Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка методів і програмних засобів управління проєктами та задачами в командному середовищі»

Виконав: студент II курсу
групи 1ПІ-22м спеціальності
121 – Інженерія програмного забезпечення
К.Ю.О. Корольчук Ю.О.

Керівник: к.т.н., доц. каф. ПЗ Ракитянська Г.Б.
«11» грудня 2023 р.

Опонент: к.т.н., доц. каф. ЗІ Баришев Ю.В.
«11» грудня 2023 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)
«11» грудня 2023 р.

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор Романюк О.Н.
« 19 » вересня 2023 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Корольчуку Юрію Олеговичу

1. Тема роботи – розробка методів і програмних засобів управління проєктами та задачами в командному середовищі.

Керівник роботи: Ракитянська Ганна Борисівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від « 18 » вересня 2023 р. № 247.

2. Строк подання студентом роботи
5 грудня 2022 р.

3. Вихідні дані до роботи: аналіз сучасних методів та програмних засобів управління проєктами та задачами в командному середовищі, аналіз технологій створення веб-застосунків з інтерактивним інтерфейсом, базові поняття структури веб-застосунку, базові поняття структури сторінок веб-застосунку.

4. Зміст текстової частини: вступ; аналіз методів і програмних засобів управління проєктами та задачами в командному середовищі; моделювання інформаційної системи управління проєктами та задачами в командному середовищі; реалізація програмного засобу управління проєктами та задачами в командному середовищі; тестування роботи програмного засобу; економічна частина; висновки; додатки.

5. Перелік ілюстративного матеріалу: назва роботи, актуальність дослідження, об'єкт, предмет та мета дослідження, задачі дослідження, етапи розробки, порівняння аналогів, метод управління проєктами та задачами в командному середовищі на основі аналізу даних про пріоритет задач, алгоритм визначення пріоритетності задачі та часу на її виконання за допомогою машинного навчання, блок-схема алгоритму визначення пріоритетності задачі та часу на її

виконання за допомогою машинного навчання, основні алгоритми засто
використані технології, авторизація у системі, головна сторінка, сторінка пр
створення нової задачі, тестування, економічна частина, наукова но
висновки, публікації.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийня
1-4	Ракитянська Г.Б., к.т.н., доцент кафедри ПЗ	19.09.2023	05.12.202
5	Причепа І.В., к.е.н., доцент кафедри ЕПВМ	20.11.2023	01.12.202

7. Дата видачі завдання 19 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи	Приміт
1	Аналіз методів і програмних засобів управління проєктами та задачами в командному середовищі	20.09.2023 – 28.09.2023	Вик.
2	Розробка методу та моделі системи управління проєктами та задачами в командному середовищі	29.09.2023 – 25.10.2023	Вик.
3	Реалізація програмного засобу управління проєктами та задачами в командному середовищі	26.10.2023 – 10.11.2023	Вик.
4	Тестування роботи програмного засобу	11.11.2023 – 19.11.2023	Вик.
5.	Економічна частина	20.11.2023 – 01.12.2023	Вик.

Студент К.Ю.О. **Корольчук Ю.О.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи **Ракитянська Г.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 519.7:004.89

Корольчук Ю. О. Розробка методів і програмних засобів управління проєктами та задачами в командному середовищі. Магістерська кваліфікаційна робота зі спеціальності 121 – Інженерія програмного забезпечення. Вінниця: ВНТУ, 2023: 102 с.

На укр. мові. Бібліогр.: 37 назв; рис.: 44; табл.: 7.

Магістерська кваліфікаційна робота присвячена розробці методів та засобів управління проєктами та задачами в командному середовищі, особливістю якого є використання технологій штучного інтелекту для покращення рішень у проєкті та передбачення появи можливих ризиків.

В процесі виконання роботи проведено аналіз предметної області, здійснено аналіз методів управління проєктами та порівняльний аналіз аналогів програмних засобів. Враховуючи всі недоліки та переваги проаналізованих аналогів, визначено основні задачі розробки, розроблено діаграми варіантів використання системи, спроектовано логічну та фізичну структури застосунку, розроблено метод управління проєктами та задачами.

Проведено аналіз відомих засобів розробки веб-застосунків. Розроблено модуль управління задачами. Виконано розробку веб-застосунку.

Результати тестування застосунку та модулів системи виявились успішними.

Для розробки застосунку було обрано JavaScript-бібліотеку для розробки користувацьких інтерфейсів React, та BaaS-сервіс Firebase, у якості середовища розробки використано Visual Studio Code.

Ключові слова: проєкт, задача, управління, метод, ітерація, цілі, ресурси.

ANNOTATION

Korolchuk Y. O. Development of methods and software tools for managing projects and tasks in a team environment. Master's qualification work on the specialty 121 - Software engineering. Vinnytsia: VNTU, 2023: 102 p.

Written in Ukrainian language. Bibliography: 37 titles; pictures: 44; tables: 7.

The master's qualification work is devoted to the development of methods and means of managing projects and tasks in a team environment, the feature of which is the use of artificial intelligence technologies to improve decisions in the project and predict the appearance of possible risks.

In the course of the work, an analysis of the subject area was carried out, an analysis of project management methods and a comparative analysis of software analogues were carried out. Taking into account all the shortcomings and advantages of the analyzed analogues, the main tasks of the development were defined, diagrams of options for the use of the system were developed, the logical and physical structure of the application was designed, and a method of managing projects and tasks was developed.

An analysis of known means of web application development was carried out. A task management module has been developed. Web application development completed.

The results of testing the application and system modules were successful.

The JavaScript library for the development of user interfaces React and the BaaS service Firebase were chosen for the development of the application, Visual Studio Code was used as the development environment.

Keywords: project, task, management, method, iteration, goals, resources.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Вибір теми дослідження обґрунтовується актуальністю та розвитком інформаційних технологій у сучасному світі. Командна робота та управління проєктами стали важливими у багатьох сферах бізнесу, освіти та державного управління. Відмінність командної роботи полягає в необхідності спільних зусиль для досягнення цілей, що ставить нові вимоги до систем управління проєктами та задачами.

Потреба в розробці нових методів та програмних засобів управління виникає через зростання складності проєктів, над якими працюють команди. Сучасні проєкти включають в себе великий обсяг інформації, різноманітні завдання та швидкозмінюючі вимоги з боку замовників. Тому необхідно розробляти методи та інструменти, які дозволять ефективно керувати ресурсами, забезпечуючи гнучкість та швидкість у вирішенні завдань.

Крім того, розвиток технологій аналізу даних та машинного навчання відкриває нові можливості для створення систем управління проєктами. Використання цих технологій дозволяє автоматизувати процеси прийняття рішень, передбачати можливі ризики та вчасно реагувати на зміни в проєкті.

Дослідження у цій області має великий потенціал для розвитку не лише в бізнесі, але й у наукових дослідженнях. Впровадження інноваційних методів та програмних засобів у сфері управління проєктами може сприяти оптимізації ресурсів, зменшенню витрат та підвищенню якості продукту чи послуги.

Крім того, розробка нових методів та програмних засобів управління проєктами може сприяти розв'язанню соціальних проблем, таких як підвищення ефективності державних проєктів, сприяючи економічному розвитку і покращенню якості життя громадян.

Таким чином, обгрунтування вибору даної теми дослідження базується на її широкому спектрі можливостей для впровадження та позитивному впливі на різні сфери суспільства. Саме тому розробка і впровадження системи

управління проєктами та задачами в командному середовищі є актуальною задачею.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є підвищення ефективності процесу управління проєктами та задачами в командному середовищі за рахунок запропонування нового методу, що передбачає використання штучного інтелекту в даних процесах.

Основними задачами дослідження є:

— провести порівняльний аналіз існуючих програмних засобів управління проєктами та задачами в командному середовищі, виділити переваги та недоліки для формування функціональних вимог розроблюваного програмного застосунку;

— провести аналіз існуючих методів вирішення задачі дослідження;

— запропонувати метод управління проєктами та задачами в командному середовищі на основні аналізу даних про пріоритет задач і час на її виконання;

— розробити модель системи та алгоритми роботи системи управління проєктами із моделлю машинного навчання;

— розробити веб-систему на основі запропонованого методу;

— провести тестування розробленого застосунку.

Об'єктом дослідження є процес розробки системи управління проєктами та задачами в командному середовищі.

Предметом дослідження є методи та засоби розробки системи управління проєктами та задачами в командному середовищі.

Методи дослідження. У процесі досліджень використовувались: методи проектування архітектури програмного забезпечення для створення архітектури системи, методи синтезу для розробки моделей системи, методи

інтелектуального аналізу даних для обробки даних задач проєкту, методи алгоритмізації процесів для розробки алгоритмів роботи модулів системи, методи тестування для перевірки роботи створеного застосунку.

Наукова новизна отриманих результатів.

— Отримав подальшого розвитку метод управління проєктами та задачами в командному середовищі, відмінністю якого є аналіз часових витрат залежно від пріоритету задач, що дозволяє прогнозувати ризики та час виконання ітерацій або проєкта в цілому.

— Отримала подальший розвиток модель веб-системи, яка на відміну від відомих моделей управління проєктами використовує модель машинного навчання, що дозволяє підвищити ефективність процесу прийняття рішень та зменшити ризики відхилення від графіку проєкта.

Практичне значення одержаних результатів. Практичне значення магістерської кваліфіаційної роботи полягає у практичному застосуванні розроблених моделей, методів та веб-системи для управління проєктами та задачами в різних сферах діяльності.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто.

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення магістерської кваліфікаційної роботи доповідалися на Міжнародній науково-практичній Інтернет-конференції 2023 року «Електронні інформаційні ресурси: створення, використання, доступ».

Публікації. Основні положення магістерської кваліфікаційної роботи опубліковані у збірнику матеріалів Міжнародної науково-практичної Інтернет-конференції 2023 року «Електронні інформаційні ресурси: створення, використання, доступ» [1].

Структура та обсяг роботи. Магістерська кваліфікаційна робота складається зі вступу, п’яти розділів, висновків, списку літератури, що містить 37 найменувань, 4 додатки. Робота містить 44 ілюстрації, 7 таблиць.

1. АНАЛІЗ МЕТОДІВ І ПРОГРАМНИХ ЗАСОБІВ УПРАВЛІННЯ ПРОЄКТАМИ ТА ЗАДАЧАМИ В КОМАНДНОМУ СЕРЕДОВИЩІ

1.1. Аналіз стану питання управління проєктами та задачами в командному середовищі

Проєкт — це набір конкретних дій, які відбуваються протягом обмеженого періоду часу та мають на меті вирішення певної проблеми або досягнення конкретної мети. Проєкти включають багато зацікавлених сторін, вимагають ретельного планування, передбачають координацію між різними командами та мають часовий графік, який може тривати кілька місяців [2]. Задача — це окрема частина проєкту. Задача, як правило, менші за обсягом, ніж загальний проєкт, і можуть залучати лише одну людину чи команду. Задачі можуть залежати одна від одної і залежати від попередньої задачі. Для виконання задач може знадобитися як одна так і декілька людей, тоді як для проєкту майже завжди потрібна команда.

Управління проєктом – це процес, який має фіксовану дату початку та завершення, через різні етапи планування, виконання, моніторингу та закриття проєкту [2, 3]. Проєкт спрямований на виконання певної кінцевої мети шляхом виконання кількох задач для її досягнення.

Управління проєктами вимагає чіткої постановки задач, яка полягає у визначенні мети, обсягу та ресурсів, необхідних для досягнення цієї мети. Коректно сформульовані задачі стають основою для планування, виконання, відстеження та оцінки результатів проєктів. Вони визначають, що саме потрібно зробити, коли це повинно бути зроблено, і які ресурси необхідні для цього.

Управління задачами — це процес моніторингу завдань вашого проєкту на різних етапах від початку до кінця [4]. Цей процес включає в себе активне прийняття рішень щодо ваших завдань з урахуванням змін, які можуть

відбутися в реальному часі, з вашою кінцевою метою — успішне виконання ваших завдань. Управління задачами проєкту також означає ефективне управління всіма аспектами задачі, такими як бюджет, час, обсяг, ресурси, повторення тощо.

Керування проєктами вимагає бачення загальної картини. Успішне управління проєктом вимагає належного аналізу взаємозв'язків і залежностей, які існують між його різними компонентами. Управління задачами, з іншого боку, передбачає мікрофокус на досягненні лише виконання завдання. Управління як проєктом, так і задачею потребує участі та управління командою, щоб переконатися, що ресурси використовуються якнайбільш оптимально.

Управління задачами допомагає покращити наступні аспекти командної роботи [4]:

— продуктивність: гарантуючи, що потрібні ресурси використовуються в потрібний час. Це включає надання членам команди задач і обов’язків, які відповідають їхнім здібностям та інтересам.

— організованість: організація задач таким чином, щоб спростити робочий процес, що значною мірою допоможе підвищити ефективність і досягти мети;

— фокусування: систематичний спосіб підходу до всіх задач, що домагає бути більше зосереджена на кінцевій меті.

У опитуванні, проведеному Інститутом Менеджменту Проєктів у 2017 році [5], зазначається, що 37% виконавчих керівників сказали, що «основною причиною невдачі проєктів у їхній організації була відсутність чітко визначених цілей і етапів для вимірювання прогресу» та «відсутність дисципліни під час реалізації стратегії». З огляду на це дослідження, ось кілька основних кроків, пов’язаних із управління задачами:

— пріоритезація. Незалежно від того, який інструмент використовується для керування задачами, будь то простий список справ чи

комплексний інструмент керування завданнями проєкту, найважливішим аспектом керування задачами є встановлення пріоритетів. Це допомагає ефективно виконувати всі поставлені задачі, дотримуючись запланованих обмежень;

— відстеження цілей. Результати цього дослідження також показують, що встановлення чітких цілей має важливе значення для успішного завершення проєкту. Хоча для управління задачами кінцева мета не є обов'язковою, визначення етапів допомагає мотивувати команди успішно виконувати задачі;

— управління розкладом. Потрібно переконатись, що задача буде виконана вчасно, це саме те, що впливає на загальний стан проєкту. Ключовим тут є встановлення правильного часового проміжку шляхом оцінки залучених ресурсів;

— розподіл ресурсів. Оптимальне управління ресурсами є наступним кроком для забезпечення ефективного управління задачами. Залежно від бюджету та графіка потрібно призначити правильну особу і кількість ресурсів для виконання роботи.

1.2. Порівняльний аналіз існуючих програмних засобів управління проєктами та задачами

Один із найрозповсюдженіших методів у проведенні наукових досліджень - це метод порівняння. У випадку, коли об'єкти мають певну схожість чи спільні характеристики, можна використовувати цей метод. Під час розгляду кожного об'єкта, який підлягає порівнянню, можна визначити конкретні критерії та ознаки, за якими їх можна порівняти.

Для ефективної розробки програмного продукту важливо провести докладний порівняльний аналіз існуючих аналогів на ринку. Це дозволить не лише оцінити актуальність розробки, але й виділити переваги та недоліки існуючих програмних рішень. Цей аналіз допоможе впровадити найбільш

цінні аспекти інших продуктів у нашу розробку, забезпечуючи таким чином створення продукту з меншою кількістю недоліків вже на етапі проектування.

Розглянемо деякі з існуючих рішень:

1) Microsoft Project [6].

Mircosoft Project – система управління проєктами, розроблена компанією Microsoft. Система доступна для завантаження як для настільного ПК та і на мобільні пристрої, також користувач має доступ до веб версії.

Перевагами Microsoft Project є велика кількість функцій для докладного планування та відстеження проєктів, інтеграція з іншими продуктами Microsoft, що полегшує співпрацю та можливість створювати складні проєкти з різними завданнями та ресурсами. Недоліки: висока вартість ліцензії, що робить його менш доступним для невеликих компаній та індивідуальних користувачів та схильність до складності, що може затягнути процес впровадження.

