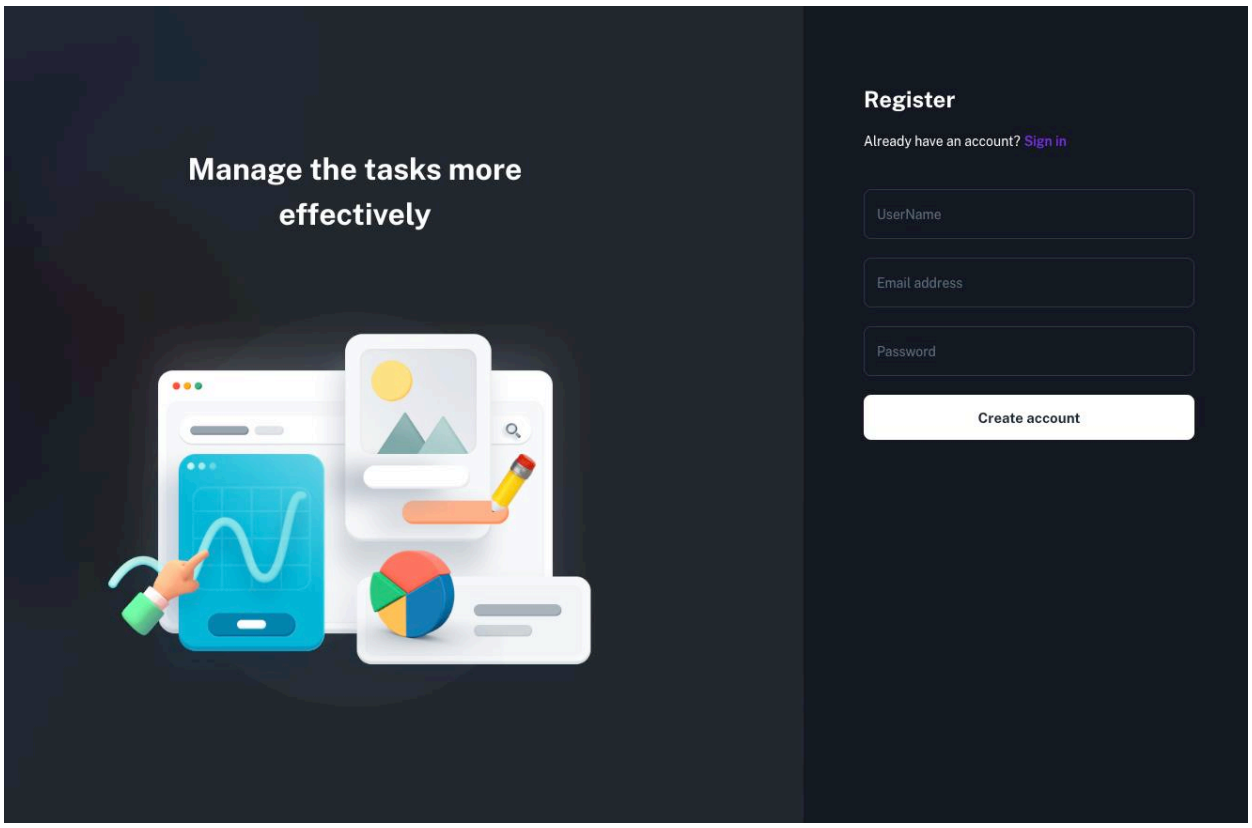

Рисунок 4.2 — Сторінка реєстрації

Для створення нового проекту користувач повинен натиснути на кнопку “Add new project” у верхньому правому куті сторінки зі своїми проектами. На екрані з’явиться вікно, де можна ввести назву проекту, проекту та додати учасників проекту. Учасників проекту можна додавати за їх електронними листами, якщо вони вже зареєстровані в веб-додатку, або надіслати їм запрошення на реєстрацію. Після заповнення всіх полів потрібно натиснути на кнопку “Create” для створення проекту. Проект буде додано до списку проектів користувача.

Для того, щоб переглянути або редагувати дошку, натисніть на її назву на головній сторінці.

Ви побачите сторінку дошки, де ви можете додавати, редагувати, видаляти, переміщувати, призначати, коментувати, фільтрувати, сортувати, шукати, списки, задачі тощо. Сторінка проекту (дошки) зображено на рисунку 4.4.

Для того, щоб додати новий список, натисніть на кнопку “+ Add section” у верхній частині сторінки дошки.

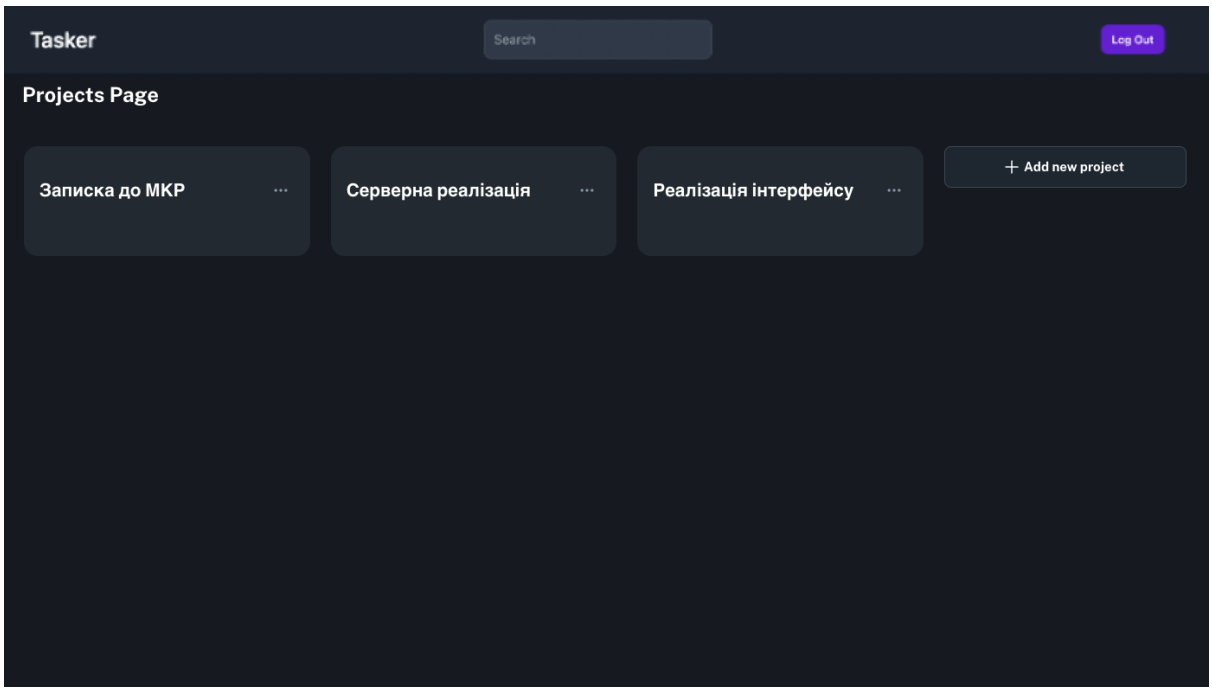

Рисунок 4.3 — Сторінка проектів

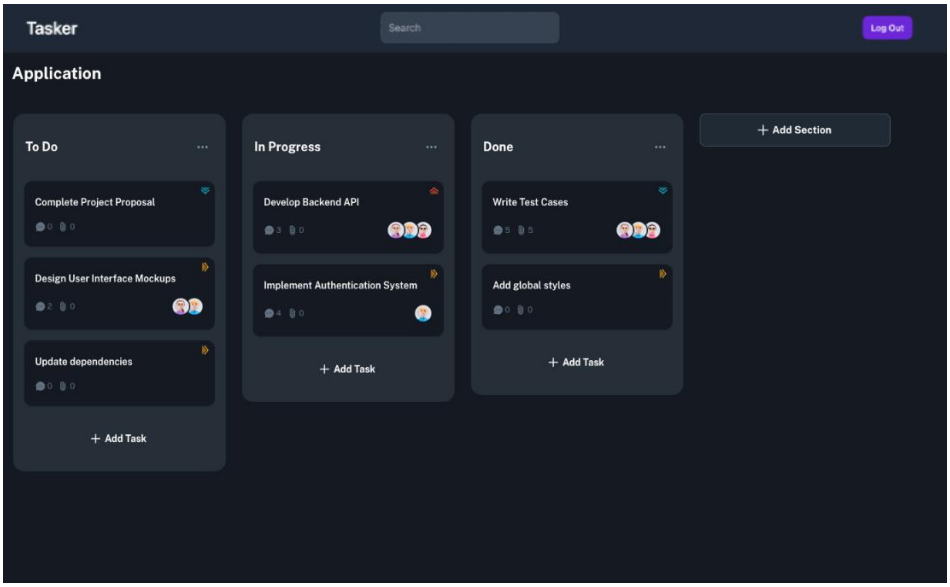

Рисунок 4.4 — Сторінка проекту

Для того, щоб додати нову картку, натисніть на кнопку “+ Add task” у нижній частині списку. Введіть назву картки, натисніть на кнопку “Add task”.