Інтерфейс інструменту зображено на рисунку 1.1:

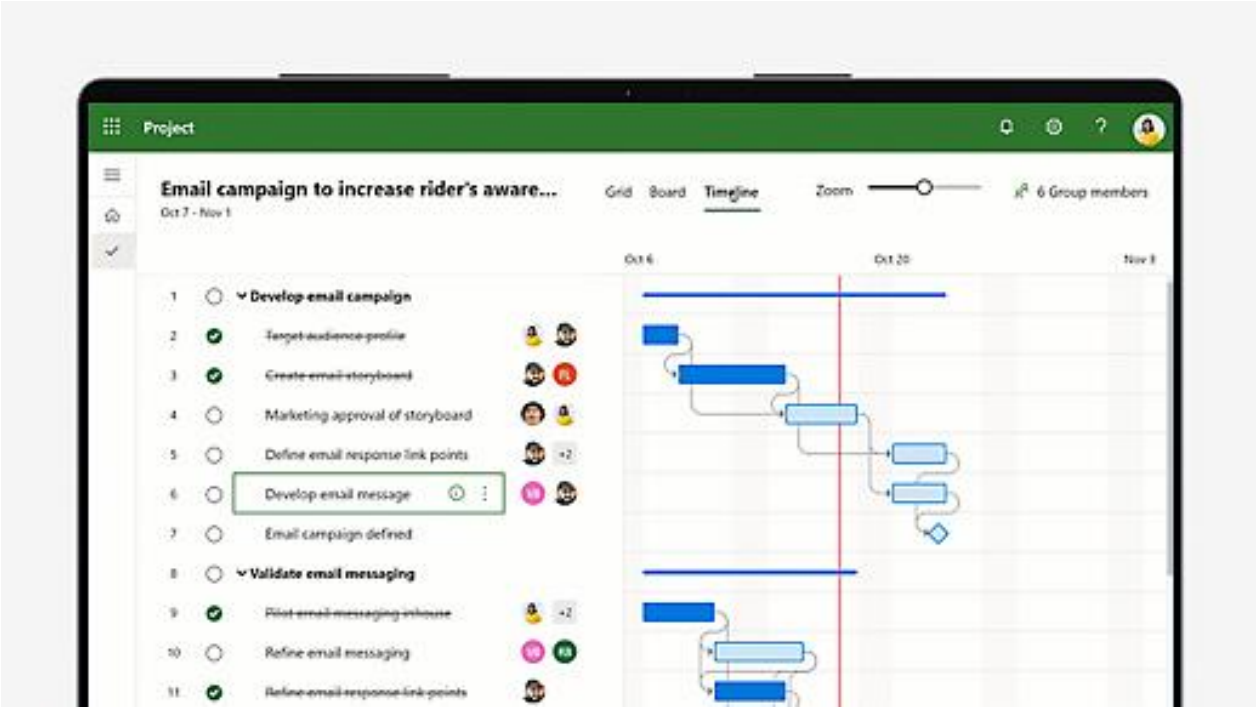

Рисунок 1.1 – Інтерфейс Microsoft Project

2) Trello [7].

Trello – інструмент для керування проєктами та процесами всередині команди. Інструмент має можливість додавати файли, контрольні списки та правила автоматизації. Інструмент доступний у веб та на мобільних пристроях.

Перевагами Trello є простий та інтуїтивно зрозумілий інтерфейс, що полегшує швидке створення та організацію завдань, гнучкість в користуванні, можливість адаптації до різних видів проєктів та можливість створювати невеликі команди в базовій версії. Недоліками в свою чергу є обмежена можливість розширення функціоналу в безкоштовній версії та порівняно невелика глибина функціоналу у базовій версії.

Інтерфейс інструменту зображено на рисунку 1.2:

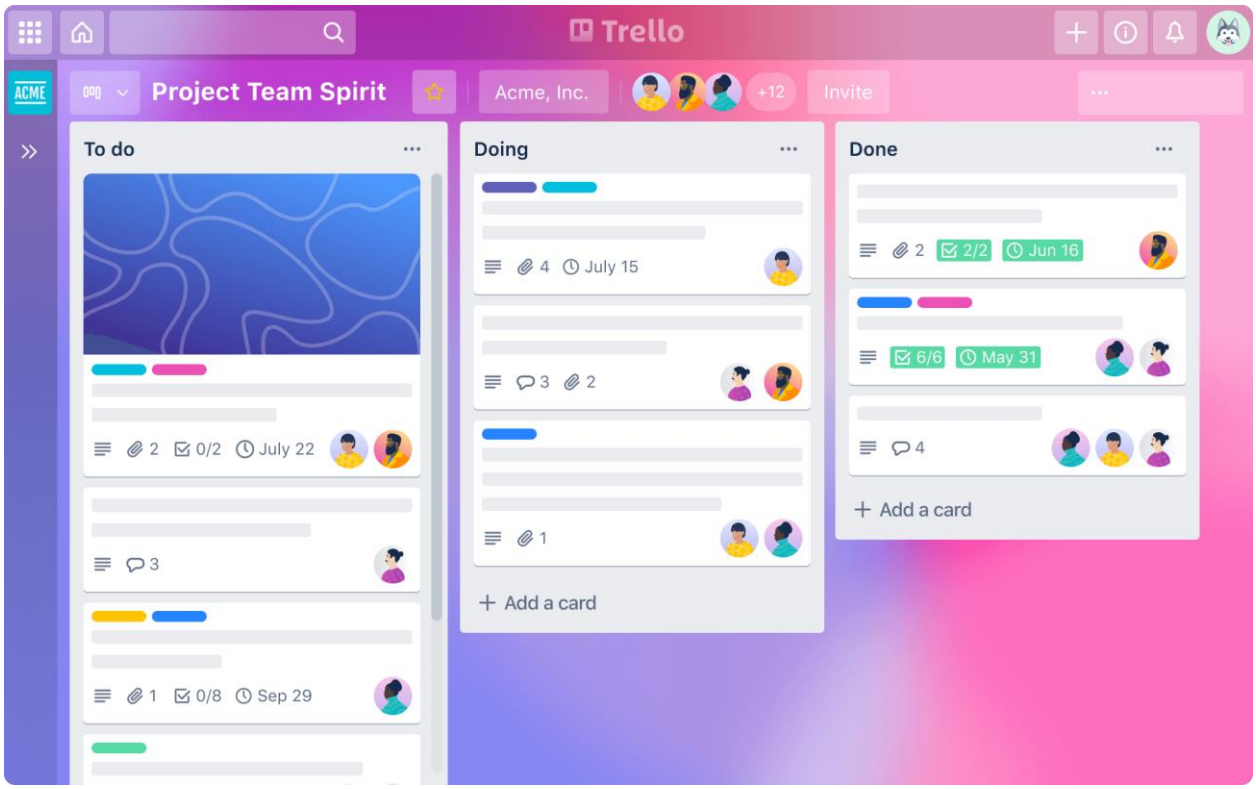

Рисунок 1.2 – Інтерфейс Trello

3) Asana [8].

Asana – програмне забезпечення для веб і мобільних пристроїв для спільної роботи над проєктами. Одною з особливостей даного інструменту є можливість працювати без електронної пошти. Інструмент доступний у веб, настільній та мобільній версіях.

Asana видається відмінним інструментом для управління проєктами через його інтуїтивний інтерфейс, який полегшує користувачам швидку та легку навігацію. Ця платформа також забезпечує зручну систему коментування та можливість співпраці в режимі реального часу, що покращує комунікацію та спільну роботу в команді.

Проте, варто зазначити, що Asana має свої недоліки. Деякі продуктивні функції доступні лише у платних версіях програми, що може бути обмежуючим фактом для користувачів з обмеженим бюджетом. Крім того, функціонал Asana може виявитися недостатньо розвиненим для складних та масштабних проєктів, де вимагається більше спеціалізованих інструментів.

Інтерфейс інструменту зображено на рисунку 1.3:

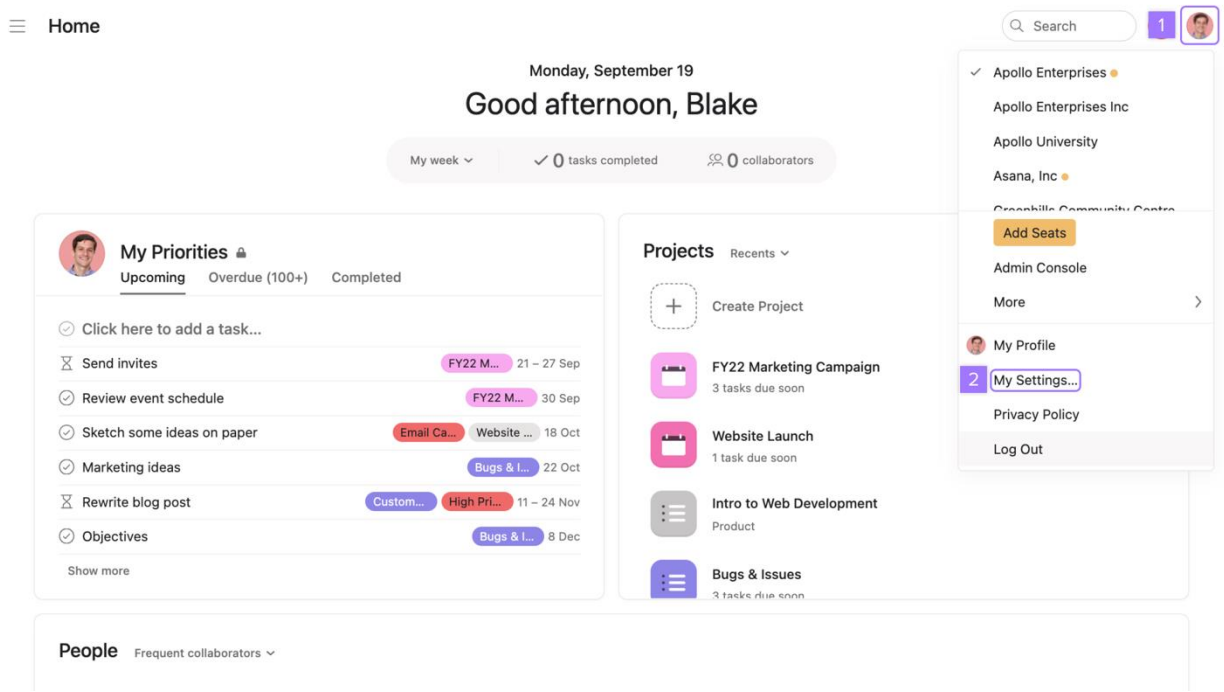

Рисунок 1.3 – Інтерфейс Asana

4) Jira [9].

Jira Core – система управління проєктами, доступна у двох версіях: хмарній і серверній. Одною з особливостей системи є її абстрагування від IT, тобто її можна використовувати у різних задачах, такий як управління HR, риск-менеджменту та управління вимогами.

Система відрізняється рядом переваг. По-перше, вона має величезний арсенал розширень і плагінів, які дозволяють розширити її базовий функціонал. Це надає користувачам можливість налаштувати робочі процеси точно під їхні потреби. Крім того, система відзначається широким спектром можливостей для налаштування процесів розробки, що робить її зручною для технічних експертів, які потребують високоіндивідуалізованих рішень.

Проте, слід зазначити деякі недоліки. По-перше, висока складність в налаштуванні та використанні може стати викликом для не-технічних користувачів, що може вплинути на їхню продуктивність. Крім того, велика кількість додаткових плагінів може призвести до додаткових витрат часу і ресурсів на їхню інтеграцію та вивчення, а також призвести до збільшення вартості використання системи.

Інтерфейс системи зображено на рисунку 1.4:

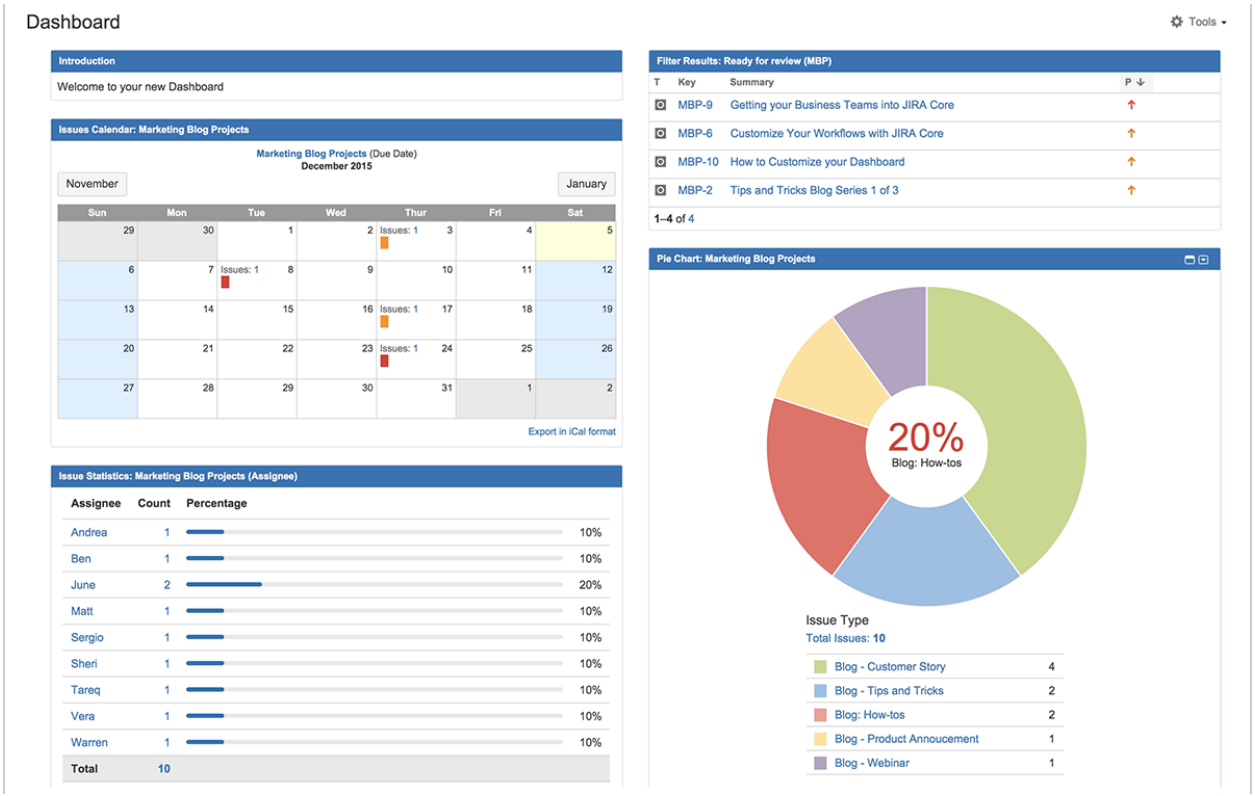

Рисунок 1.4 – Інтефейс Jira Core

Створимо порівняльну таблицю розглянутих засобів:

Таблиця 1.1 – Порівняльна характеристика програмних продуктів

Критерії	MS Project	Trello	Asana	Jira	Розроблювана система
Простий інтерфейс	-	+	+	-	+
Складний в налаштуванні	+	-	+	+	-
Необхідність інших інструментів для повного функціоналу	+	+	+	+	-
Наявність безкоштовної версії	-	+	+	-	+
Наявність засобів аналітики	+	-	-	-	+
Складний в інтеграції	+	-	+	+	-

Після докладного аналізу функціоналу та можливостей даних інструментів, можна дійти висновку, що кожен з них має свої переваги та обмеження, проте, недолік який зустрівся у більшості з них – занадто базовий функціонал у безкоштовній версії та переважно велика ціна за додаткові можливості. Також недоліком який зустрічався часто є складність користувацького інтефейсу, що також буде враховано при розробці дизайну та інтерфейсу розроблюваного програмного засобу у наступних розділах.