Для того, щоб переглянути або редагувати картку, натисніть на її назву або іконку у списку. Ви побачите вікно картки, де ви можете переглянути детальну інформацію про картку, призначити відповідальних за цю задачу, виставити дедлайн, змінити пріоритет, назву, статус, опис, прикріпити файл, додати

комент, та скористатись підказкою від ChatGPT. Сторінка задачі зображена на рисунку 4.5

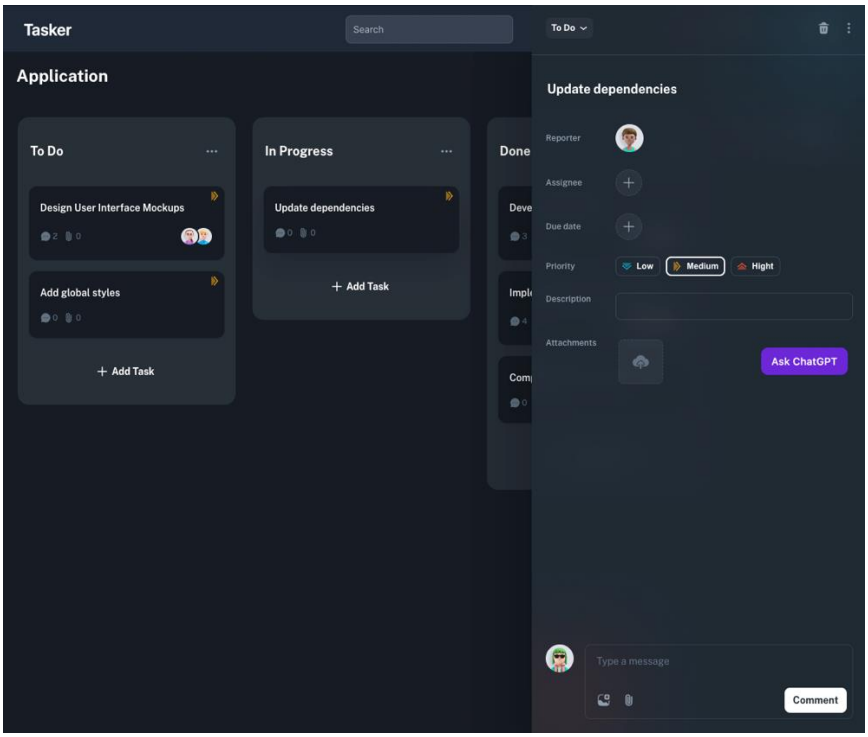

Рисунок 4.5 — Сторінка задачі

Для використання моделі генерації тексту ChatGPT для написання опису та рекомендацій для задач користувач повинен вибрати задачу зі свого списку та натиснути на кнопку “Ask ChatGPT” у верхньому правому куті сторінки з деталями задачі. На екрані з’явиться вікно, де можна побачити текст, який згенерувала модель ChatGPT на основі назви задачі. Текст може містити опис задачі, рекомендації щодо виконання та поліпшення задачі, а також запитання до користувача. Користувач може скопіювати текст, який згенерувала модель ChatGPT, та вставити його у поле опису задачі, якщо він вважає його корисним та релевантним.

4.2 Результати тестування коректності роботи з проектами та задачами

Для тестування коректності роботи з проектами та задачами було створено три тестові проекти: “Магістерська робота”, “Особисті справи” та “Командний проект”. Кожен проект мав чотири колонки: “To do”, “In progress”, “Review” та

“Done”. Кожна колонка мала кілька задач з різними параметрами, такими як назва, опис, термін виконання, відповідальний, пріоритет, коментарі тощо. Для кожної операції з проектами та задачами було перевірено, чи відповідає результат очікуваному. Результати тестування коректності роботи з проектами та задачами показано в таблиці 4.1.

Таблиця 4.1. Результати тестування коректності роботи з проектами та задачами

Операція	**Очікуваний результат**	**Фактичний результат**	**Статус**
Створення проекту	Проект створено в базі даних, користувач отримує ідентифікатор та посилання на проект	Проект створено в базі даних, користувач отримує ідентифікатор та посилання на проект	Успішно
Перегляд проекту	Користувач бачить дошку з колонками та задачами проекту	Користувач бачить дошку з колонками та задачами проекту	Успішно
Редагування проекту	Користувач може змінювати назву, учасників та інші параметри проекту, зміни зберігаються в базі даних	Користувач може змінювати назву, учасників та інші параметри проекту, зміни зберігаються в базі даних	Успішно
Видалення проекту	Проект видаляється з бази даних, користувач отримує підтвердження про видалення	Проект видаляється з бази даних, користувач отримує підтвердження про видалення	Успішно
Додавання колонки	Користувач може додавати нову колонку до дошки , вказуючи її назву, колонка зберігається в базі даних	Користувач може додавати нову колонку до дошки , вказуючи її назву, колонка зберігається в базі даних	Успішно
Переміщення колонки	Користувач може пересувати колонку вліво або вправо, змінюючи її порядок, зміни зберігаються в базі даних	Користувач може пересувати колонку вліво або вправо, змінюючи її порядок, зміни зберігаються в базі даних	Успішно
Редагування колонки	Користувач може змінювати назву колонки, зміни зберігаються в базі даних	Користувач може змінювати назву колонки, зміни зберігаються в базі даних	Успішно

Продовження таблиці 4.1

Видалення колонки	Колонка видаляється з бази даних разом із задачами в ній, користувач отримує підтвердження про видалення	Колонка видаляється з бази даних разом із задачами в ній, користувач отримує підтвердження про видалення	Успішно
Додавання задачі	Користувач може додавати нову задачу до колонки, вказуючи її назву, опис, термін виконання, відповідального, пріоритет, коментарі тощо, задача зберігається в базі даних	Користувач може додавати нову задачу до колонки, вказуючи її назву, опис, термін виконання, відповідального, пріоритет, коментарі тощо, задача зберігається в базі даних	Успішно
Переміщення задачі	Користувач може пересувати задачу з однієї колонки в іншу, змінюючи її статус, зміни зберігаються в базі даних	Користувач може пересувати задачу з однієї колонки в іншу, змінюючи її статус, зміни зберігаються в базі даних	Успішно
Редагування задачі	Користувач може змінювати параметри задачі, зміни зберігаються в базі даних	Користувач може змінювати параметри задачі, зміни зберігаються в базі даних	Успішно
Видалення задачі	Задача видаляється з бази даних, користувач отримує підтвердження про видалення	Задача видаляється з бази даних, користувач отримує підтвердження про видалення	Успішно
Робота з ChatGPT	Користувач отримає опис та рекомендації до задачі згенеровані за допомогою штучного інтелекту	Користувач отримає опис та рекомендації до задачі згенеровані за допомогою штучного інтелекту	Успішно

З таблиці 4.1 видно, що веб-додаток правильно виконує всі операції з проектами та задачами, які було перевірено.

4.3 Методика перевірки працездатності та тестування програмних компонентів.

У цьому розділі описано методику перевірки працездатності та тестування програмних компонентів веб-додатку для створення, організації та відстеження індивідуальних й командних задач і проектів. Методика базується на

застосуванні різних видів та типів тестування, які дозволяють виявити та усунути помилки та дефекти в програмному коді, інтерфейсі, функціональності та безпеці веб-додатку.

Методика складається з наступних етапів.

На етапі планування тестування визначаються цілі та завдання тестування, вибираються необхідні види та типи тестування, розробляється план тестування, в якому вказуються тестові сценарії, тестові дані, критерії прийнятності, очікувані результати, необхідні ресурси та інструменти для тестування.

Розробка тестових сценаріїв. На цьому етапі складаються тестові сценарії, які описують конкретні дії, які необхідно виконати для перевірки певних аспектів веб-додатку. Тестові сценарії мають бути чіткими, зрозумілими, повними та однозначними. Тестові сценарії можуть бути розроблені на основі вимог до веб-додатку, специфікацій, дизайну, а також на основі досвіду та інтуїції тестувальника.

На етапі підготовки тестових даних готуються дані, які будуть використовуватися для виконання тестових сценаріїв. Тестові дані мають бути репрезентативними, різноманітними, достатніми та валідними. Тестові дані можуть бути синтетичними, тобто створеними штучно, або реальними, тобто взятими з існуючих джерел. Тестові дані мають враховувати різні ситуації, які можуть виникнути під час роботи веб-додатку, такі як нормальні, межові, негативні, екстремальні тощо.

На етапі тестових сценаріїв запускаються сценарії з використанням тестових даних та інструментів для тестування. Виконання тестових сценаріїв може бути ручним, тобто здійснюватися безпосередньо тестувальником, або автоматизованим, тобто здійснюватися за допомогою спеціальних програм, які імітують дії користувача. Виконання тестових сценаріїв має бути документованим, тобто фіксуватися результати та відхилення від очікуваних.

На етапі аналізу тестування проводиться аналіз, виявляються та класифікуються помилки та дефекти, які були знайдені під час тестування. Помилки та дефекти мають бути описані, пріоритизовані, призначені

відповідальним особам для їх виправлення. Також має бути проведена оцінка якості веб-додатку на основі критеріїв прийнятності та результатів тестування [19].

Види та типи тестування, які застосовуються для перевірки працездатності та тестування програмних компонентів веб-додатку, описані нижче.

Функціональне тестування — це вид тестування, який перевіряє, чи відповідає функціональність веб-додатку вимогам та очікуванням користувачів. Функціональне тестування перевіряє такі аспекти веб-додатку, як:

— наявність та коректність роботи основних та додаткових функцій, таких як створення, редагування, видалення, переміщення, пошук, співпраця тощо;

— відповідність веб-додатку стандартам та правилам бізнес-логіки, таким як валідація даних, обробка помилок, обмеження доступу, права доступу тощо;

— сумісність веб-додатку з різними браузерами, операційними системами, пристроями, роздільними здатностями тощо;

— інтеграція веб-додатку з іншими системами, сервісами, додатками тощо.

Функціональне тестування може бути розбито на наступні типи.

Модульне тестування — це тип тестування, який перевіряє працездатність окремих програмних модулів, які є найменшими самостійними частинами програмного коду. Модульне тестування дозволяє перевірити, чи правильно працюють окремі функції, класи, методи, змінні тощо. Модульне тестування зазвичай виконується розробниками за допомогою спеціальних бібліотек та фреймворків, таких як Jest, Mocha, Chai, Enzyme тощо.

Інтеграційне тестування — це тип тестування, який перевіряє, як взаємодіють між собою різні модулі, компоненти, системи, сервіси, додатки тощо. Інтеграційне тестування дозволяє перевірити, чи правильно передаються та обробляються дані, чи виконуються очікувані дії, чи відсутні конфлікти та залежності між різними елементами. Інтеграційне тестування може бути

виконане ручним або автоматизованим способом за допомогою таких інструментів, як Cypress, Selenium, Puppeteer, Supertest тощо.

Системне тестування — це тип тестування, який перевіряє, як працює веб-додаток як цілісна система, чи відповідає він загальним вимогам та очікуванням користувачів. Системне тестування перевіряє такі аспекти веб-додатку, як продуктивність, надійність, стабільність, безпека, зручність, сумісність тощо. Системне тестування може бути виконане ручним або автоматизованим способом за допомогою таких інструментів, як LoadRunner, JMeter, Postman, OWASP ZAP тощо

Нетестоване тестування — це вид тестування, який перевіряє, чи відповідає веб-додаток нефункціональним вимогам, таким як зовнішній вигляд, дизайн, стиль, естетика, емоційний вплив тощо. Нетестоване тестування перевіряє наступні аспекти веб-додатку.

Зручність використання — це міра того, наскільки легко, зрозуміло, приємно та задовільно користувач може використовувати веб-додаток. Зручність використання перевіряється за допомогою таких методів, як експертна оцінка, тестування з користувачами, аналіз відгуків, аналітика тощо.

Доступність — це міра того, наскільки веб-додаток доступний та зрозумілий для різних категорій користувачів, особливо для тих, хто має певні обмеження, такі як вік, мова, культура, освіта, фізичні або психічні вади тощо. Доступність перевіряється за допомогою таких методів, як аудит, тестування з користувачами, використання стандартів та кращих практик, таких як WCAG, WAI-ARIA, ADA тощо [21].

Тестування програмного продукту є важливим етапом розробки веб-додатку, який дозволяє перевірити його функціональність, надійність, продуктивність та якість. Тестування також допомагає виявити та усунути помилки, що можуть негативно впливати на користувацький досвід та задоволення від роботи з додатком.

У цьому розділі ми описуємо методи, види, інструменти та результати тестування нашого веб-додатку для створення, організації та відстеження індивідуальних й командних задач і проектів.

Ми використовували два основні методи тестування: статичне та динамічне.

4.4 Тестування програмного продукту

Статичне тестування полягає в аналізі програмного коду, документації, специфікації та інших артефактів розробки без їх виконання. Статичне тестування дозволяє виявити синтаксичні, логічні, структурні та інші помилки на ранніх етапах розробки, що зменшує час та витрати на їх виправлення. Статичне тестування включає наступні види перевірок.

Рев'ю коду — це процес перегляду та оцінки коду розробниками або іншими фахівцями з метою виявлення та усунення помилок, порушень стандартів, поганої практики та інших проблем. Рев'ю коду може бути формальним або неформальним, індивідуальним або груповим, ручним або автоматизованим.

Аналіз коду — це процес застосування спеціальних інструментів для аналізу коду з метою виявлення та усунення помилок, вразливостей, недоліків та інших проблем. Аналіз коду може бути статичним або динамічним, залежно від того, чи виконується код під час аналізу. Аналіз коду допомагає підвищити якість, безпеку, продуктивність та сумісність коду.

Аудит коду — це процес перевірки відповідності коду встановленим стандартам, правилам, критеріям та вимогам. Аудит коду дозволяє забезпечити однорідність, читабельність, зрозумілість та підтримуваність коду. Аудит коду може бути ручним або автоматизованим, внутрішнім або зовнішнім, періодичним або постійним.

Динамічне тестування полягає в виконанні програмного продукту в певних умовах та спостереженні за його поведінкою та результатами. Динамічне тестування дозволяє перевірити функціональність, надійність, продуктивність та

якість програмного продукту в реальних або модельованих сценаріях. Динамічне тестування включає наступнівиди перевірок.

Модульне тестування — це процес перевірки окремих програмних модулів, компонентів або функцій на відповідність їх специфікації та очікуваній поведінці. Модульне тестування дозволяє виявити та усунути помилки на низькому рівні абстракції, що полегшує їх локалізацію та виправлення. Модульне тестування зазвичай виконується розробниками на етапі кодування.

Інтеграційне тестування — це процес перевірки взаємодії та співпраці різних програмних модулів, компонентів або функцій, що об’єднуються в групи або підсистеми. Інтеграційне тестування дозволяє виявити та усунути помилки на середньому рівні абстракції, що виникають при комбінуванні окремих елементів програмного продукту. Інтеграційне тестування зазвичай виконується тестувальниками на етапі інтеграції.

Системне тестування — це процес перевірки повної або часткової системи програмного продукту на відповідність її вимогам, очікуваній поведінці та призначенню. Системне тестування дозволяє виявити та усунути помилки на високому рівні абстракції, що впливають на функціональність, надійність, продуктивність та якість системи в цілому. Системне тестування зазвичай виконується тестувальниками на етапі тестування.

Приймальне тестування — це процес перевірки готового програмного продукту на відповідність очікуванням та потребам замовника або користувача. Приймальне тестування дозволяє виявити та усунути помилки, що можуть заважати ефективному використанню програмного продукту в реальних умовах. Приймальне тестування зазвичай виконується замовником або користувачем на етапі приймання [20].

Ми використовували різні види тестування, залежно від мети, об’єкта, критеріїв та рівня тестування.

Функціональне тестування — це процес перевірки функціональності програмного продукту, тобто його здатності виконувати очікувані функції відповідно до вимог. Функціональне тестування дозволяє виявити та усунути

помилки, що порушують логіку, алгоритми, інтерфейси, валідацію, обробку даних та інші аспекти функціональності. Функціональне тестування може бути модульним, інтеграційним, системним або приймальним, залежно від рівня тестування.

Нефункціональне тестування — це процес перевірки нефункціональних характеристик програмного продукту, тобто його здатності задовольняти невимірювані вимоги, такі як надійність, продуктивність, безпека, сумісність, масштабованість, користувацький досвід та інші. Нефункціональне тестування дозволяє виявити та усунути помилки, що впливають на якість, стабільність, швидкість, захищеність, сумісність та інші аспекти нефункціональності. Нефункціональне тестування може бути системним або приймальним, залежно від рівня тестування.

Регресивне тестування — це процес повторного виконання вже проведених тестів після внесення змін в програмний продукт, таких як виправлення помилок, додавання функціональності, оновлення версії тощо. Регресивне тестування дозволяє перевірити, чи не виникли нові помилки або не порушена попередня функціональність внаслідок змін. Регресивне тестування може бути модульним, інтеграційним, системним або приймальним, залежно від рівня тестування [21].