Таким чином, основними задачами розроблюваного програмного засобу буде забезпечення максимально обширного функціоналу для користувача без необхідності вставновлення додаткових платних плагінів, при цьому максимально зберігаючи простоту користувацького інтерфейсу та простоту налаштування використання.

Крім того, важливо замітити, що усі з розглянутих аналогів мають веб-версію застосунку, тому доцільним буде розпочати розробку функціоналу саме для веб з подальшим переносом на мобільні та настільні пристрої.

1.3. Аналіз методів управління проєктами та задачами в командному середовищі

Ефективність управління проєктами та задачами в командному середовищі визначається правильним вибором і ефективним застосуванням відповідних методів та стратегій. Ключовим компонентом успішного управління є вміння відзначати та використовувати різноманітні методи, які стимулюють співпрацю та залучають потенціал команди.

Розглядаючи управління з точки зору командної дії, важливо розрізняти основні процеси управління: планування, організацію, мотивацію, контроль та координацію. Гармонійна взаємодія цих елементів визначає успішність керівництва. Підбір принципів та методів управління повинен базуватися на взаємодії індивідів у команді та спрямовуватися на досягнення колективних цілей.

Один із фундаментальних аспектів управління проєктами в командному середовищі - це методи формування та виконання завдань. Успішний вибір метода вимагає розробки чіткої системи, яка враховує індивідуальні навички та потенціал кожного члена команди.

Метод адаптивного управлінського циклу є стратегією управління, яка акцентує увагу на постійному адаптивному вдосконаленні та корекції стратегій у процесі виконання проєктів чи завдань [10]. Цей метод сприймає

невизначеність та змінливість у процесі розробки та навчає команду швидко реагувати на зміни в умовах та вимогах проєкту.

Основні принципи адаптивного управлінського циклу:

— гнучкість: система повинна бути гнучкою та здатною відповідати на зміни в середовищі;

— ітерації: процес управління повторюється через короткі ітерації, що дозволяє команді швидко реагувати на нову інформацію;

— взаємодія та зворотній зв'язок: важливість співпраці та постійного зворотного зв'язку для постійного удосконалення;

— моніторинг та аналіз: систематичний моніторинг та аналіз результатів для прийняття обґрунтованих рішень.

Метод адаптивного управлінського циклу дозволяє команді ефективно пристосовуватися до невизначеності та змін у процесі проєкту, покращуючи результативність та забезпечуючи оптимальний вихід.

Отже, метод адаптивного управлінського циклу розглядається як високоефективний та перспективний підхід до управління проєктами та завданнями в командному середовищі. Цей метод сприяє активізації командної діяльності, поглибленню розуміння завдань та їх деталізації, що сприяє загальній ефективності управлінського процесу.

Метод самоорганізації є концепцією управління та організації, яка покладає акцент на внутрішній спроможності системи або групи до організованої саморегуляції та адаптації до змін в навколишньому середовищі без централізованого контролю [11]. Цей підхід часто застосовується в управлінні командами, організаціями та проєктами, де акцент робиться на створенні умов для того, щоб система могла ефективно адаптуватися та еволюціонувати без постійного втручання керівництва.

Основні аспекти методу самоорганізації включають:

— гнучкість та адаптивність: система або група повинна бути гнучкою та здатною адаптуватися до змін в середовищі, без необхідності централізованого керівництва;

— взаємодія та співпраця: засновано на ідеї взаємодії та співпраці між учасниками системи. Важливий аспект - обмін інформацією та взаємодопомога;

— емерджентність: пов'язаний з ідеєю, що організаційна структура та поведінка можуть еманувати (виникати) в результаті взаємодії окремих частин системи, а не бути заздалегідь визначеною;

— навчання та розвиток: система постійно навчається та розвивається через власний досвід та здатність адаптуватися до нових обставин.

Використання методу самоорганізації у контексті управління проєктами та задачами створює сприятливі умови для кожного учасника команди. Кожен член команди може активно внести свій внесок та спрямувати власні зусилля на досягнення спільної мети, враховуючи власні сильні сторони та індивідуальний стиль роботи. Цей підхід дає можливість кожному членові команди самостійно визначати свої завдання, темп роботи та взаємодію з іншими учасниками проєкту. Основною перевагою є здатність до саморегулювання, що сприяє підвищенню ефективності та досягненню успішних результатів.

1.4. Постановка задачі для розроблюваної системи управління проєктами та задачами в командному середовищі

Після аналізу питання розробки методів і програмних засобів управління проєктами і задачами в командному середовищі варто виділити наступні завдання, які потрібно зробити для розробки програмного засобу:

— змоделювати предметну область;

— розробити архітектуру програмного засобу;

— сформувати вимоги до системи;

— розробити специфікації вимог у вигляді діаграм варіантів використання за допомогою інструментального програмного забезпечення;

— розробити дизайн програмного засобу;

— проаналізувати вибір засобів реалізації;

— проаналізувати вибір середовища розробки;

— розробити програмний засіб та провести тестування створеного застосунку.

Виділимо наступні нефункціональні вимоги системи:

— система повинна бути стійкою до великої кількості користувацьких запитів або транзакцій без втрати продуктивності;

— після виникнення помилок система повинна автоматично відновлювати свою працездатність без втрати даних;

— система повинна забезпечувати механізми аутентифікації користувачів і авторизації доступу до різних функцій на основі ролей та прав доступу;

— система повинна бути сумісною з різними веб-браузерами;

— інтерфейс системи повинен бути інтуїтивно зрозумілим та забезпечувати легку навігацію навіть для не-технічних користувачів.

Сформуємо функціональні вимоги до системи:

— системи повинна надавати можливість створювати та редагувати новий проєкт;

— системи повинна надавати можливість створювати та редагувати нову задачу у проєкті;

— системи повинна надавати можливість запрошувати нових людей у проєкт за допомогою унікального коду проєкту;

— система повинна надавати можливість створювати резервне копіювання проєкту у хмарі;

1.5. Висновок

У першому розділі дослідження було проаналізовано методи і програмниі засоби управління проєктами та задачами в командному середовищі.

Проаналізовано такі існуючи аналоги як Microsoft Project, Trello, Asana та Jira та проведено їх порівняльний аналіз з розроблюваною системою. У результаті порівняльного аналізу було доведено доцільність розробки власної системи, яка, на відміну від інших, не потребуватиме інших інструментів, матиме засоби аналітики та буде простою у інтеграції в командне середовище. Крім цього, проведений аналіз допоміг визначити основні задачі розроблюваної системи.

Проведений детальний аналіз наявних методів розв'язання визначеної задачі, в результаті чого було визначено оптимальний напрямок. Обрано технологію адаптивного управлінського циклу, яка базується на використанні методу самоорганізації. Цей підхід визначено як ключовий для створення високоефективної системи управління проєктами та задачами в командному середовищі. Такий підхід дозволяє досягати оптимального рівня організації та ефективно вирішувати завдання в умовах командної діяльності.

Розробка програмного засобу передбачає ряд спланованих етапів, таких як моделювання предметної області, розробка архітектури, дизайну, програмного засобу та тестування функціоналу. Були визначені функціональна та нефункціональні вимоги системи.

2. РОЗРОБКА МЕТОДУ ТА МОДЕЛІ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ ТА ЗАДАЧАМИ В КОМАНДНОМУ СЕРЕДОВИЩІ

2.1. Розробка методу управління проєктами та задачами в командному середовищі на основі аналізу даних про пріоритет задач

З метою дотримання принципів адаптивного управлінського циклу, що базується на використанні методу самоорганізації необхідним є використання пріоритетності задач. Визначення пріорітету задачі вимагає врахування різноманітних факторів, щоб кожне завдання могло бути розглянуте в контексті його важливості та термінів виконання. Ось кілька ключових аспектів:

— важливість для проєкту або цілей: визначення, наскільки дана задача важлива для досягнення основних цілей проєкту. Чим більше завдання спрямоване на досягнення стратегічних цілей, тим вищий його пріоритет;

— терміни виконання: врахування часових рамок для завершення задачі. Задачі з коротшими термінами виконання можуть мати вищий пріоритет, особливо, якщо це важливо для наступних етапів проєкту;

— відносна складність: аналіз складності та трудомісткості завдання. Менш складні завдання можуть мати більш високий пріоритет, особливо, якщо їх можна виконати швидше та ефективніше;

— взаємозв'язок із іншими задачами: розгляд взаємозв'язків між різними завданнями. Завдання, які є передумовою для інших, можуть отримати вищий пріоритет, оскільки їх вчасне виконання впливає на подальший хід проєкту;

— ступінь важливості для проєкта: якщо задача пов'язана із задоволенням конкретних потреб чи очікувань клієнта, то це може збільшити її пріоритет, оскільки вона визначає клієнтську задоволеність;

— ефективність виконання: оцінка того, наскільки ефективно може бути виконана задача з наявними ресурсами. Завдання, які можна виконати швидше та без перевищення бюджету, можуть мати вищий пріоритет.

Метод визначення пріоритетності задачі базується на шкалі пріоритетності:

Високий Пріоритет (1-3):

— завдання, які критично важливі для досягнення стратегічних цілей;

— терміни виконання - невеликі, термінові завдання;

— складність - висока, але вирішення необхідно;

— мають великий вплив на інші завдання та проєкт в цілому.

Середній Пріоритет (4-6):

— завдання, які важливі, але не термінові;

— терміни виконання - середні, є тривалість для реалізації;

— складність – середня;

— мають вплив на інші завдання, але менший порівняно з високим пріоритетом.

Низький Пріоритет (7-10):

— завдання, які можна виконати пізніше, якщо є вільний час та ресурси;

— терміни виконання - достатньо довгі;

— складність - низька, можна вирішити без великих зусиль;

— мають мінімальний вплив на інші завдання.

Правильне визначення пріоритетності задачі та часу на її виконання є інструментом здійснення контролюючої функції у методі управління проєктами та задачами в командному середовищі. Ось кілька прикладів конкретних задач з визначеними пріоритетами та часом на їх виконаня:

Розробка прототипу:

— пріоритет: високий (3);

— час виконання: 2 тижні.

Тестування та виправлення помилок:

— пріоритет: високий (1);

— час виконання: 2 тижні.

Оптимізація дизайну інтерфейсу:

— пріоритет: середній (5);

— час виконання: 1 тиждень.

Існує декілька шляхів правильного визначення пріоритетності задачі та часу на її виконання. Один з варіантів – використання машинного навчання.

Машинне навчання може бути використане для оптимізації визначення пріоритетів та часу виконання завдань в командному середовищі. Ця технологія дозволить аналізувати історію попередніх проєктів, враховуючи різні критерії, такі як важливість завдання та терміновість. Моделі машинного навчання можуть оцінювати різноманітні фактори, такі як обсяг роботи та ресурси, для прогнозування часу виконання завдань.

Такі моделі також можуть автоматично призначати пріоритети завданням, враховуючи поточні умови проєкту. Крім того, система може оптимізувати розподіл завдань в команді, враховуючи навички та досвід кожного члена для максимізації продуктивності та мінімізації часу виконання. З часом система оновлює свої моделі для адаптації до змін у проєкті та команді, щоб забезпечити ефективні стратегії управління завданнями.

Використання машинного навчання в управлінні завданнями може не лише спростити процес прийняття рішень, але й покращити точність та ефективність стратегій управління проєктами в командному середовищі.

Основні етапи алгоритму визначення пріоритетності задачі та часу на її виконання за допомогою машинного навчання:

1. Збір даних: зібрати дані про попередні проєкти та задачі, включаючи пріоритет, час виконання, та фактичний час завершення.

2. Підготовка даних: провести попередню обробку даних, включаючи очищення від непотрібної інформації та заповнення пропущених значень.

3. Визначення параметрів: визначити параметри для врахування при визначенні пріоритету, такі як важливість задачі, терміни виконання, та залежності від інших завдань.

4. Навчання моделі: навчити модель на зібраних та підготовлених даних, використовуючи алгоритми машинного навчання.

5. Визначення пріоритету: використовуючи навчену модель, визначити пріоритет задачі на основі введених параметрів.

6. Визначення часу виконання: Прогнозувати час виконання задачі, використовуючи навчену модель та введені параметри.

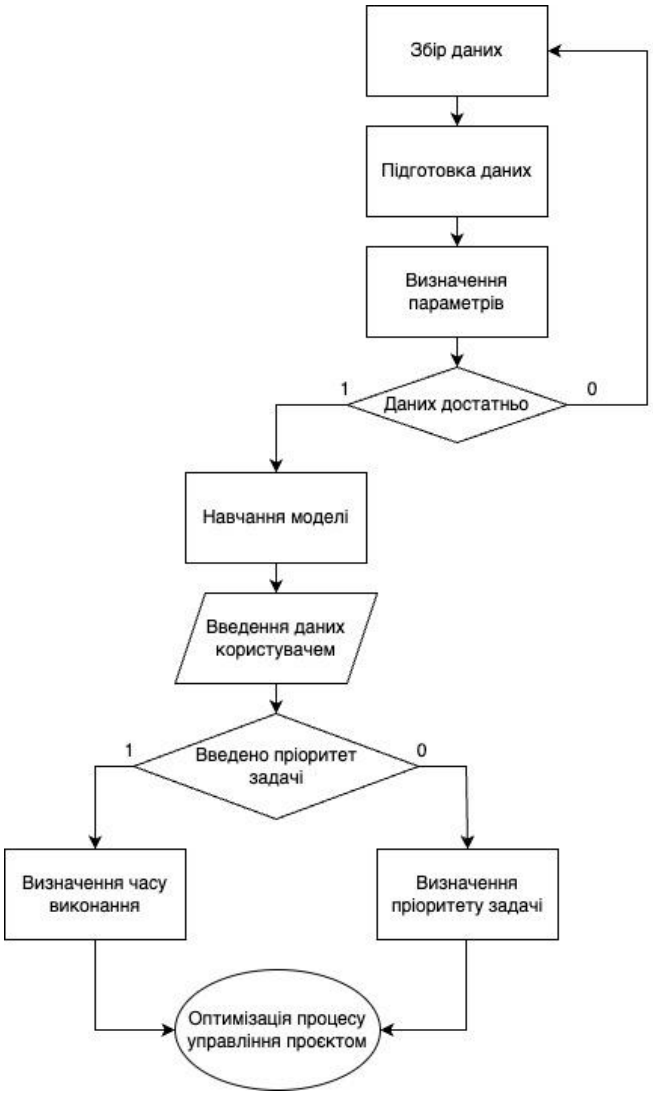

Рисунок 2.1 – Алгоритм методу визначення пріоритетності та часу задачі

Примітки:

— алгоритм може бути доповнений залежно від конкретних вимог та характеристик проєкту;

— періодично перенавчання моделі може бути необхідним для підтримки актуальності результатів.

Розроблений алгоритм дозволяє врахувати різноманітні фактори при визначенні пріоритету та часу виконання задачі, забезпечуючи оптимальне управління проєктом.

2.2. Проектування моделі системи з інтегрованою моделлю машинного навчання

Процес розробки моделі системи включає в себе декілька ключових етапів, починаючи з аналізу вимог та закінчуючи впровадженням та підтримкою.

1. Проектування архітектури:

— вибір архітектурного стилю: визначення, який архітектурний стиль буде найбільш ефективним для даної системи;

— розробка схеми бази даних: проектування структури бази даних, включаючи таблиці, зв'язки та індекси.

2. Розробка моделі:

— вибір технологій: визначення мов програмування, фреймворків та інших технологій, які будуть використовуватися для реалізації системи;

— розробка бізнес-логіки: реалізація коду, який виконує функціональні завдання системи.

3. Тестування та відладка:

— інтеграційне тестування: перевірка окремих компонентів та їх взаємодії для впевненості в їх коректності;

— системне тестування: перевірка системи в цілому для переконання у відповідності вимогам та виявлення можливих проблем.