Експлораторне тестування — це процес невизначеного та творчого тестування програмного продукту без попереднього планування, сценаріїв, кейсів або очікувань. Експлораторне тестування дозволяє виявити та усунути помилки, що можуть уникнути стандартних тестів, а також перевірити користувацький досвід, інтуїтивність, зручність та інші аспекти програмного продукту. Експлораторне тестування зазвичай виконується тестувальниками або користувачами на етапі тестування або приймання. Ми використовували різні інструменти тестування, залежно від методу, виду, об’єкта та рівня тестування.

Jest — це фреймворк для модульного тестування JavaScript-коду, який дозволяє писати, запускати та відлагоджувати тестові сценарії, а також отримувати звіти про покриття коду тестами. Jest підтримує тестування React-

компонентів, Next.js-сторінок, Node.js-серверів та інших елементів нашого веб-додатку.

Cypress — це фреймворк для інтеграційного та системного тестування веб-додатків, який дозволяє писати, запускати та відлагоджувати тестові сценарії, а також отримувати звіти про результати тестування. Cypress підтримує тестування функціональності, продуктивності, сумісності, безпеки та інших характеристик нашого веб-додатку.

ESLint — це інструмент для статичного аналізу та аудиту TypeScript-коду, який дозволяє виявляти та виправляти синтаксичні, логічні, структурні та інші помилки, а також перевіряти відповідність коду встановленим стандартам, правилам, критеріям та вимогам. ESLint допомагає підвищити якість, безпеку, продуктивність та сумісність нашого JavaScript-коду.

Prettier — це інструмент для форматування та вирівнювання TypeScript-коду, який дозволяє забезпечити однорідність, читабельність, зрозумілість та підтримуваність коду. Prettier допомагає покращити стиль, зовнішній вигляд та якість нашого JavaScript-коду.

5 ЕКОНОМІЧНА ЧАСТИНА

Щоб науково-технічна розробка отримала прийняття та успішно впроваджувалася, важливо, щоб вона відповідала поточним вимогам науково-технічного прогресу та враховувала економічні аспекти. Таким чином, оцінка економічної ефективності результатів науково-дослідної роботи є невід'ємною частиною процесу.

Дослідження в магістерській роботі, присвяченій розробці та вивченню "Веб-додаток для створення, організації та відстеження індивідуальних й командних задач і проектів", відноситься до науково-технічних робіт, спрямованих на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі самої роботи), відзначаючи так звану комерціалізацію науково-технічної розробки. Цей напрямок розглядається як пріоритетний, оскільки отримані результати можуть бути корисними для різноманітних зацікавлених сторін, приносячи економічні вигоди. Проте для успішної реалізації цього процесу важливо знайти потенційного інвестора, зацікавленого у втіленні даного проекту, та переконати його в обґрунтованості вкладання інвестицій в цей конкретний проект.

Для цього визначені наступні етапи виконання робіт:

1. Проведено комерційний аудит науково-технічної розробки, що включає в себе визначення науково-технічного рівня та комерційного потенціалу.

2. Розраховані витрати на реалізацію науково-технічної розробки.

3. Проведено розрахунок економічної ефективності науково-технічної розробки в разі її впровадження та комерціалізації потенційним інвестором, а також обґрунтовано економічну доцільність комерціалізації для інвестора.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою "Веб-додаток для створення, організації та відстеження індивідуальних й

командних задач і проектів" є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки back-end частини плагіна для WordPress з функціональними можливостями управління рекламою на веб-ресурсі з метою покращення ефективності та результативності рекламних зусиль.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [23].

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевіре но працезда тність
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властиво сті
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуат аційні витрати значно
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивн
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкуре нтів немає
Практична здійсненність					

8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупност
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці. Для оцінки науково-технічного рівня і комерційного потенціалу розробки експертами було запрошено трьох незалежних експертів з Вінницького національного технічного університету кафедри «Обчислювальної техніки»: Крупельницький Леонід Віталійович, кандидат технічних наук, доцент; Войцеховська Олена Валеріївна кандидат технічних наук, доцент, Городецька Оксана Степанівна кандидат технічних наук, доцент.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	Крупельницький Леонід Віталійович	Городецька Оксана Степанівна	Войцеховська Олена Валеріївна кандидат технічних наук, доцент,
	Бали:		
1. Технічна здійсненність концепції	3	3	3
2. Ринкові переваги (наявність аналогів)	2	2	2
3. Ринкові переваги (ціна продукту)	4	4	4
4. Ринкові переваги (технічні властивості)	2	1	2
5. Ринкові переваги (експлуатаційні витрати)	4	4	4
6. Ринкові перспективи (розмір ринку)	4	3	4
7. Ринкові перспективи (конкуренція)	2	1	1
8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	4	4	4
10. Практична здійсненність (необхідність нових матеріалів)	4	4	4
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	4	4	4
Сума балів	$СБ_1$=40	$СБ_2$=37	$СБ_3$=39
Середньоарифметична сума балів $СБ_c$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{40+37+39}{3} = 38.66$		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3 [23].

Таблиця 5.3 — Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою "Веб-додаток для створення, організації та відстеження індивідуальних й командних задач і проектів" становить 38,6 балів, що, відповідно до таблиці 5.3 рівень комерційного потенціалу розробки вище середнього, що свідчить про комерційну важливість проведення даних досліджень.

Результатом магістерської кваліфікаційної роботи є веб-додаток для створення, організації та відстеження індивідуальних й командних задач і проектів

Веб-додаток може бути використаний для ефективного управління робочим процесом в різних організаціях, зокрема, в компаніях, стартапах, командах розробників, може бути використаний для підвищення продуктивності та ефективності роботи користувачів

5.1 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему "Веб-додаток для створення, організації та відстеження індивідуальних й командних задач і проектів", під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.1.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників.

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [23]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (5.1)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=21 дні.

$$З_о = 18000 \cdot 5 / 21 = 4091 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.4 — Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	18000	818,2	5	4091
Інженер-програміст	15000	681,8	40	27273
Всього				31364

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.2)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод} = (31364) \cdot 11 / 100\% = 3450$ *грн.*

5.1.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.3)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

$З_н$ = (31364+3450) · 22 / 100% = 7659 грн.

5.1.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою "Веб-додаток для створення, організації та відстеження індивідуальних й командних задач і проектів".

Витрати на матеріали ($М$), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (5.4)$$

де $Н_j$ – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

$В_j$ – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.5 — Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Вартість витраченого матеріалу, грн
Папір А4	180	1	180
Ручка	30	2	60
Диск оптичний OPTIMA CD	14	2	28
Flesh-пам'ять 64	400	1	400
Всього			668
З врахуванням коефіцієнта транспортування			734,8

5.1.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему "Веб-додаток для створення, організації та відстеження індивідуальних й командних задач і проектів" відсутні.

5.1.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення в роботі відсутні.

5.1.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення в роботі відсутні.

5.1.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.5)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (35000 \cdot 1) / (2 \cdot 12) = 1458{,}33$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.10 — Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Комп'ютер	35000	2	1	1458,33
Приміщення лабораторії	270000	20	1	1125,00
Всього				2583,33

5.1.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (5.6)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 7,5 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,25 \cdot 300,0 \cdot 7,5 \cdot 0,5 / 0,8 = 351,56$ *грн.*

5.1.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему "Веб-додаток для створення, організації та відстеження індивідуальних й командних задач і проектів" належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження

на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень. В нашій роботі відсутні.

5.1.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» відсутні.

5.1.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.7)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів} = 50\%$.

$I_в = (31364) \cdot 50 / 100\% = 15681{,}82$ грн.

5.1.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}\ , \qquad (5.8)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$В_{нзв}$ = (31364) · 100 / 100% = 31363,64 грн.

Витрати на проведення науково-дослідної роботи на тему "Веб-додаток для створення, організації та відстеження індивідуальних й командних задач і проектів" розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + А_{обл} + B_е + B_{св} + B_{сп} + І_в + B_{нзв}\ . \qquad (5.9)$$

$В_{заг}$= 31364+3450+7659+734,8+2583,33+351,56+15681,82+31363,64 = 93187,79 грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}\ , \qquad (5.10)$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,7.

ЗВ = 93187,79 / 0,7 = 133125,41 грн.

5.2 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою "Веб-додаток для створення, організації та відстеження індивідуальних й командних задач і проектів" передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN – збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

N – кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 1 особа

$Ц_б$ – вартість послуги у році до впровадження інформаційної системи, приймемо 5000,00 грн;

$\pm\Delta Ц_о$ – зміна вартості послуги від впровадження результатів, приймемо зростання на 500,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою **[**23**]**:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (5.11)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо $\rho = 40\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (1 \cdot 1000 + 5000 \cdot 200) \cdot 0,83 \cdot 0,4 \cdot (1 - 0,18/100\%) = 299031,79$ грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (1 \cdot 500 + 5000 \cdot (500 + 300)) \cdot 0{,}83 \cdot 0{,}4 \cdot (1 - 0{,}18/100\%) = 478814{,}2$ грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (1 \cdot 500 + 5000 \cdot (500 + 300 + 200)) \cdot 0{,}83 \cdot 0{,}4 \cdot (1 - 0{,}18/100\%) = 598392{,}75$ грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \tag{5.12}$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=18%;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП = 299031{,}79 / (1+0{,}18)^1 + 478814{,}2 / (1+0{,}18)^2 + 598392{,}75 / (1+0{,}18)^3 =$

$= 929605{,}78$ грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{5.13}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 133125,41грн.

$PV = k_{інв} \cdot ЗВ = 2 \cdot 133125{,}41 = 266250{,}82$ грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{5.14}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 929605,78 грн;

PV – теперішня вартість початкових інвестицій, 266250,82 грн.

$E_{абс} = ПП - PV = 929605{,}78 — 266250{,}82 = 663354{,}96$ грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.18}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, грн;

PV – теперішня вартість початкових інвестицій, грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+663354{,}96 / 266250{,}82)^{1/3} - 1 = 0{,}82.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f\,, \tag{5.19}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,1+0,25 = 0,35 < 0,82 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою "Веб-додаток для створення, організації та відстеження індивідуальних й командних задач і проектів" доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}\,, \tag{5.20}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,82 = 1,2 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Висновки до розділу

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою "Веб-додаток для створення, організації та відстеження індивідуальних й командних задач і проектів" становить 39 балів, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

Також термін окупності становить 1,2 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою "Веб-додаток для створення, організації та відстеження індивідуальних й командних задач і проектів".

ВИСНОВКИ

У даній магістерській кваліфікаційній роботі було досліджено теоретичні та практичні аспекти, розроблено та реалізовано веб-додаток для створення, організації та відстеження індивідуальних й командних задач і проектів. Метою роботи було створити функціональний, зручний, швидкий і атрактивний веб-додаток, який би використовував сучасні технології та інтегрував штучний інтелект для підтримки користувачів у вирішенні їх задач.

Для досягнення цієї мети було виконано наступні завдання:

— проведено аналіз сучасних веб-додатків для управління проектами та визначено їх переваги та недоліки;

— проведено огляд сучасного стану по темі роботи, в якому було розглянуто існуючі підходи, теорії і практики в області веб-розробки, управління проектами і задачами;розроблено методологію дослідження, в якій було вибрано і обґрунтовано методи збору, аналізу і інтерпретації даних, а також практичні аспекти дослідження;

— сформульовано вимоги до функціональності та дизайну веб-додатку, який би задовольняв потреби різних користувачів;

— розроблено архітектуру та інтерфейс веб-додатку, який дозволяє створювати, редагувати, переміщувати, видаляти, призначати, коментувати, фільтрувати та сортувати задачі на дошках та списках;

— реалізовано веб-додаток, який використовує сучасні технології для забезпечення швидкості, надійності, безпеки та масштабованості;

— інтегровано ChatGPT для генерації опису та рекомендацій для тасків на основі їх назви, типу, статусу, терміну, пріоритету та інших параметрів;

— проведено тестування та оцінку веб-додатку за різними критеріями, такими як функціональність, зручність, продуктивність, сумісність, безпека та інші.

Веб-додаток базується на сучасних технологіях, таких як Next.js, PostgreSQL, Supabase, Prisma, React-Query, Tailwind та ChatGPT. Він має

інтуїтивний інтерфейс користувача, що дозволяє легко створювати, редагувати, переміщувати, призначати та закривати задачі. Веб-додаток також має функцію генерації опису та рекомендацій для задач, яка використовує нейромережеву модель ChatGPT для створення змістовного та корисного тексту.

Основні результати та висновки, отримані в ході дослідження, можна сформулювати наступним чином.

Веб-додаток є ефективним інструментом для планування та управління проектами, який може застосовуватися як індивідуально, так і в командній роботі. Веб-додаток допомагає підвищити продуктивність, організованість, мотивацію та співпрацю користувачів. Він має простий, інтуїтивний і атрактивний дизайн, який відповідає сучасним стандартам веб-розробки, а також адаптується до різних пристроїв, роздільних здатностей і браузерів, що забезпечує користувачам комфортне і приємне використання додатку.

Веб-додаток використовує сучасні та надійні технології, які забезпечують швидкість, стабільність, безпеку та масштабованість розробленого продукту. Веб-додаток також використовує переваги серверного рендерингу, статичної генерації, оптимізації запитів, адаптивного дизайну та штучного інтелекту.

Веб-додаток має унікальну функцію генерації опису та рекомендацій для задач, яка використовує потужну нейромережеву модель ChatGPT. Ця функція дозволяє автоматично створювати текст, який відповідає контексту та цілям задачі, а також надавати користувачам поради щодо її виконання.

Робота має перспективи подальшого вдосконалення та розвитку, а саме:

— веб-додаток може бути інтегрований з іншими сервісами, такими як електронна пошта, календар, соціальні мережі, хмарні сховища, тощо;

— веб-додаток може бути удосконалений за допомогою додавання нових функцій, модулів, інтерфейсів або технологій;

— веб-додаток може бути розширений за допомогою додавання нових типів задач, проектів, користувачів, ролей, прав, тощо.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Шведа Н. Система управління проектами в україні / Н. Шведа // Тернопіль, 25 листоп. 2015 р. – Тернопіль, 2015. – С. 246–247.
2. Смирнов І. Що таке веб-додаток? [Електронний ресурс] / Ілля Смирнов // https://webcase.com.ua. – Режим доступу: https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/.
3. Особливості web-додатків [Електронний ресурс] // sites.znu.edu.ua. – Режим доступу: https://sites.znu.edu.ua/webprog/lect/1191.ukr.html.
4. Цільова аудиторія: як її визначити й аналізувати? [Електронний ресурс] // YouScan. – Режим доступу: https://youscan.io/ua/blog/target-audience-analysis-aim-accurately/.
5. ТОП 7 інструментів управління проектами у 2023 році [Електронний ресурс] // Worksection. – Режим доступу: https://worksection.com/ua/blog/5-project-management-tools.html.
6. Популярні методології управління проєктами: від гнучкої Agile до вимогливої PRINCE2 — Wizeclub Education [Електронний ресурс] // Wizeclub Education. – Режим доступу: https://wizeclub.education/blog/populyarni-metodologiyi-upravlinnya-proyektami-vid-gnuchkoyi-agile-do-vimoglivoyi-prince2/?gclid=CjwKCAiA1fqrBhA1EiwAMU5m_8mnroIyJn6Qjlv4kpOLxnvMzrRyqi0UJ7tfp1QkvTtXdqoDUI7zGRoCKoAQAvD_BwE .
7. Методологія управління проєктами: корисний посібник | nt.ua [Електронний ресурс] // IT-курси, бізнес-тренінги, сертифікація | nt.ua. – Режим доступу: https://nt.ua/blog/methodology-of-project-management.
8. IBM Documentation [Електронний ресурс] // IBM in Deutschland, Österreich und der Schweiz | IBM. – Режим доступу: https://www.ibm.com/docs/en/rsm/7.5.0?topic=structure-class-diagrams.

9. UML — Class Diagram [Електронний ресурс] // Online Tutorials, Courses, and eBooks Library | Tutorialspoint. – Режим доступу: https://www.tutorialspoint.com/uml/uml_class_diagram.htm. –
10. What is sequence diagram? [Електронний ресурс] // Ideal Modeling & Diagramming Tool for Agile Team Collaboration. – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/.
11. JavaScript with syntax for types. [Електронний ресурс] // TypeScript: JavaScript With Syntax For Types. – Режим доступу: https://www.typescriptlang.org/.
12. Next.js tutorial with examples: Build better React apps with Next [Електронний ресурс] // Educative. – Режим доступу: https://www.educative.io/blog/nextjs-tutorial-examples.
13. What Is PostgreSQL? [Електронний ресурс] // Kinsta. – Режим доступу: https://kinsta.com/knowledgebase/what-is-postgresql/.
14. What is web application architecture? Best practices, tutorials [Електронний ресурс] // Stackify. – Режим доступу: https://stackify.com/web-application-architecture/.
15. How web works — web application architecture for beginners — geeksforgeeks [Електронний ресурс] // GeeksforGeeks. – Режим доступу: https://www.geeksforgeeks.org/how-web-works-web-application-architecture-for-beginners/.
16. Fadali Y. How to create a full-stack application with next.js – A step-by-step tutorial for beginners [Електронний ресурс] / Yazdun Fadali // freeCodeCamp.org. – Режим доступу: https://www.freecodecamp.org/news/build-a-full-stack-application-with-nextjs/.
17. Bongers C. Next 13 — server and client components [Електронний ресурс] / Chris Bongers // DEV Community. – Режим

доступу: https://dev.to/dailydevtips1/next-13-server-and-client-components-5b2f.