5. Впровадження та підтримка:

— впровадження системи: перенесення системи з тестового середовища в робоче та забезпечення її нормальної роботи;

Загальна модель системи управління проєктами та задачами в командному середовищі зображено на рисунку 2.2:

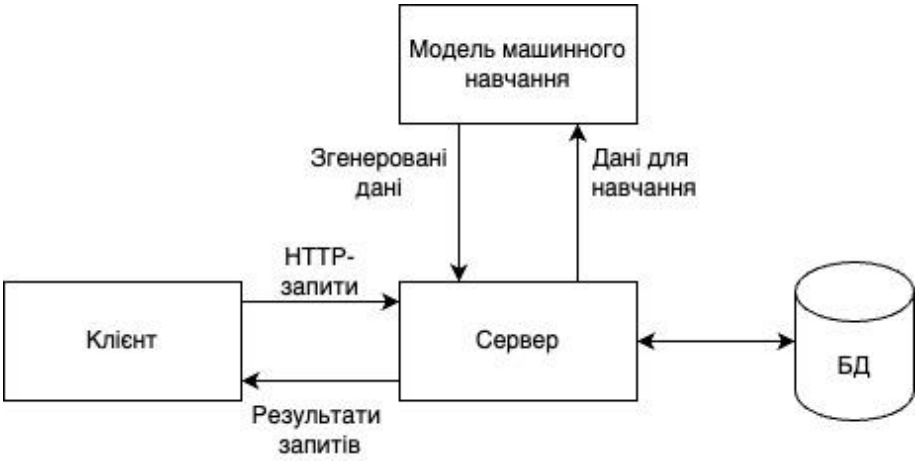

Рисунок 2.2 – Загальна модель системи

Клієнт (браузер) відправляє HTTP-затипи на сервер для отримання чи зміни інформації, сервер відбирає необхідні дані для тренування моделі машинного навчання, після генерації необхідних даних сервер враховує їх та відправляє їх разом з результатами HTTP-запитів клієнта.

2.3. Розробка архітектури системи

Архітектура програмного забезпечення підтримує аналіз якості системи, коли приймаються рішення щодо системи, а не після впровадження, інтеграції чи розгортання. Незалежно від того, чи йдеться про розробку нової системи, розробку успішної системи чи модернізацію застарілої системи, цей своєчасний аналіз дає змогу визначити, чи дадуть обрані ними підходи прийнятне рішення. Ефективна архітектура служить концептуальним зв’язком між кожними фазами проєкту разом для всіх його зацікавлених сторін, забезпечуючи гнучкість, економію часу та коштів і раннє визначення ризиків проектування [12, 13].

Правильно побудована архітектура допомагає:

— зменшити бізнес-ризики і загальні ризики, наприклад ризики для вартості, графіку, часу виходу на ринок тощо;

— дозволяють гнучку, ітераційну розробку;

— розробляти якісні продукти та послуги;

— модернізувати застарілі системи;

— оцінити існуючі архітектурні рішення;

— аналізувати альтернативи розробки системи.

Побудова ефективної архітектури, яка забезпечує швидке постачання продукту для сучасних потреб, а також досягнення довгострокових цілей, може виявитися складним завданням. Нездатність визначити та керувати компромісами між архітектурно значущими якостями часто призводить до затримок проєкту, дорогої переробки або ще гірше [13].

Для побудови програмного рішення з метою пришвидшення і полегшення розробки буде використана «безсерверна» архітектура застосунку.

«Безсерверний» вперше використовувався для опису додатків, які значною мірою або повністю включають сторонні програми та служби, розміщені в хмарі, для керування логікою та станом на стороні сервера. Зазвичай це «розширені клієнтські» програми — наприклад, односторінкові веб-програми або мобільні програми — які використовують величезну екосистему хмарних баз даних (наприклад, Parse, Firebase), служб автентифікації (наприклад, Auth0, AWS Cognito) і так далі. Ці типи служб описуються як «(мобільний) бекенд як послуга», або «BaaS» як скорочення [14].

Завдяки такій архітектурі клієнтська сторона застосунку може бути відносно «нерозумною», оскільки більша частина логіки системи — автентифікація, навігація сторінками, пошук, транзакції — реалізується серверною програмою.

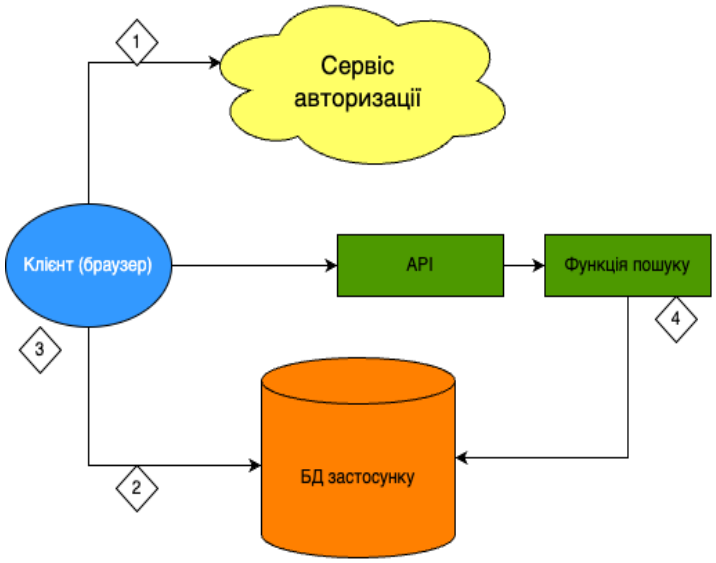

Рисунок 2.3 – Схема роботи застосунку з безсерверною архітектурою

Основні аспекти роботи програмного засобу з безсерверною архітектурою:

1) Сервіси автентифікації розміщуються на BaaS–сервісі.

2) Клієнтська сторона застосунку має прямий доступ до підмножини бази даних, яка сама повністю розміщена на BaaS–сервісі. З мір безпеки доречно забезпечити більший рівень безпеки для клієнта, що має доступ до бази даних, ніж для ресурсів сервера, які звертаються до бази даних.

3) Два попередні пункти означають дуже важливий третій: певна логіка, яка при використанні іншої архітектури застосунку була б на сервері, тепер знаходиться всередині клієнта, наприклад, відстеження сеансу користувача, розуміння UX-структури програми, читання з бази даних і перетворення цього в зручне для клієнта подання тощо.

4) Деякі функції, пов’язані з UX, всеодно повинні зберігатись на сервері, якщо, наприклад, вони мають інтенсивні обчислення або потребують доступу до значних обсягів даних. Замість постійно запущеного сервера, як це є в інших архітектурах, можна реалізувати функцію, яка відповідає на запити HTTP через шлюз API.

У «класичних» архітектурах, контролем і безпекою керує центральна серверна програма. У безсерверній версії немає центрального вирішення цих

питань. Натомість ми бачимо перевагу «хореографії над оркестровкою», коли кожен компонент відіграє більш архітектурно обізнану роль — ідея також поширена в підході до мікросервісів. Системи, побудовані таким чином, часто більш гнучкі та легше піддаються змінам як у цілому, так і через незалежні оновлення компонентів; є кращий розподіл залежностей; а також є деякі вражаючі економічні переваги підтримки серверної частини [15].

Звичайно, така конструкція є і в свою чергу компромісом: вона вимагає кращого розподіленого моніторингу, і ми більше покладаємося на можливості безпеки обраного сервісу. Те, чи варті переваги гнучкості та вартості додаткової складності багатьох серверних компонентів, дуже залежить від контексту.

2.4. Проектування діаграми варіантів використання

Для розуміння можливості дій користувача у застосунку доцільно буде спроектувати діаграму використання.

Діаграма використання визначає функціональність системи з точки зору її користувачів. Це набір взаємодій між акторами (користувачами чи іншими системами) та системою, що допомагає зорієнтувати команду розробників та зацікавлених сторін щодо того, як система повинна взаємодіяти з реальним світом [16]. Діаграма використання - це не тільки інструмент визначення вимог, але й основа для подальших етапів проектування та розробки.

Компоненти діаграми:

— актори - особи, які взаємодіють з системою. Це можуть бути користувачі, зовнішні системи або навіть інші програми;

— використання (Use Cases) - це конкретні функціональні можливості системи, що відповідають на потреби акторів;

— зв'язки - вказують на те, як актори взаємодіють з використаннями та один з одним.

Переваги діаграми використання:

— зрозумілість - легко зрозуміти навіть для не-технічних зацікавлених сторін;

— комунікація - ефективно передає вимоги та очікування користувачів команді розробників;

— орієнтованість на користувача - Допомагає розглянути систему з точки зору її фактичних користувачів.

Розглянемо варіанти використання застосунку (рисунок 2.4).

Рисунок 2.4 – Діаграма варіантів використання застосунку

Деталізуємо деякі з варіантів використання системи:

— реєстрація – застосунок дає можливість створення облікового запису за допомогою електронної пошти та паролю або Google-сервісів [17];

— створення новго проєкту – застосунок після авторизації відразу дає змогу створити новий проєкт;

— створення нової задачі всередині проєкту – застосунок дає змогу створити нову задачу для конкретного проєкту;

— редагування задачі – застосунок дає змогу змінити деталі створеної задачі чи видалити її;

— зміна статусу задачі – після створення задача автоматично матиме статус «створено». Надалі можна надати їй такі статуси як «в процесі» або «завершено».

— генерація інвайт-коду – автоматична генерація унікального коду для його подальшого використання іншим користувачем для отримання доступу до проєкту (робота в командному середовищі);

— використаня інвайт-коду – застосування попередньо згенерованого коду для отримання доступу до вже створеного проєкту.

2.5. Розробка логічної структури програмного засобу

Структура веб-застосунку є ключовим елементом його успішної реалізації та користувацького досвіду. Вона визначає, як інформація буде організована і доступна для користувачів, як навігація буде впорядкована і як користувачі можуть взаємодіяти з різними частинами додатку [18].

Логічна структура додатку визначає основні секції, їх функціонал та взаємозв'язки. Це передбачає, що користувачі можуть з легкістю переходити від однієї частини додатку до іншої, знаходячи потрібну інформацію або використовуючи потрібні інструменти.

При проектуванні логічної структури застосунку, що включає в себе сторінки та навігацію між ними, варто дотримуватись наступних принципів:

— користувацька доступність: потрібно забезпечити, щоб сторінки були доступні для всіх користувачів, включаючи тих, хто використовує адаптивні технології та інструменти читання вголос;

— простота та лаконічність: варто уникати перевантаження сторінок зайвою інформацією. Вибирайте лише найважливіші та найактуальніші елементи для відображення;

— інтуїтивна навігація: необхідно створити легку та зрозумілу навігацію, яка дозволяє користувачам легко знаходити потрібну інформацію, використовувати зрозумілі назви меню та кнопок;

— консистентність дизайну: необхідно зберігати стиль та дизайн сторінок однаковими, щоб створити єдиний ідентифікаційний стиль для всього веб-застосунку;

— швидкість завантаження: варто оптимізувати великі мультимедійні файли та зображення, щоб сторінки завантажувались швидко. Швидка відповідь сторінок позитивно впливає на користувацький досвід [19];

— застосування інтерактивних елементів: варто використовувати віджети та елементи взаємодії, які полегшують користувачам взаємодію з веб-застосунком, такі як кнопки, форми та анімації;

— перевірка на помилки: варто ретельно перевіряти сторінки на наявність помилок, які можуть впливати на коректність відображення чи функціональність;

— забезпечення контекстуальної інформації: варто додавати достатньо контекстуальної інформації та підказок, щоб користувачі завжди розуміли, що відбувається на сторінці та як користуватися різними функціями веб-застосунку.

Логічна структура застосунку зображена на рисунку 2.5:

Рисунок 2.5 – Логічна структура застосунку

Після запуску застосунку користувач попадає на сторінку авторизації. При відсутності створеного акаунту, користувач має можливість зареєструвати новий. Детальніше розглянемо алгоритм авторизації у системи:

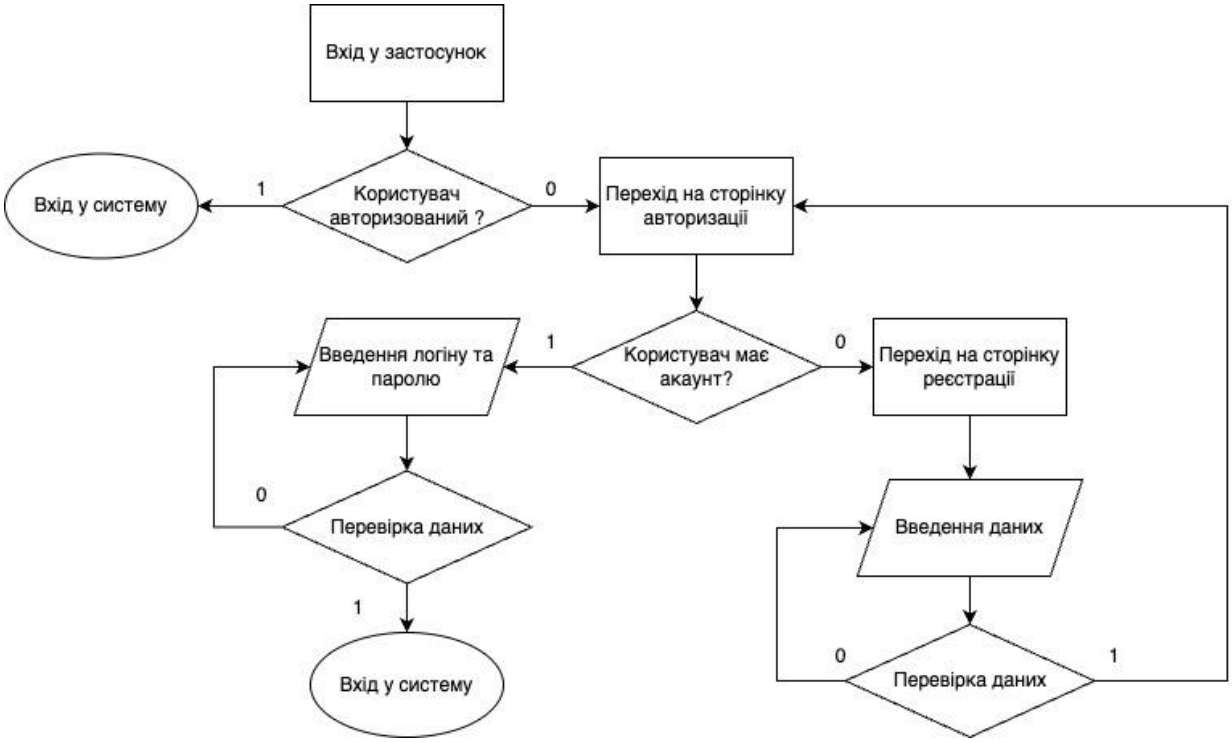

Рисунок 2.6 – Алгоритм авторизації у системі

Далі користувач потрапляє на головну сторінку, на якій зображені усі проєкти, до яких користувач має доступ. Звідси користувач може потрапити на сторінку створення нового проєкту. Після створення нового проєкту користувач перенаправляється на сторінку тільки що створеного проєкту, де зображена уся основна інформація про проєкт та його задачі. На даній сторінці користувач може управляти задачами проєкту та відслідковувати їх статус. Алгоритм створення нової задачі з використання алгоритму визначення пріоритетності та часу завдання з використанням машинного навчання зображено на рисунку 2.7. З цієї сторінки користувач може потрапити на сторінку налаштування проєкту, на якій він має змогу змінити основну інформацію про проєкт, ресурси та інше.