18. Chatgpt integration an overview of its applications in business and industry [Електронний ресурс] // Digittrix Web and App development company in Chandigarh. – Режим доступу: https://www.digittrix.com/blogs/chatgpt-integration-an-overview-of-its-applications-in-business-and-industry.
19. Verification and validation in software testing [Електронний ресурс] // BrowserStack. – Режим доступу: https://www.browserstack.com/guide/verification-and-validation-in-testing.
20. Component testing: techniques and practical examples [Електронний ресурс] // bluezorro.com. – Режим доступу: https://bluezorro.com/blog/what-is-component-testing/.
21. Тестування програм та систем [Електронний ресурс] // Pidru4niki. – Режим доступу: https://pidru4niki.com/1628011847733/informatika/testuvannya_program_sistem.
22. Головна — Репозитарій Вінницького Національного Технічного Університету. – Режим доступу: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/20639/5110.pdf?sequence=3.
23. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТОК А

Міністерство освіти та науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
проф. д.т.н.,
_____ Азаров О.Д.
"_26_" ____вересня____ 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи
«Веб-додаток для створення, організації та відстеження індивідуальних й командних задач і проектів»
08-54.МКР.009.00.000 ПЗ

Науковий керівник д.т.н., проф. каф.ОТ
_____ Азаров О.Д.

Студент групи 1КІ–22м
_____ Льопа Б.В.

Вінниця 2023

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Сучасні технології та швидкий темп розвитку інформаційного суспільства вимагають ефективних інструментів для організації та відстеження індивідуальних і командних задач і проектів. Розробка такого веб-додатка є важливою складовою для покращення робочого процесу та досягнення високої продуктивності. Використання сучасних та надійних технології, забезпечують швидкість, стабільність, автоматизованість, безпеку та масштабованість розробленого продукту;

1.2Наказ про затвердження теми МКР.

2 Мета МКР і призначення розробки

2.1 Мета магістерської роботи полягає у створенні та вдосконаленні веб-додатка на базі Next.js для ефективного управління індивідуальними та командними задачами і проектами. Використання PostgreSQL, Supabase та Prisma гарантує надійність та швидкість бази даних, а React Query та Tailwind забезпечують оптимальну продуктивність та дизайн інтерфейсу. Інтеграція ChatGPT допомагатиме автоматизувати створення описів та рекомендацій для завдань, покращуючи робочий процес користувачів. Результатом роботи буде інноваційний та функціональний інструмент, спрямований на поліпшення організації та виконання завдань в робочому середовищі;

2.2 Призначення даної магістерської роботи полягає в розробці та вдосконаленні веб-додатка, який слугуватиме ефективним інструментом для керування індивідуальними та командними задачами і проектами. Цей додаток спрямований на забезпечення користувачів зручним та інтуїтивно зрозумілим інтерфейсом для створення, організації та відстеження їх завдань.

Розроблений веб-додаток має вирішити проблеми, пов'язані з неефективністю та розпорошеністю завдань, що може виникати в робочих групах та проектах. Призначення роботи також включає розширення можливостей додатка за допомогою інтеграції ChatGPT, що дозволить

автоматизувати певні аспекти комунікації та документації через генерацію тексту.

Загалом, мета цієї роботи — створити інноваційний інструмент, який полегшить керування завданнями, підвищить продуктивність та сприятиме кращій організації робочих процесів для користувачів.

3 Вихідні дані для виконання МКР

3.1 Проведення аналізу сучасного стану теорії і практики;

3.2 Представлення статичної структури моделі системиб вибір технологій для розробки;

3.3 Проектування та розробка веб-додатку;

3.4 Виконання розрахунків для доведення доцільності нової розробки з економічної точки зору;

4 Вимоги до виконання МКР

— розробити веб-додаток на базі Next.js для керування індивідуальними та командними задачами та використовувати сучасні технології;

— впровадження системи реєстрації та авторизації користувачів з можливістю відновлення паролю;

— забезпечити функціонал для створення, редагування, переміщення та видалення завдань і проектів;

— відстеження стану завдань з використанням різних статусів (в розробці, відкладено, завершено);

— інтеграція ChatGPT для автоматизованого генерування текстового опису завдань та рекомендацій;

— інтерфейс користувача повинен бути інтуїтивно зрозумілим та адаптивний для різних типів пристроїв та роздільних здатностей екрану.

— провести тестування для перевірки правильності роботи додатка та виявлення можливих помилок.

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналітичний огляд сучасного стану управління завданнями та проектами			Розділ 1
2	Теоретичні дослідження			Розділ 2
3	Розробка програмного засобу			Розділ 3
4	Тестування експериментальних досліджень			Розділ 4
5	Підготовка економічної частини			Розділ 5
6	Оформлення пояснювальної записки, графічного матеріалу і презентації			ПЗ, графічний матеріал і презентація
7	Підготовка і підпис супроводжуючих документів, нормоконтроль та тест на плагіат			Оформлені документи

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлюванні МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104-2006 «Единa система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп'ютерна інженерія»;

— документи на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21».

ДОДАТОК Б

Діаграма класів

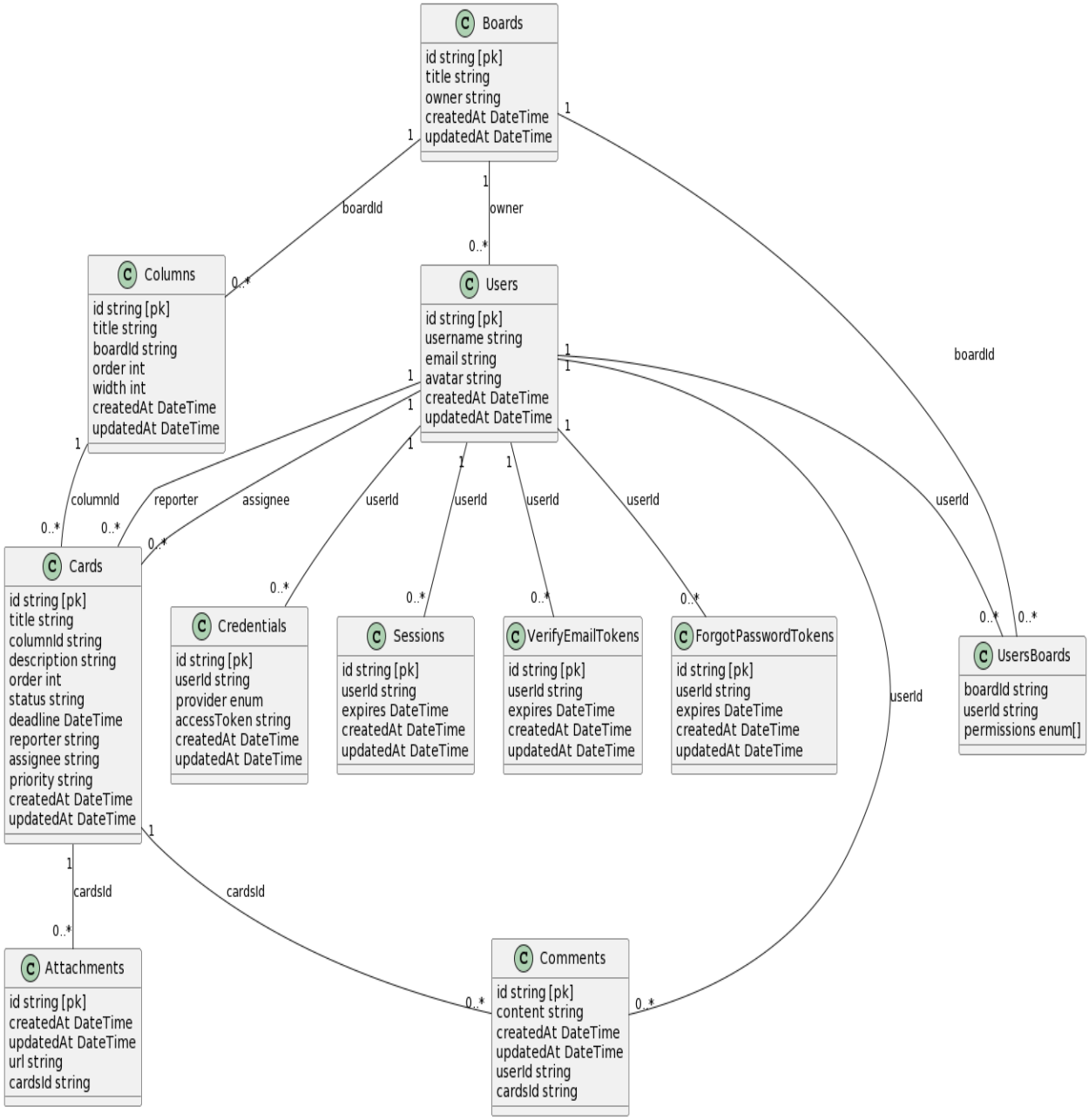

Рисунок Б.1 — Діаграма класів

ДОДАТОК В

Діаграма послідовностей

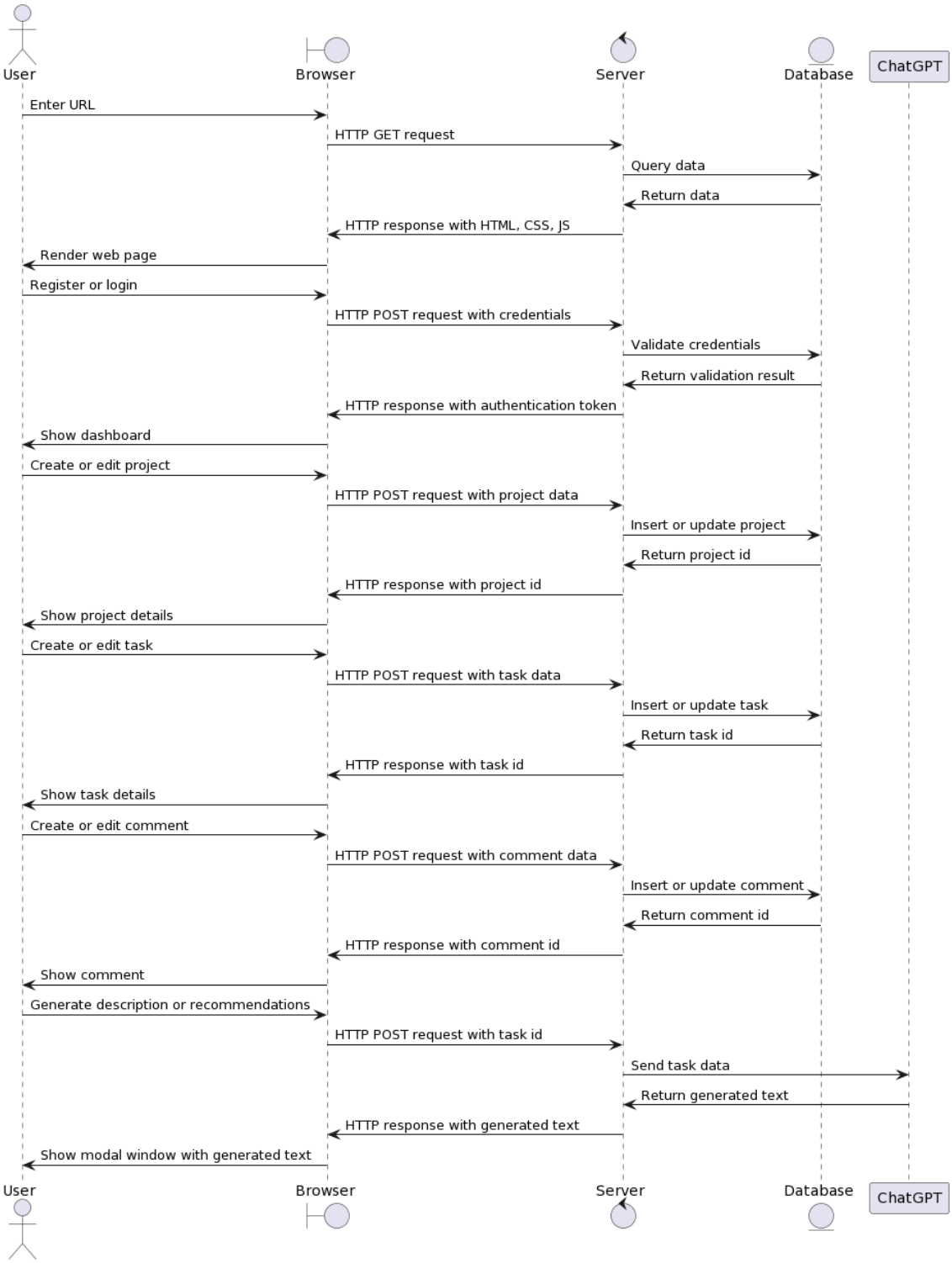

Рисунок В.1 — Діаграма послідовностей

ДОДАТОК Г

Лістинг файлу schema.prisma.ts

```
datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}
generator client {
  provider = "prisma-client-js"
}
model Board {
  id        String   @id @default(uuid())
  title     String
  owner     User     @relation(name: "BoardToOwner", fields: [ownerId], references:
[id])
  ownerId   String
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
  columns   Column[]
  users     User[]   @relation(name: "BoardToUsers")
}
model Column {
  id        String   @id @default(uuid())
  title     String
  board     Board    @relation(fields: [boardId], references: [id])
  boardId   String
  order     Int
  width     Int
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
```

```
  cards     Card[]
}
model Card {
  id        String   @id @default(uuid())
  title     String
  column    Column   @relation(fields: [columnId], references: [id])
  columnId  String
  description String?
  order     Int
  status    String
  deadline  DateTime?
  reporter  User    @relation(name: "CardToReporter", fields: [reporterId],
references: [id])
  reporterId String
  assignee  User?   @relation(name: "CardToAssignee", fields: [assigneeId],
references: [id])
  assigneeId String?
  priority  String
  createdAt  DateTime @default(now())
  updatedAt  DateTime @updatedAt
}
model User {
  id             String          @id @default(uuid())
  username       String
  email          String          @unique
  avatar         String?
  createdAt      DateTime        @default(now())
  updatedAt      DateTime        @updatedAt
  boards         Board[]         @relation(name: "BoardToUsers")
  ownedBoards    Board[]         @relation(name: "BoardToOwner")
```

```
  credentials        Credential[]
  sessions           Session[]
  verifyEmailTokens  VerifyEmailToken[]
  forgotPasswordTokens ForgotPasswordToken[]
  reportedCards      Card[]            @relation(name: "CardToReporter")
  assignedCards      Card[]            @relation(name: "CardToAssignee")
}
model Credential {
  id          String   @id @default(uuid())
  user        User     @relation(fields: [userId], references: [id])
  userId      String
  provider    String
  accessToken String
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt
}
model Session {
  id        String   @id @default(uuid())
  user      User     @relation(fields: [userId], references: [id])
  userId    String
  expires   DateTime
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
}
model VerifyEmailToken {
  id        String   @id @default(uuid())
  user      User     @relation(fields: [userId], references: [id])
  userId    String
  expires   DateTime
  createdAt DateTime @default(now())
```

```
  updatedAt DateTime @updatedAt
}
model ForgotPasswordToken {
  id        String   @id @default(uuid())
  user      User     @relation(fields: [userId], references: [id])
  userId    String
  expires   DateTime
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
}
```

ДОДАТОК Д

Лістинг файлу SignUpForm.tsx

```
"use client";
import { zodResolver } from "@hookform/resolvers/zod";
import Link from "next/link";
import { useForm } from "react-hook-form";
import { z } from "zod";
import { Button, Input } from ".";
import { useSignUpMutation } from "@/hooks/use-sign-up-mutation";
import { AxiosError } from "axios";
import { useRouter } from "next/navigation";
const signUpFormSchema = z
  .object({
    username: z.string().trim().min(3),
    email: z.string().trim().email(),
    password: z.string().trim().min(6),
    passwordConfirmation: z.string().trim().min(6),
  })
  .refine((data) => data.password === data.passwordConfirmation, {
    message: "Passwords do not match",
    path: ["passwordConfirmation"],
  });
type SignUpFormValues = z.infer<typeof signUpFormSchema>;
export function SignUpForm() {
  const { mutateAsync } = useSignUpMutation();
  const router = useRouter();
  const {
    register,
    handleSubmit,
```

```
  formState: { errors, isSubmitting, isValid, isDirty },
  setError,
} = useForm<SignUpFormValues>({
  resolver: zodResolver(signUpFormSchema),
});
const onSubmit = handleSubmit(async (values) => {
  try {
    await mutateAsync({
      username: values.username,
      email: values.email,
      password: values.password,
    });
    router.replace(`/auth/sign-up/success?email=${values.email}`);
  } catch (e) {
    if (e instanceof AxiosError) {
      if (e.response?.data.code === "user_already_exists") {
        setError("root", {
          type: "manual",
          message: "This username or email is already taken.",
        });
      }
    } else {
      setError("root", {
        type: "manual",
        message: "Ooops, something went wrong. Please try again later.",
      });
    }
  }
});
return (
```

```
    <form onSubmit={onSubmit} className="block-wrapper">
      <h1 className="text-3xl text-white font-bold text-center">Register</h1>
      <p className="text-white text-center">
        Already have an account?
        <Link
          href="/auth/sign-in"
          className="underline decoration-1 underline-offset-2 hover:decoration-
dashed"
        >
          Sign in
        </Link>
      </p>
      {errors.root?.message && (
        <p className="text-center text-red-500 font-medium">
          {errors.root.message}
        </p>
      )}
      <Input
        {...register("username")}
        placeholder="Enter your username"
        error={errors.username?.message}
      />
      <Input
        {...register("email")}
        type="email"
        placeholder="Enter your email"
        error={errors.email?.message}
      />
      <Input
        {...register("password")}
```

```
        type="password"
        placeholder="Password"
        error={errors.password?.message}
      />
      <Input
        {...register("passwordConfirmation")}
        type="password"
        placeholder="Confirm password"
        error={errors.passwordConfirmation?.message}
      />
      <Button
        isLoading={isSubmitting}
        disabled={!isValid || !isDirty}
        type="submit"
      >
        Create accout
      </Button>
    </form>
  );
}
```