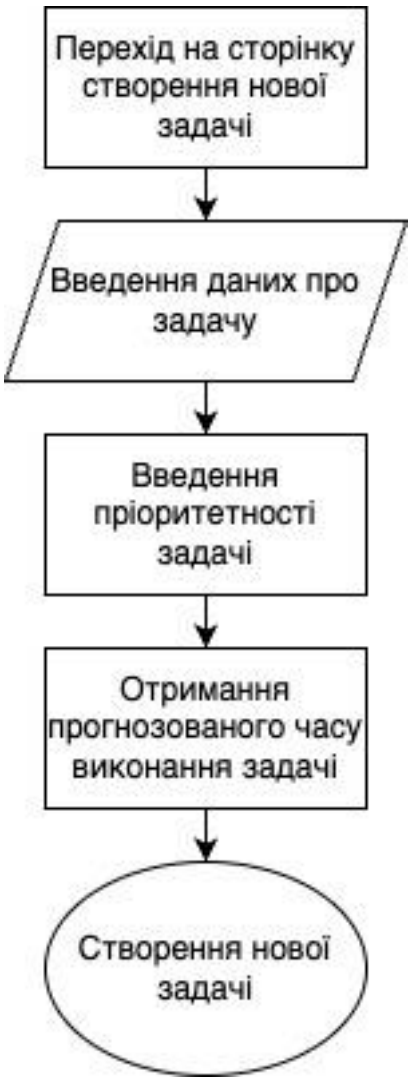

Рисунок 2.7 – Алгоритм створення нової задачі

2.6. Розробка фізичної структури програмного засобу

Фізична структура веб-застосунку визначає організацію програмного коду, зображення, файлів та інших ресурсів. Це важливий аспект розробки, оскільки правильна фізична структура може поліпшити продуктивність, зробити розробку більш ефективною та спростити підтримку застосунку [20].

При організації фізичної сруктури веб-застосунку варто дотримуватись наступних принципів:

— розділення коду застосунку на модулі та класи. Кожен модуль або клас повинен виконувати конкретну функцію або мати певний набір пов'язаних функцій;

— робити код модульним, що означає, що кожна частина програми (функція, метод, клас) повинна виконувати лише одну задачу. Це спрощує тестування та перевикористання коду;

— необхідно додавайте як коментарі, так і документацію до коду. Коментарі повинні пояснювати незрозумілий код, а документація повинна надавати зрозуміле пояснення того, що робить функція або клас;

— розділення HTML-шаблони та CSS-стилів на окремі файли. Це дозволяє підтримувати чистоту та читабельність веб-сторінок;

— варто постійно оновлювати та оптимізувати код, видаляючи непотрібні або застарілі частини та забезпечуючи високу продуктивність.

Фізична структура застосунку зображена на рисунку 2.8:

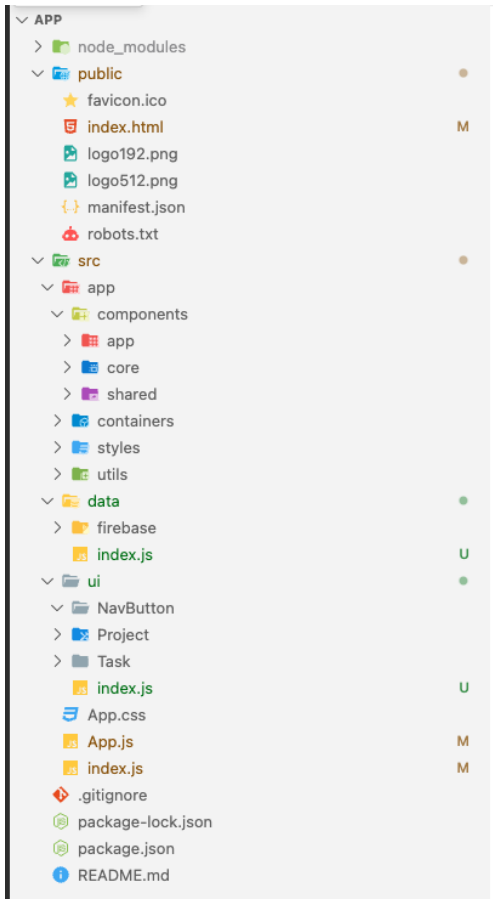

Рисунок 2.8 – Фізична структура застосунку

Основні компоненти фізичної структури застосунку:

— public – містить головний компонент кожного веб-застосунку – index.html, що містить у собі посилання на JavaScript-скрипти;

— src – головна директорія застосунку;

— app – містить у собі усі сторінки застосунку, контейнери (навігація між сторінками), стилі та утиліти застосунку;

— data – директорія, що містить у собі код, пов’язаний із взаємодією з сервером, у нашому ж випадку – з BaaS-сервісом Firebase;

— ui – містить у собі всі перевикористані компоненти, наприклад Task – компонент задачі у проєкті.

2.7. Проектування бази даних застосунку

Проектування бази даних для веб-застосунку є ключовим етапом в розробці, оскільки вона забезпечує зберігання та організацію даних, необхідних для правильної роботи застосунку. Виділимо кілька важливих кроків та принципів проектування бази даних:

— необхідно ретельно зрозуміти вимоги до даних веб-застосунку. Варто визначити, які дані зберігатимуться, як вони пов'язані між собою та як вони будуть використовуватися в застосунку;

— необхідно провести нормалізацію даних, щоб уникнути дублювання інформації та забезпечити консистентність даних. Нормалізація розбиває таблиці на менші, що ускладнює структуру, але робить їх більш ефективними та менш схильними до помилок;

— необхідно створити таблиці для кожної сутності (наприклад, користувачі, продукти, замовлення) та визначити стовпці для кожної таблиці (наприклад, ідентифікатор, ім'я, ціна тощо);

— необхідно визначити взаємозв'язки між таблицями. Наприклад, проєкт може бути пов'язаний з користувачем через унікальний ідентифікатор користувача. Використовуйте зовнішні ключі для встановлення зв'язків;

— необхідно додати індекси до таблиць для швидкого пошуку та сортування даних. Індекси підвищують продуктивність запитів до бази даних;

— варто приділити увагу безпеці даних, особливо якщо база даних містить конфіденційну інформацію. Необхідно використовувати захисні методи, такі як хешування паролів, щоб забезпечити безпеку користувачів;

— варто документувати структуру бази даних, включаючи таблиці, стовпці, зв'язки та індекси. Це допоможе розробникам, які будуть працювати з базою даних у майбутньому;

При проектуванні архітектури веб-застосунку була обрана безсерверна архітектура, тому варто обрати найлегшу у використанні СУБД. Сервіс Firebase надає можливість використовувати вбудоване рішення для створення баз даних – Firebase Firestore.

Firestore - це NoSQL база даних від Google, спроектована для зберігання та синхронізації даних у реальному часі [21]. Вона використовує колекції для групування документів та може легко масштабуватися від невеликих проєктів до великих додатків з мільйонами користувачів.

Кожен документ в Firestore - це набір полів, які можуть містити різні типи даних. Документи групуються в колекції, які можна порівняти з таблицями у звичайних реляційних базах даних. Для звертання до конкретного документа вказується його ідентифікатор, і він може знаходитися в колекції чи вкладений в інший документ.

Одна з ключових особливостей Firestore - це здатність до синхронізації даних в реальному часі. Це означає, що якщо дані змінюються в базі даних, зміни автоматично відображаються на всіх підключених клієнтах без необхідності ручного оновлення.

Firestore також надає потужні можливості запитів та фільтрації даних, що дозволяє отримувати лише необхідну інформацію. Вона інтегрована з іншими сервісами Firebase, такими як аутентифікація та авторизація, що робить її універсальним інструментом для розробки різних видів веб-застосунків.

Firestore також дозволяє працювати офлайн. Якщо пристрій втратить з'єднання з Інтернетом, зміни, внесені користувачем, будуть збережені локально і синхронізовані з базою даних, коли з'єднання буде відновлено.

Firestore автоматично масштабується з ростом додатку, забезпечуючи стабільну роботу навіть при великих навантаженнях. Firebase, як власник Firestore, бере на себе управління серверами та забезпечує високу доступність вашої бази даних.

Firestore надає потужні інструменти для налаштування прав доступу до даних. Є змога контролювати, хто може читати та записувати дані в базі даних, і встановлювати правила автентифікації для захисту від несанкціонованого доступу.

Перш за все було створено таблицю User. В ній містяться наступні поля:

— id – унікальний ідентифікатор;
— firstName – ім’я;
— lastName – фамілія;
— email – електронна пошта;
— projects – проєкти, до яких має доступ.

Поле projects пов’язано зв’язком один до багатьох з таблицею Project, яка містить дані поля:

— id – унікальний ідентифікатор;
— name – ім’я;
— resources – ресурси проєкту;
— tasks – задачі проєкту.

Поле tasks пов’язано зв’язком один до багатьох з таблицею Task, яка містить дані поля:

— id – унікальний ідентифікатор;
— name – ім’я;
— dateStart – час початку задачі;
— duration – запланована тривалість задачі;

— resources – ресурси задачі;

— priority – приорітетність задачі.

Поле resources пов’язано зв’язком один до багатьох з таблицею Resource, яка містить дані поля:

— id – унікальний ідентифікатор;

— name – ім’я;

Схема бази даних зображена на рисунку 2.9:

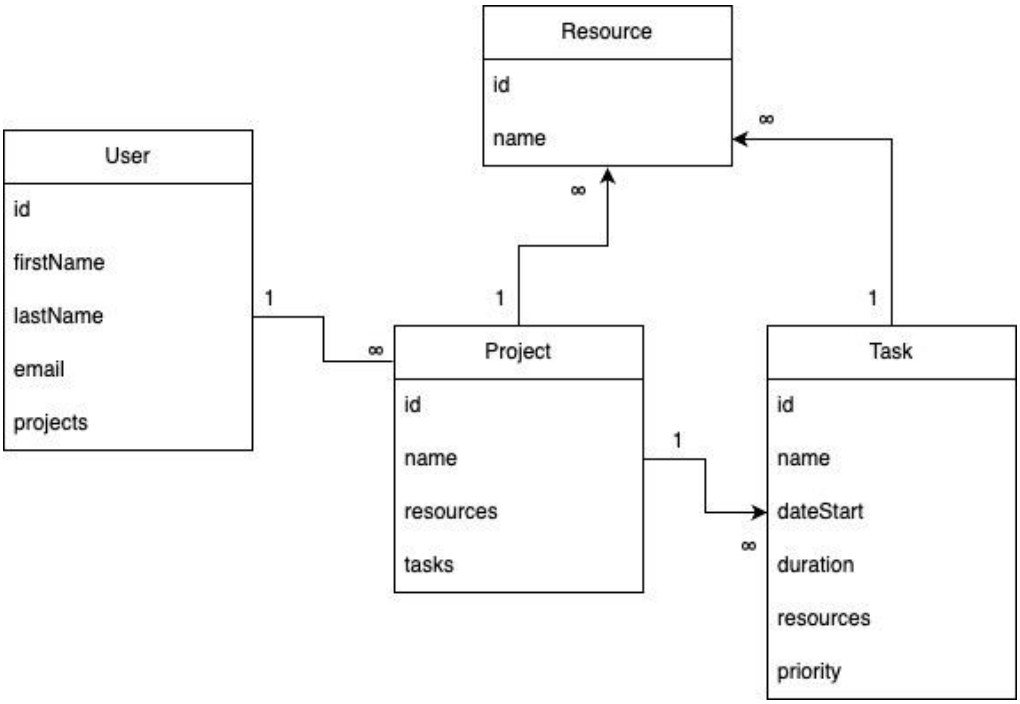

Рисунок 2.9 – Схема бази даних застосунку

2.8. Проектування інтерфейсу застосунку

Створення інтерфейсу веб-застосунку - це важливий етап розробки, який включає в себе проектування та реалізацію користувацького інтерфейсу для взаємодії з користувачем. Нижче подано кроки та найкращі практики для створення ефективного та зручного інтерфейсу веб-застосунку:

Розуміння аудиторії користувачів:

— проведення аналізу цільової аудиторії: хто вони, які у них потреби та очікування.

Проектування інтерфейсу:

— wireframing: створення макетів екранів, що відображають структуру та компоненти;

— mockups: розробка деталізованих макетів з графікою та текстом;

— prototyping: створення прототипу для інтерактивного тестування функцій.

Дизайн та вигляд:

— кольори та стиль: необхідно обрати палітру кольорів, шрифти та стилі для вигляду додатку;

— мобільний дизайн: розгляд адаптації інтерфейсу для мобільних пристроїв.

Розробка інтерфейсу:

— HTML/CSS/JavaScript: використання HTML для структури, CSS для стилізації та JavaScript для динамічних ефектів;

— фреймворки: розгляд використання CSS фреймворків (наприклад, Bootstrap) для швидкої розробки та адаптивності;

— реактивні бібліотеки: при використанні фронтенд фреймворків (наприклад, React.js або Vue.js), використовувати його компоненти для реактивності.

Взаємодія та навігація:

— меню та навігація: забезпечення легкої навігації між різними частинами додатку;

— форми та валідація: розробка інтуїтивної форми вводу даних та її валідація для уникнення помилок.

Тестування та вдосконалення:

— адаптивність: інтерфейс адаптивний та добре виглядає на різних пристроях та в різних браузерах;

— Ттстування користувацького досвіду (UX): проведення тестів з користувачами для визначення зручності взаємодії та виправлення можливих проблем.

Оптимізація та швидкість:

— завантаження: оптимізація графічних елементи для швидкого завантаження сторінок;

— кешування та запити: використання кешування та оптимізація запитів до сервера для швидкодії.

Підтримка та оновлення:

— підтримка користувачів: забезпечення механізмів зворотного зв'язку для користувачів;

— регулярні оновлення: оновлення інтерфейсу з часом, додаючи нові функції та вдосконалюючи існуючі.

2.9. Розробка прототипу дизайну застосунку

Перед програмною реалізацією застосунку варто розробити прототип дизайну застосунку. Розробка дизайн-макета перед початком розробки застосунку є ключовим етапом у процесі створення програмного продукту. Це необхідно з кількох причин.

По-перше, дизайн-макет візуалізує ідеї і концепції, що допомагає зрозуміти, як саме буде виглядати застосунок після завершення розробки. Він надає можливість побачити кожну сторінку, кожен елемент і взаємодію між ними.

По-друге, дизайн-макет служить точним джерелом інформації для розробників. Він вказує, які елементи повинні бути на кожній сторінці, як вони повинні виглядати та як взаємодіяти з користувачем. Це усуває непорозуміння та непродуктивність під час процесу розробки.

По-третє, дизайн-макет допомагає визначити пріоритети в дизайні, розміщення елементів на сторінці, а також забезпечує єдність стилю та кольорової палітри. Це забезпечує консистентність та професійний вигляд застосунку.

Крім того, дизайн-макет може служити засобом комунікації між різними учасниками проєкту, включаючи замовників, дизайнерів та розробників. Він допомагає уточнити вимоги та очікування всіх сторін, що забезпечує успішний результат у розробці застосунку.

Для виконання дизайн-макету буде використано інструментальне програмне забезпечення Figma. Figma - це інструмент для дизайну та прототипування, який дозволяє користувачам створювати візуальні компоненти для веб-сайтів та мобільних додатків [22]. Однією з ключових переваг Figma є можливість працювати в режимі реального часу, навіть якщо користувачі знаходяться на великій відстані один від одного.

За допомогою Figma, користувачі можуть виготовляти векторні графічні елементи, іконки та ілюстрації. Користувачі можуть визначити стилі, такі як кольори та шрифти, щоб забезпечити єдність в дизайні проєкту.

Ще однією особливістю Figma є можливість створення інтерактивних прототипів. Користувач може додавати взаємодію на свої макети, створюючи переходи між сторінками, анімацію та інші взаємодійні ефекти. Це дозволяє перевіряти функціональність та навігацію до того, як розпочати фактичну розробку застосунку.

Однією з важливих функцій є можливість спільної роботи над проєктами в режимі реального часу. Кілька користувачів можуть одночасно редагувати проєкти та залишати коментарі, що полегшує комунікацію в команді.