ДОДАТОК Е

Вигляд інтерфейсу веб-додатку

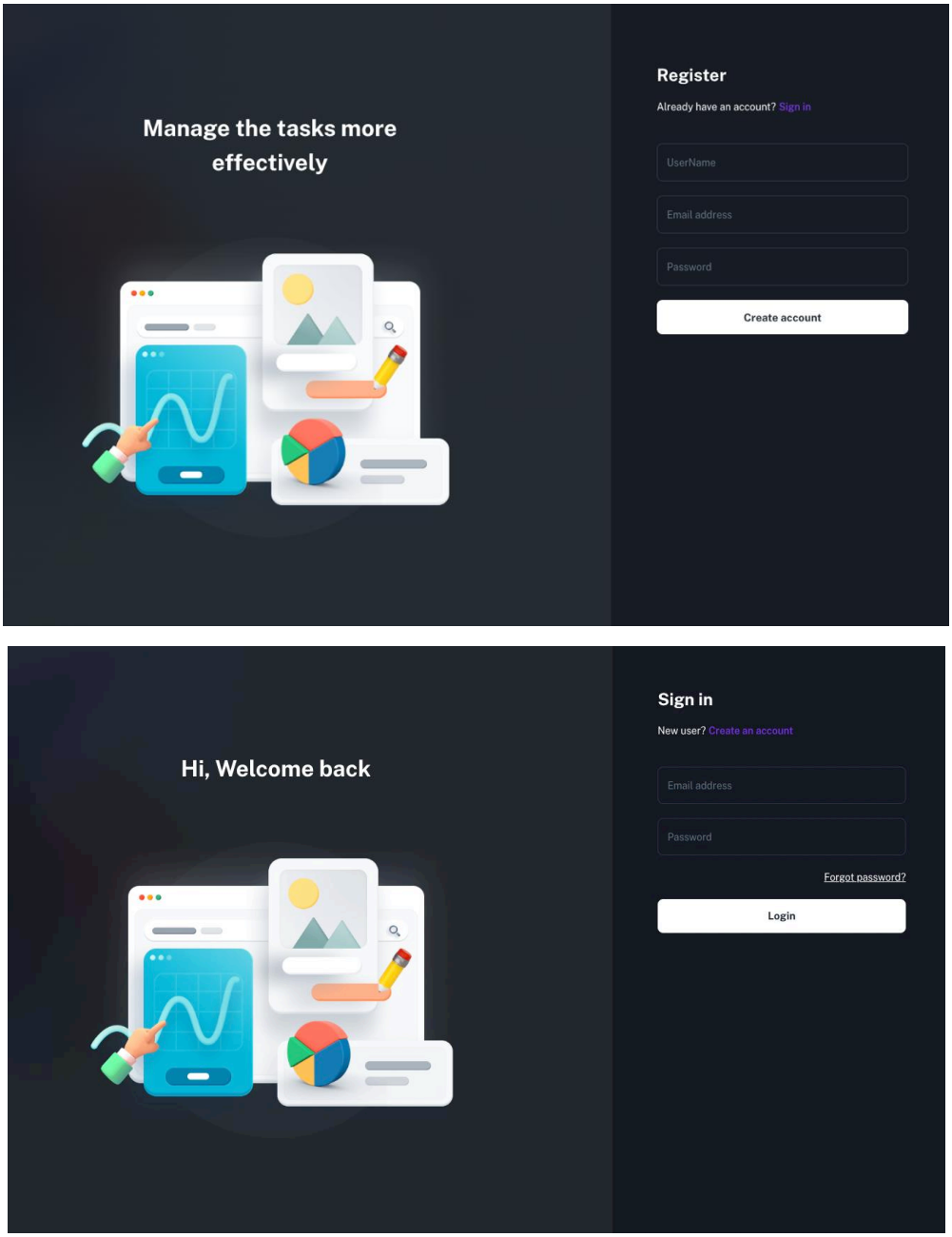

Рисунок Е.1 — Вигляд інтерфейсу веб-додатку

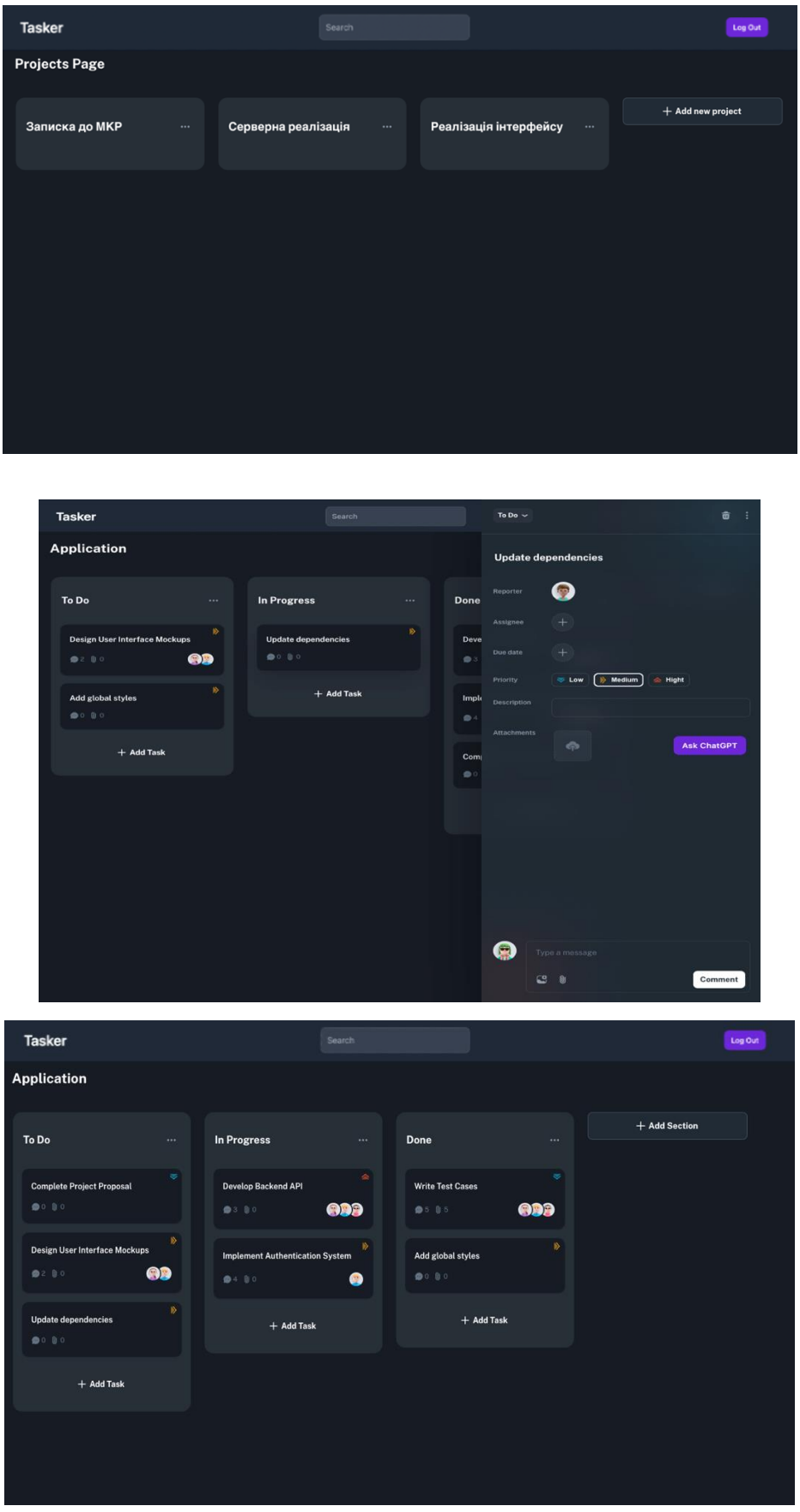

Рисунок Е.1 — Вигляд інтерфейсу веб-додатку

ДОДАТОК Ж

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Веб-додаток для створення, організації та відстеження індивідуальних й камандних задач і проектів

Тип роботи: магістерська кваліфікаційна робота.
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 85,6% Схожість 14,4%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Льопа Б.В.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Азаров О. Д.
(підпис) (прізвище, ініціали)