Figma також надає можливість зручного експорту готових дизайнів в різних форматах, включаючи PNG та SVG. Інтеграція з іншими інструментами та сервісами також робить Figma універсальним інструментом для дизайну та прототипування.

Перш за все, варто розробити дизайн сторінки авторизації (рисунок 2.10). Сторінка містить у собі два поля вводу – для логіна та пароля користувача відповідно. Крім того, сторінка містить дві кнопки – перша для

авторизації, друга – для переходу на сторінку для реєстрації. Додатково користувач має змогу увійти до системи за допомогою сервісів Google.

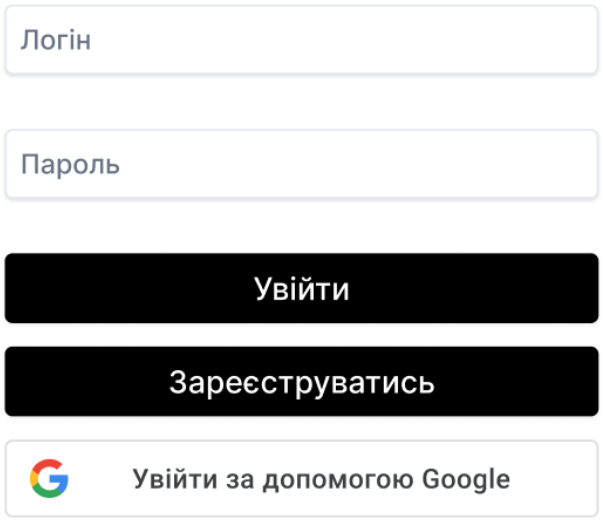

Рисунок 2.10 – Сторінка-форма авторизації

Наступною розробленою сторінкою буде сторінка реєстрації (рисунок 2.11). Сторінка містить у собі форму реєстрації, а саме поля для вводу логіну, паролю, підтвердження паролю та ім’я. Крім цього, сторінка містить кнопку для повернення на сторінку авторизації.

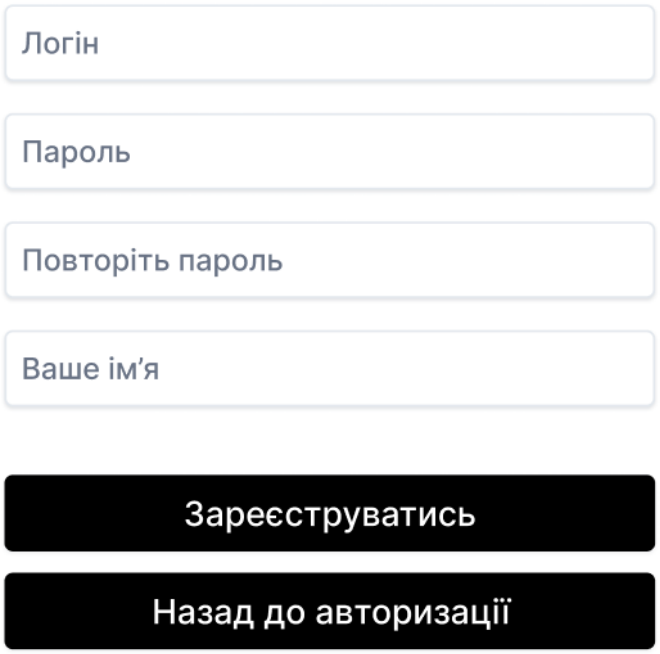

Рисунок 2.11 – Сторінка-форма реєстрації

Після успішної авторизації користувач потрапляє на головну сторінку (рисунок 2.12). Дана сторінка містить у собі, перш за все, меню навігації зверху екрану. Меню містить у собі кнопку переходу на головну сторінку та кнопку для виходу з системи, сама ж сторінка містить у собі кнопку для створення нового проєкту, а також кнопку для приєднання до існуючого проєкту за допомогою «інвайт-коду». Під даними кнопками знаходиться список проєктів, до яких користувач має доступ.

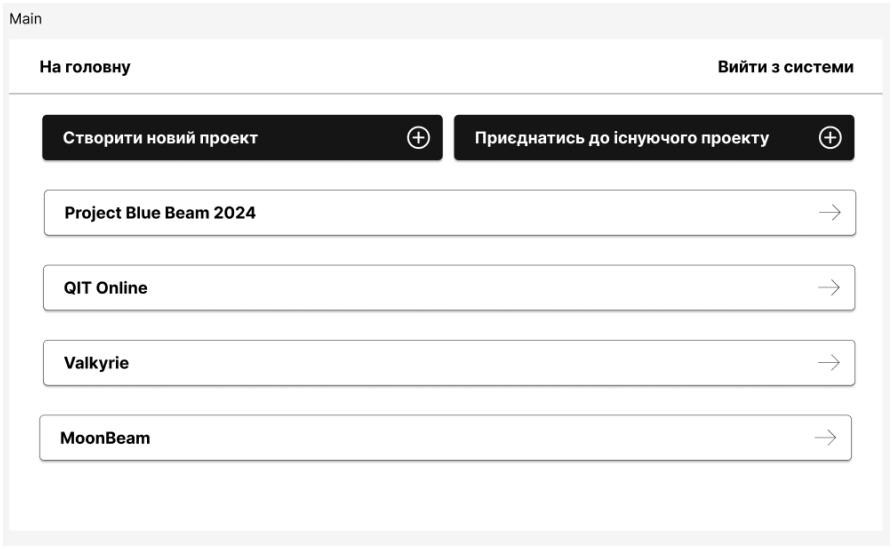

Рисунок 2.12 – Головна сторінка

Варто створити дизайн-прототип сторінки створення нового проєкту (рисунок 2.13). Сторінка міститиме у собі поле для вводу назви проєкту, кнопку для підтвердження створення проєкту та кнопку для повернення до головної сторінки. Усі інші деталі можна буде налаштувати далі на сторінці налаштування проєкту.

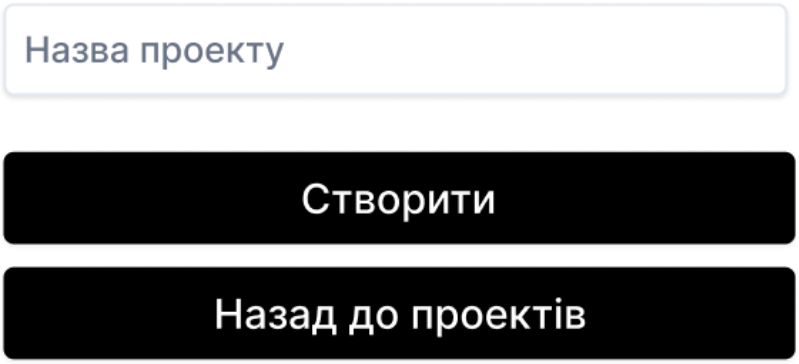

Рисунок 2.3 – Сторінка-форма створення нового проєкту

На рисунку 2.14 зображено детальний огляд задач проєкту.

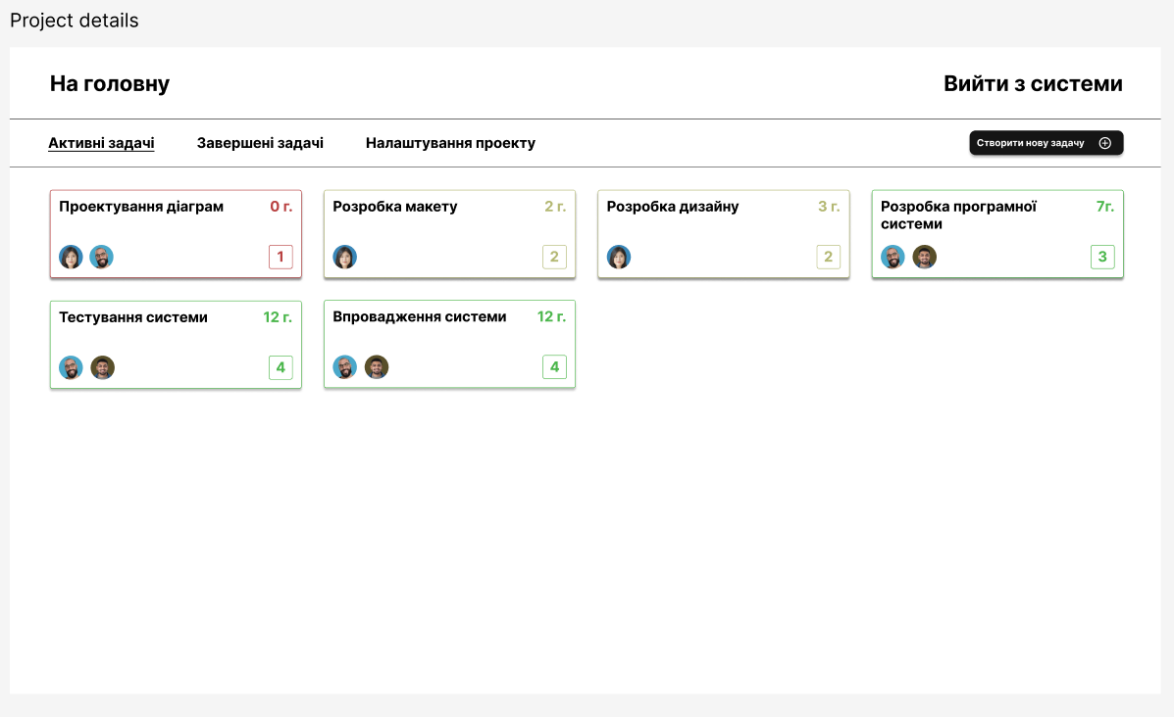

Рисунок 2.14 – Сторінка задач проєкту

Сторінка містить у собі додаткову навігацію, а саме сторінку активних задач, завершених задач та налаштування проєкту. Також у навігації міститься кнопка створення нової задачі у проєкті. Самі ж задачі містять у собі назву, час, що залишився на виконання, відповідальних осіб та приорітетність. При натисканні на задачу відкривається детальний огляд задачі з можливістю внести зміни.

На рисунку 2.15 зображено форму створення нової задачі. Форма дає змогу вказати назву задачі, тривалість, відповідальних осіб, ресурси та пріоритетність. Додатково форма містить підказки від навченої моделі машинного навчання для обирання оптимальних значень. Додатково модель підказує, що створення даної задачі може сильно вплинути на строки ітерації та проєкту загалом.

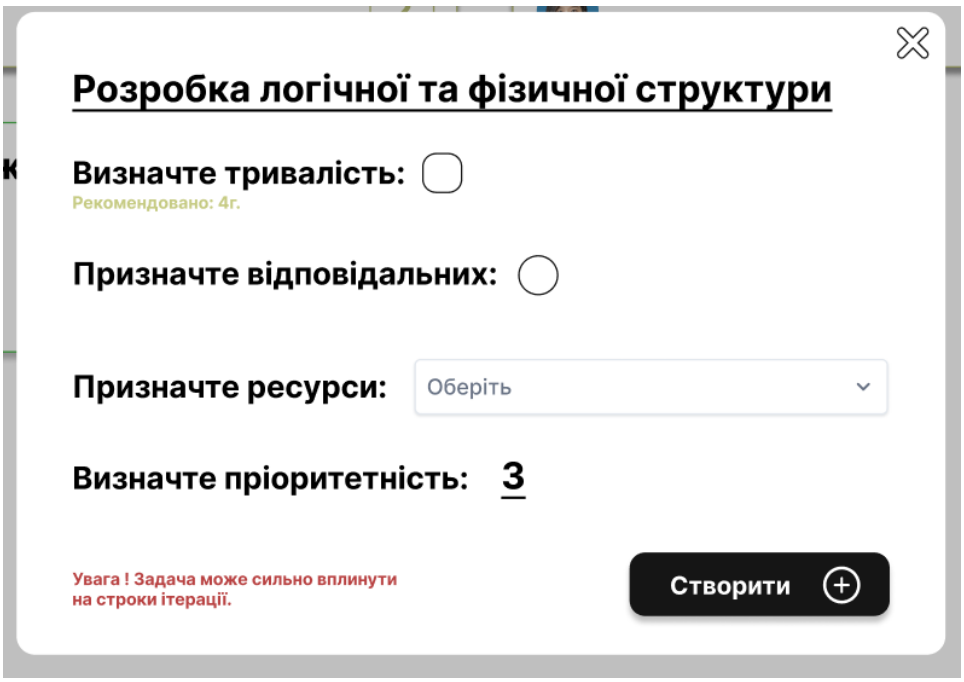

Рисунок 2.15 – Форма створення нвоої задачі

2.10. Висновок

В ході моделювання інформаційної системи управління проєктами та задачами в командному середовищі було успішно виконано ряд завдань, а саме:

Спроектовано архітектуру системи. В ході аналізу завдань системи була обрано використання «безсерверної» архітектури.

Для візуального представлення можливості дій користувача у системі була спроектована діаграма варіантів використання.

Розроблено логічну та фізичну структуру застосунку, визначено основні сторінки застосунку та зв’язки між ними.

Спроектовано базу даних застосунку, визначено основні таблиці та зв’язки між ними.

Розроблено дизайн-макети основних сторінок застосунку, а саме: сторінка авторизації, сторінка реєстрації, головна сторінка, сторінка детального огляду проєкту, сторінка створення проєкту, сторінка створення нової задачі у проєкті.

3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАСОБУ УПРАВЛІННЯ ПРОЄКТАМИ ТА ЗАДАЧАМИ В КОМАНДНОМУ СЕРЕДОВИЩІ

3.1. Варіантний аналіз та обґрунтування вибору засобів реалізації

На основі проведеного у першому розділі аналізу аналогів було обрано створення веб застосунку. Класичні веб застосунки складаються з двох частин: клієнтський код та код на стороні серверу [23]. Як уже зазаначалось у другому розділі, для створення застосунку була обрана безсерверна архітектура, тому в якості сервеної частини буде використано BaaS-сервіс. Розглянемо деякі можливі варіанти:

Kinvey

Kinvey - це BaaS-платформа, яка надає розробникам інструменти для швидкого розгортання та управління мобільними та веб-додатками в хмарі [24]. Цей сервіс надає широкий спектр функціональності, включаючи зберігання даних, автентифікацію користувачів, геолокацію, пуш-сповіщення, інтеграцію з сторонніми службами та багато іншого.

Однією з ключових переваг Kinvey є його простота використання та гнучкість. Завдяки готовим компонентам та інтуїтивному інтерфейсу, розробники можуть швидко створити додатки без значних зусиль у розробці backend-інфраструктури.

Kinvey також відомий своєю високою масштабованістю та можливістю реалізації в реальному часі. Він надає можливість створювати додатки, які здатні обробляти великий обсяг даних та забезпечувати миттєву реакцію на зміни.

Крім того, Kinvey дозволяє розробникам використовувати власні скрипти та логіку на стороні сервера, що розширює можливості кастомізації та дозволяє втілити у життя навіть найскладніші вимоги проєкту.

Переваги Kinvey:

— простота використання та готові компоненти для швидкого розгортання мобільних додатків;

— розширений функціонал, включаючи зберігання даних, аутентифікацію, геолокацію та інтеграцію зі сторонніми сервісами;

— масштабованість та можливість реалізації реального часу.

Недоліки Kinvey:

— вартість користування може бути високою для розширених застосунків;

— обмежена кількість безкоштовних опцій у платних планах.

AWS Amplify (Amazon Web Services)

AWS Amplify - це BaaS-сервіс, який розроблено для швидкої та ефективної розробки мобільних та веб-додатків [25]. Він надає розробникам інструменти для спрощення розгортання, інтеграції та масштабування додатків в хмарі. Однією з ключових переваг є глибока інтеграція з іншими Amazon Web Services, такими як Amazon DynamoDB, AWS Cognito та Amazon S3, що робить його потужним вибором для розробників, які вже використовують AWS-сервіси.

AWS Amplify підтримує різні програмні мови і фреймворки, що надає розробникам гнучкість у виборі технологій для своїх додатків. Крім того, сервіс надає спрощений процес розгортання, що дозволяє командам швидко виводити свої додатки на ринок та забезпечує автоматичне масштабування та надійність, що важливо для додатків з великою кількістю користувачів.

Також варто зазнчити, що AWS Amplify має вбудовану систему аналітики та моніторингу, яка дозволяє розробникам відстежувати використання додатків та аналізувати поведінку користувачів. Це надає цінні дані для прийняття управлінських рішень та вдосконалення функціоналу додатків.

Переваги AWS Amplify:

— має набір інструментів для розгортання інфраструктури та розробки мобільних і веб-додатків;

— широка підтримка різних програмних мов і фреймворків;

— висока масштабованість та надійність через інтеграцію з AWS-сервісами.

Недоліки AWS Amplify:

— ціни на використання AWS Amplify можуть бути високими, особливо для додатків з великою кількістю користувачів чи які потребують великих обсягів трафіку та зберігання даних;

— для деяких розробників може бути важко зрозуміти та правильно налаштувати всі параметри та можливості Amplify через його велику кількість опцій та конфігурацій.

Firebase

Firebase, розроблений Google, є однією з найпопулярніших і впливових платформ для розробки мобільних та веб-додатків у хмарі. Цей сервіс надає розробникам потужний інструментарій для створення динамічних та інтерактивних додатків з високою продуктивністю [26].

Однією з ключових особливостей Firebase є його широкий функціонал, який включає в себе базу даних в реальному часі, аутентифікацію, зберігання файлів, хостинг, пуш-сповіщення, аналітику та багато іншого. Firebase дозволяє розробникам зосередитися на самому додатку, не турбуючись про складності управління інфраструктурою.

Ще однією суттєвою перевагою Firebase є його легка інтеграція з іншими сервісами Google та сторонніми бібліотеками, що робить розробку більш ефективною та продуктивною. Крім того, Firebase надає можливість автоматичного масштабування, що означає, що додаток може зростати разом із збільшенням кількості користувачів без необхідності значних технічних зусиль.

Сервіс також пропонує багато інструментів для аналізу даних, що дозволяє розробникам відстежувати та розуміти поведінку користувачів, удосконалювати взаємодію з аудиторією та виробляти докладні стратегії для покращення додатку.

Переваги Firebase:

— широкий функціонал, включаючи базу даних в реальному часі, аутентифікацію, зберігання файлів, індексацію та аналітику;

— легко інтегрується з іншими Google-сервісами та сторонніми бібліотеками;

— безкоштовний план для початкового розвитку.

Недоліки Firebase:

— з підвищенням обсягу даних або трафіку може збільшити вартість обслуговування.

Firebase вирізняється своєю простотою використання та широким спектром можливостей для розробки як мобільних, так і веб-додатків. Наявність безкоштовного плану для початкового розвитку забезпечує можливість спробувати сервіс без великих витрат.

Загалом, враховуючи специфіку проєкту та технічні вимоги, Firebase є відмінним вибором завдяки своїй легкості використання, обширній функціональності та можливості безкоштовного використання на початковому етапі.

Надалі виділимо декілька фреймворків для ствоення клієнтської сторони застосунку та виберемо один з них:

Angular

Angular - це один із фреймворків для розробки веб-додатків. Цей фреймворк розроблений та підтримується Google і став вибором багатьох розробників у всьому світі. Angular визначається своєю компонентною архітектурою, яка забезпечує прозорий та швидкий розвиток великих веб-додатків [27].

Angular спроектований навколо концепції компонентів, які є основними будівельними блоками додатків. Кожен компонент відповідає за певний елемент інтерфейсу користувача та взаємодіє зі структурою DOM.

Angular надає потужний механізм для двостороннього зв'язування даних між шаблоном та компонентом. Це означає, що якщо дані змінюються в компоненті, вони автоматично оновлюються в шаблоні та навпаки. Також фреймворк використовує сервіси для організації логіки, яка не відноситься до представлення. Залежності можна внести в компоненти, що дозволяє використовувати сервіси для спільних завдань.

Додатки Angular розділяються на модулі, які допомагають впорядковувати код та спрощують його управління. Модулі можна легко використовувати для організації функціональності за логічними групами.

Недоліки Angular:

— Angular може виглядати складним для новачків через його велику кількість концепцій та обсяг функціональності;

— Angular додає значну кількість коду до кінцевого бандлу, що може збільшувати час завантаження та вимагати більше ресурсів в порівнянні з іншими фреймворками;

— іноді виникають затримки у виході оновлень та нових версій Angular, що може впливати на доступність нових функцій та виправлення багів;

— незважаючи на популярність, Angular менш популярний у порівнянні з React, що може впливати на величину спільноти та кількість готових рішень.

Vue

Vue - це легкий фреймворк для розробки веб-додатків та інтерфейсів користувача [28]. Він надає розробникам гнучкість та простоту, що робить його дуже популярним серед розробників усього світу.

Однією з ключових особливостей Vue є його компонентна архітектура, яка дозволяє розбивати користувацький інтерфейс на невеликі та самостійні компоненти. Це спрощує розробку та підтримку великих додатків, оскільки компоненти можна перевикористовувати в різних частинах додатку.

Vue також визначається двостороннім зв'язуванням даних, що дозволяє автоматично оновлювати інтерфейс при зміні даних та, навпаки, змінювати дані при взаємодії користувача з інтерфейсом. Це робить роботу з формами та введенням даних надзвичайно зручною та ефективною.

Ще однією сильною стороною Vue є його активна та дружня спільнота. Завдяки цій спільноті існують безліч ресурсів, таких як плагіни, бібліотеки компонентів, уроки та інші корисні матеріали, які допомагають розробникам вирішувати різноманітні завдання та розвивати свої навички.

Ще однією важливою особливістю є легкість інтеграції Vue з іншими технологіями та бібліотеками. Він може бути легко використаний зі стеком технологій, таких як Vuex для станового управління, Vue Router для роутінгу та Axios для HTTP-запитів.

Окрім цього, Vue здатний працювати на різних платформах, включаючи веб-браузери, мобільні пристрої та навіть сервери, завдяки можливості рендерінгу на боці сервера.

Хоча Vue є потужним та популярним фреймворком для розробки веб-додатків, він також має свої недоліки, які варто враховувати:

— у Vue немає жорстких стандартів, які визначають, як будуть виглядати додатки. Це може призвести до того, що розробники використовують різні підходи до розробки, що ускладнює співпрацю великих команд;

— у великих додатках компоненти Vue.js можуть стати складними та перевантаженими, особливо якщо не дотримуватись кращих практик розробки.;

— деякі старі версії браузерів можуть не повністю підтримувати всі можливості Vue.js, що може призвести до несправностей у додатках для користувачів, які використовують застарілі версії браузерів.

React

React - це бібліотека для розробки інтерфейсів користувача, яка дозволяє розробникам будувати високоякісні та ефективні веб-додатки [29]. Вона зосереджується на компонентній архітектурі, що спрощує створення складних додатків шляхом розбиття їх на малині, самостійні компоненти. Кожен компонент може мати свою власну логіку та відображення, що полегшує їхнє управління та повторне використання.

Однією з ключових особливостей React є використання віртуального DOM (Document Object Model), що дозволяє забезпечити швидше оновлення інтерфейсу. React здатний автоматично визначати оптимальні шляхи для оновлення дерева DOM, зменшуючи кількість фактичних маніпуляцій з реальним DOM та поліпшуючи продуктивність додатка.

Для управління станом додатка, React використовує концепцію "статів" (state) та "властивостей" (props). Стани використовуються для збереження внутрішнього стану компонентів, тоді як властивості передаються в компоненти ззовні та служать для конфігурації їхньої логіки та відображення.

Однією з важливих концепцій React є односторонній потік даних. Дані в додатку рухаються в одному напрямку, що полегшує слідкування за їхнім потоком та робить додаток більш передбачуваним.

React також дозволяє використовувати JSX (JavaScript XML) - розширення синтаксису JavaScript, яке дозволяє вбудовувати HTML-подібний код безпосередньо в JavaScript. Це робить роботу з інтерфейсом більш інтуїтивно зрозумілою та зручною.

Завдяки великій та активній спільноті розробників, у React є безліч сторонніх бібліотек та компонентів, які спрощують розробку та додають додатковий функціонал.

Окрім веб-додатків, React може бути використаний для розробки мобільних додатків з використанням React Native, що дозволяє розробникам створювати мобільні додатки для платформ Android та iOS, використовуючи

React-підоб'єкти. Як уже зазначалося, програмний засіб буде реалізовано як веб-застосунок з подальшим створенням мобільного та настільного застосунків, тому для реалізації веб-застосунку буде використано саме React.

3.2. Аналіз середовища розробки

Швидкість, ефективність та якість програмного продукту в значній мірі залежать від вибору правильного середовища розробки. Варто розглянути різні середовища розробки, які підтримують розробку з використанням бібліотеки React. Доцільним буде аналіз їх переваг та недоліків, функціональних можливостей та інструментів, які вони надають для полегшення процесу розробки. Детальний огляд цих середовищ допоможе обрати найбільш підходяще для сформованих вимог.

WebStorm

WebStorm, розроблене компанією JetBrains, - це потужне та інтелектуальне інтегроване середовища розробки (IDE) для роботи з веб-технологіями, включаючи React [30]. Ця платформа стала популярною серед розробників завдяки своїй простоті використання та водночас багатому функціоналу.

Однією з ключових особливостей WebStorm є його розуміння JavaScript та JSX. Інтелектуальний аналіз коду допомагає виявляти помилки, надавати рекомендації щодо оптимізації коду та прискорювати процес написання нового функціоналу. Завдяки інтеграції з іншими інструментами JetBrains, такими як дебаггер, контроль версій та термінал, розробник може ефективно управляти проєктом.

Окрім того, WebStorm має вбудовану систему автоматичного завершення коду (code completion), що полегшує написання коду та допомагає уникати синтаксичних помилок. Вона також виявляє можливі проблеми в реальному часі, надаючи користувачеві можливість виправити їх негайно.

Інтерфейс користувача WebStorm є інтуїтивно зрозумілим та добре організованим. Він надає зручний доступ до всіх інструментів та функцій через меню, контекстні меню та гарячі клавіші, що полегшує роботу розробника та забезпечує продуктивність.

Однією з переваг WebStorm є його інтеграція зі спільнотою. Через плагіни та розширення, які розробляють користувачі та сторонні розробники, IDE може бути налаштованою під індивідуальні потреби. Також, JetBrains надає широкий спектр навчальних ресурсів та підтримку для користувачів, що дозволяє швидко знайти відповіді на питання та розв'язати проблеми.

Виділимо недоліки WebStorm:

— WebStorm - це платний продукт, і його вартість може бути високою для студентів, новачків або розробників, які працюють над невеликими проєктами або проєктами з обмеженим бюджетом;

— WebStorm може бути важким для ресурсів комп'ютера, особливо при роботі з великими проєктами. Це може спричинити сповільнення роботи системи, особливо на менш потужних комп'ютерах;

— з великою кількістю функцій та опцій може виникнути небажаний ефект перенавантаження, коли користувачі не використовують більшість з них, але їхня присутність може збільшити складність інтерфейсу та збільшити споживання ресурсів;

— в деяких випадках, інтеграція з певними іншими інструментами та сервісами може бути складною або вимагати додаткових налаштувань.

Sublime Text

Sublime Text - це текстовий редактор, який здобув популярність серед розробників завдяки своїй легкості та потужності [31]. Його основною особливістю є швидкість роботи та низьке споживання ресурсів, що робить його ідеальним вибором для розробників, які шукають ефективний інструмент для редагування коду.

Один з головних аспектів Sublime Text - його розширені можливості налаштування та розширення. Розробники можуть налаштувати його відповідно до своїх вподобань, встановити різноманітні теми оформлення та розширення, що полегшує адаптацію редактора під конкретні потреби проєкту чи розробника.

Sublime Text також славиться своєю великою кількістю плагінів, які додають різноманітні функції та можливості. Це може бути все від автоматичного виведення коду до завершення синтаксичних конструкцій, що значно полегшує написання коду та зменшує кількість можливих помилок.

Однією з ключових особливостей Sublime Text є його швидкість. Він швидко завантажується, швидко реагує на команди та швидко обробляє великі файли. Це забезпечує зручність та продуктивність при роботі над проєктами будь-якої складності.

Sublime Text також підтримує використання вкладок та можливість поділу екрану, що полегшує навігацію між файлами та редагування більш складних структур.

Незважаючи на його переваги, Sublime Text має деякі обмеження. Наприклад, для повноцінної підтримки реактивних фреймворків, таких як React, може знадобитися встановлення спеціалізованих плагінів, які можуть бути складними для налаштування.

Хоча Sublime Text є популярним текстовим редактором з численними перевагами, він також має свої недоліки:

— Sublime Text має платну ліцензію. Безкоштовна версія може допускати небагато обмежень, і для повного функціоналу користувачам доведеться придбати ліцензію;

— порівняно з іншими редакторами, Sublime Text може мати менше вбудованих функцій та інструментів. Багато корисних функцій можна додати за допомогою плагінів, але це вимагає додаткового часу та налаштувань;

— порівняно з іншими IDE, Sublime Text може мати обмежену інтеграцію з іншими інструментами розробки, такими як системи керування версіями чи інструменти для автоматизації тестування;

— Хоча у Sublime Text є спільнота користувачів і плагінів, вона може бути менш активною порівняно з іншими редакторами та IDE, що може вплинути на швидкість вирішення проблем та наявність підтримки.

Visual Studio Code

Visual Studio Code (VS Code) - це безкоштовне, відкрите та легке середовище розробки, розроблене Microsoft [32]. Ця IDE здобула популярність серед розробників завдяки своєму широкому функціоналу, легкості використання та величезній спільноті користувачів.

Однією з головних переваг VS Code є його розширені можливості налаштування. Користувачі можуть адаптувати редактор під свої потреби завдяки великій кількості розширень та тем оформлення. Вбудована підтримка Git полегшує роботу з версійним керуванням та співпрацю в команді.

Однією з ключових особливостей VS Code є його інтегрована відладка. Розробники можуть легко відстежувати та виправляти помилки в своєму коді, використовуючи різні режими відладки. Це робить процес розробки більш ефективним та допомагає зменшити кількість помилок.

VS Code також володіє потужними інструментами для роботи з розширеними мовами, такими як TypeScript та JavaScript. Редактор має автозавершення коду, інтегровані інструменти для рефакторингу та підтримку ES6+ функціональності.

Інтерфейс користувача VS Code вражає своєю простотою та зручністю. Лаконічний та інтуїтивно зрозумілий інтерфейс забезпечує швидку навігацію, а вбудований термінал дозволяє взаємодіяти з системою прямо з редактора.

VS Code також відзначається великою активністю та підтримкою спільноти. Через розширення та плагіни, які розробляють як сама Microsoft,

так і незалежні розробники, VS Code може бути легко налаштований для різних типів розробки.

З огляду на велику популярність, розширену функціональність, безкоштовну ліцензію та активну спільноту, Visual Studio Code (VS Code) може бути найкращим вибором для розробки програмного засобу за допомогою React. Він надає потужний та зручний інструментарій для розробки веб-додатків з React та забезпечує високу продуктивність та якість коду.

3.3. Початкова генерація програмного коду

Створимо новий React-додаток. Зазвичай при створенні нових React-застосунків використовується інструмент create-react-app [33].

create-react-app - це інструмент командного рядка, розроблений Facebook, що дозволяє швидко створювати нові проєкти React.js з мінімальними зусиллями щодо налаштувань та конфігурацій.

Цей інструмент вбудовує в себе оптимальні налаштування, необхідні для розробки React-додатків, такі як Webpack, Babel, ESLint та інші [34]. Він автоматично налаштовує проєкт так, щоб розробники могли зосередитися на написанні коду, не турбуючись про налаштування інфраструктури застосунку.

Одна з ключових переваг create-react-app - це його простота використання. Щоб створити новий проєкт, користувачам просто потрібно виконати одну команду в терміналі, і інструмент зробить залишок за них. Це робить його ідеальним вибором для початківців, які тільки починають з React, а також для досвідчених розробників, які швидко хочуть запустити новий проєкт без витрат на налаштування.

Після запуску інструменту автоматично створюється новий уже налаштований React-додаток з усіма початково необхідними файлами.

3.4. Розробка програмних модулів застосунку

Програмну реалізацію застосунку слід розпочати з розробки модуля авторизації. Як уже зазначалося, в якості бекенд-сторони застосунку буде використовуватися BaaS-сервіс Firebase. Сервіс надає можливості авторизації для застосунків.

Перш за все, необіхдно створити новий проєкт у консолі Firebase (рисунок 3.1) та додати веб-застосунок (рисунок 3.2).

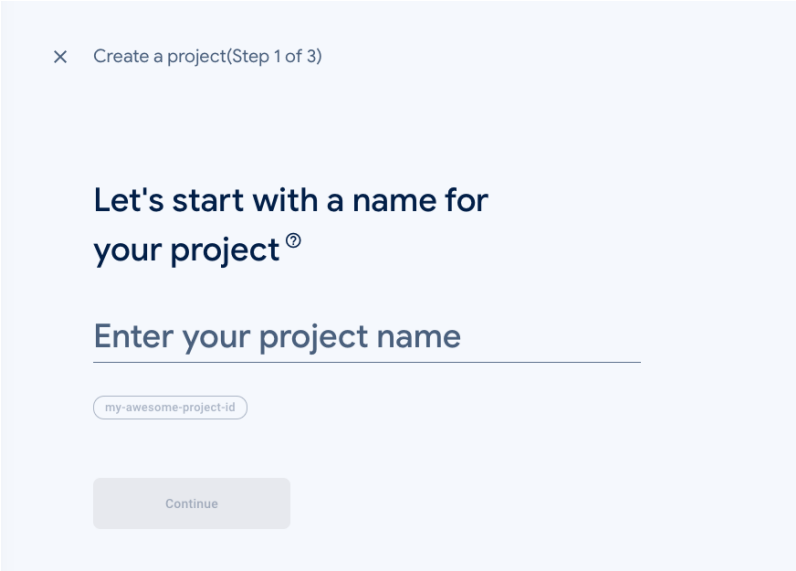

Рисунок 3.1 – Створення новго проєкту Firebase

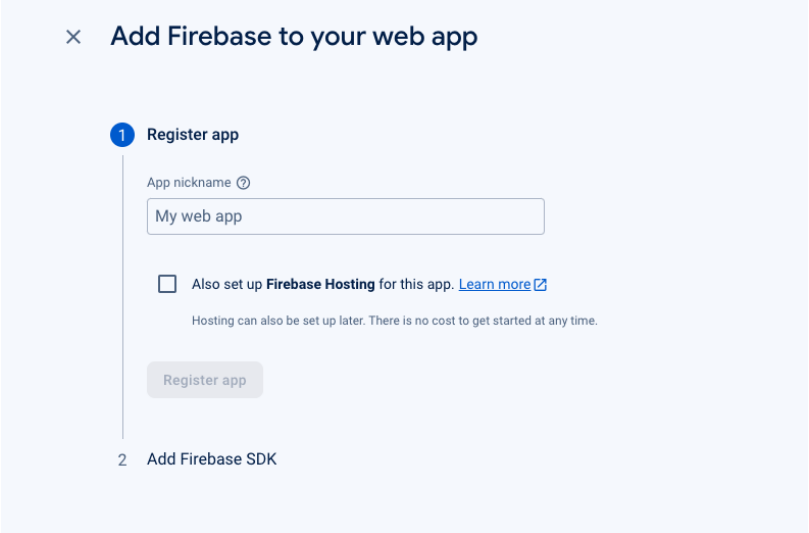

Рисунок 3.2 – Процес додавання веб-застосунку у проєкт Firebase

Для коректної роботи застосунку варто вистроїти роутинг. У React, роутинг використовується для управління навігацією та відображенням відповідних компонентів на основі шляхів URL. Один із популярних пакетів для роботи з роутингом в React - це react-router-dom.

react-router-dom надає зручний інтерфейс для визначення шляхів та компонентів, які мають бути відображені при відповідних URL. Основні компоненти, які використовуються для реалізації роутингу, - це BrowserRouter, Route, та Link [35].

Роутинг у створеному застосунку зображено на рисунку 3.3:

```
<Router>
  <Nav />
  <Routes>
    <Route path={"/"} element={<Projects />} />
    <Route path={"/create-project"} element={<CreateProject />} />
    <Route path={"/project/:projectId/active"} element={<ProjectTasks />} />
    <Route
      path={"/project/:projectId/inactive"}
      element={<ProjectTasks />}
    />
    <Route
      path={"/project/:projectId/settings"}
      element={<ProjectSettings />}
    />
    <Route path="*" element={<Navigate to="/" />} />
  </Routes>
</Router>
```

Рисунок 3.3 – Створений роутинг у застсоунку

Якщо користувач ще не авторизований, він матиме доступ лише до сторінок реєстрації та авторизації, у протилежному випадку він матиме доступ до всіх сторінок застосунку.

Створимо компонент SignIn – сторінку авторизації. Компонент повинен містити два поля вводу – для електронної пошти та паролю, дві кнопки – для входу та для переходу на сторінку реєстрації, та кнопку для авторизації за допомогою сервісів Google. Використовуючи дизайн-прототип застосунку, можна замітити, що поля для вводу та кнопки ідентичні між собою, тому варто створити окремі компоненти для кожної з них.

Програмний код компоненту кнопки Button зображено на рисунку 3.4.

```
export const Button = ({ className, ...props }) => (
  <button className={cx(s.root, className)} {...props}>
    {props.children}
  </button>
);
```

Рисунок 3.4 – Лістинг компоненту Button

Аналогічним чином створимо компонент Input для поля вводу (рисунок 3.5):

```
export const Input = ({ setValue, className, ...props }) => (
  <input
    onChange={(e) => setValue(e.target.value)}
    className={cx(s.root, className)}
    {...props}
  />
);
```

Рисунок 3.5 – Лістинг компоненту Input

Для збереження змінних електронної пошти та паролю використаємо React-хук useState. useState - це хук в бібліотеці React, який дозволяє функціональним компонентам зберігати та оновлювати їх стан. Використовується для роботи із змінними внутрішнього стану компонентів.

Враховуючи, що авторизація буде використовуватись у багатьох сторінках застосунку, варто створити контекст для збереження змінної користувача. Контекст (Context) в React - це механізм, який дозволяє передавати дані глибоко вниз по дереву компонентів без необхідності передачі їх через кожен компонент окремо. Це особливо корисно, коли потрібно поділити однакові дані між багатьма компонентами.

Перш за все, створимо нову змінну контексту (рисунок 3.6).

```
export const UserContext = createContext();
```

Рисунок 3.6 – Лістинг коду створення контексту

Далі створимо компонент-провайдер для передачі контексту дочірнім компонентам. Для цього створимо новий компонент UserContextProvider, що

буде повертати дані контексту усім дочірнім компонентам children. Лістинг компоненту зображено на рисунку 3.7:

```
export const UserContextProvider = ({ children }) => {
  const auth = getAuth(app);

  const [user, setUser] = useState(auth.currentUser);

  useEffect(() => {
    auth.onAuthStateChanged((maybeUser) => {
      setUser(maybeUser);
    });
  }, []);

  return (
    <UserContext.Provider value={{ auth, user, setUser }}>
      {children}
    </UserContext.Provider>
  );
};
```

Рисунок 3.7 – Лістинг компоненту-провайдеру контексту

Для отримання авторизації використовується метод getAuth, що імпортується з сервісу Firebase. Далі створюється перемінна стану user для збереження в ній даних користувача. Хук useEffect в React використовується для виконання побічних ефектів у функціональних компонентах. Побічні ефекти можуть бути, наприклад, взаємодія з зовнішнім API, зміни DOM, підписка на події, тощо. useEffect дозволяє виконувати ці дії відразу після того, як компонент змінюється. При зміні стану авторизації, тобто при авторизації чи виходу з системи, стан user буде змінюватись.

Для використання сервісів Google для авторизації, необхідно створити нову змінну у файлі firebase.js, яку назвемо googleAuthProvider. Це буде новий екземпляр класу GoogleAuthProvider.

При натисканні на кнопку «Вхід за допомогою Google» необхідно використати метод signInWithPopup, що також імпортується з сервісів Firebase.

Вхід за допомогою електронної пошти та паролю відбуватиметься за допмогою методу signInWithEmailAndPassword.

Аналогічним чином створимо сторінку реєстрації. Реєстрація нового користувача відбувається за допомогою методу createUserWithEmailAndPassword.

Створені сторінки реєстрації та авторизації зображені на рисунках 3.8 та 3.9 відповідно.

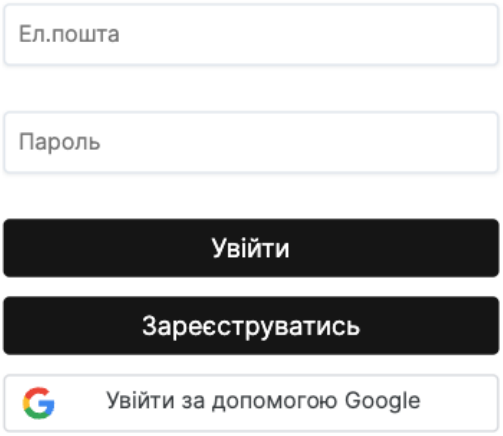

Рисунок 3.8 – Створена сторінка авторизації

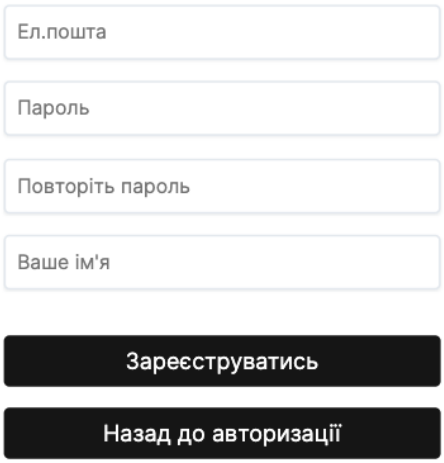

Рисунок 3.9 – Створена сторінка реєстрації

Для створення сторінки усіх проєктів, необхідний доступ до бази даних. Як визначалось у другому розділі, в якості бази даних буде використана Firestore Firebase. Для її використання, перш за все, необхідно створити нову базу даних у консолі Firebase (рисунок 3.10) та створити нову колекцію projects

(риснок 3.11). Наступним чином, необхідно ініціалізувати базу даних у файлі firebase.js за допомогою методу getFirestore.

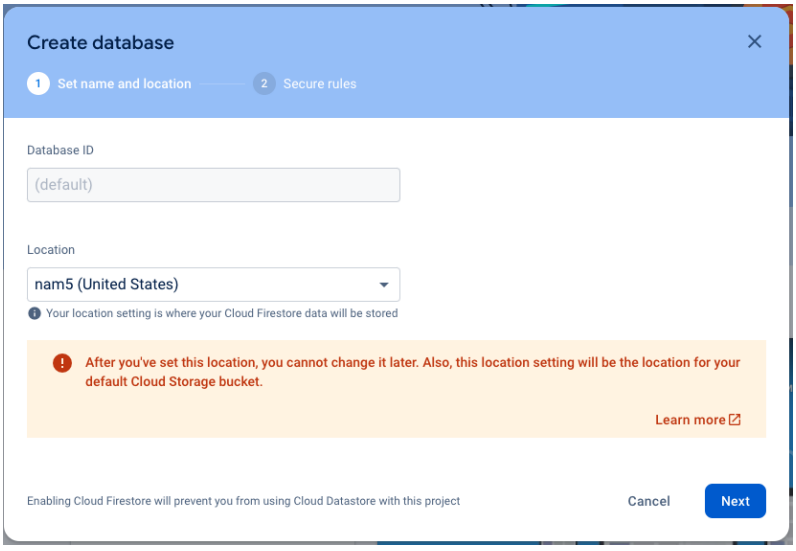

Рисунок 3.10 – Ініціалізація Firebase Firestore

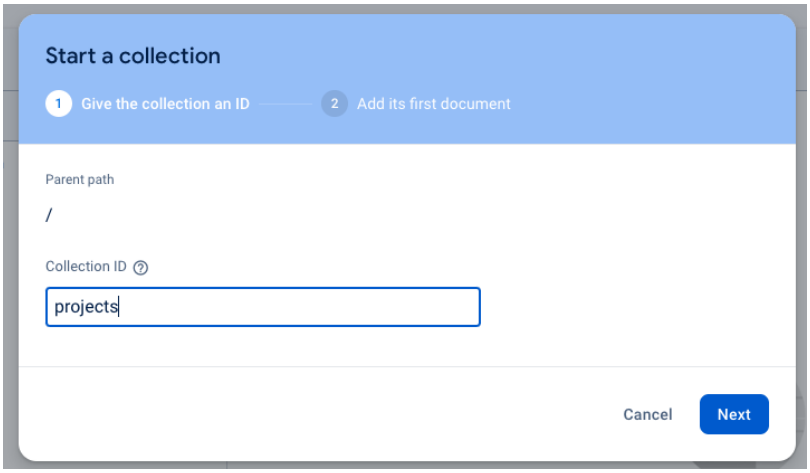

Рисунок 3.11 – Створення нової колекції

Створимо метод для доступу до колекції проєктів у базі даних. Метод прийматиме унікальний ідентифікатор користувача як параметр та повертатиме проєкти відповідно до введеного параметру ідентифікатора.

Для отримання проєктів з бази даних у застосунку, використаємо хук useEffect, в масив залежностей якого додамо дві змінні – user та inv. Таким чином, масив проєктів буде оновлюватись тоді, коли буде змінюватись користувач у системі, або програмно, коли, наприклад, користувач буде створювати новий проєкт. Лістинг коду використання хуку зображено на рисунку 3.12:

```
useEffect(() => {
  async function fetchData() {
    try {
      const docSnap = await getDoc(projectsColRef(user.uid));
      if (docSnap.exists() && docSnap.data().projects) {
        setProjects(docSnap.data().projects);
      }
    } catch (error) {
      console.log(error);
    }
  }

  if (user) {
    fetchData();
  }
}, [user, inv]);
```

Рисунок 3.12 – Лістинг використання хуку useEffect для отримання проєктів з бази даних

Для отримання усіх проєктів використовується метод getDoc, у який передається результат раніше створеного методу доступу до колекції.

Аналогічним чином, створимо решту сторінок. Для отримання задач проєкту, створимо аналогічну змінну tasksColRef, якому передамо в метод getDoc.

Решта розроблених сторінок зображені на рисунках 3.13 та 3.14.

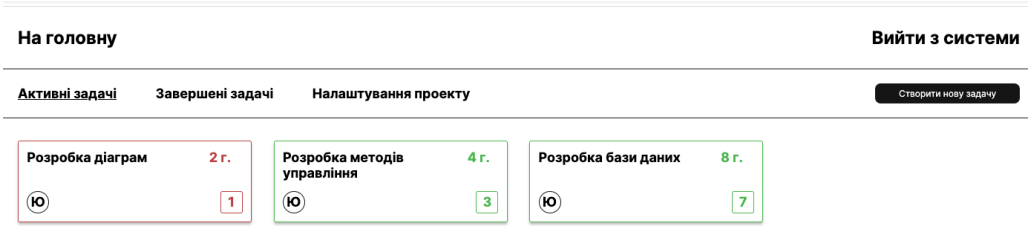

Рисунок 3.13 – Сторінка активних задач проєкту

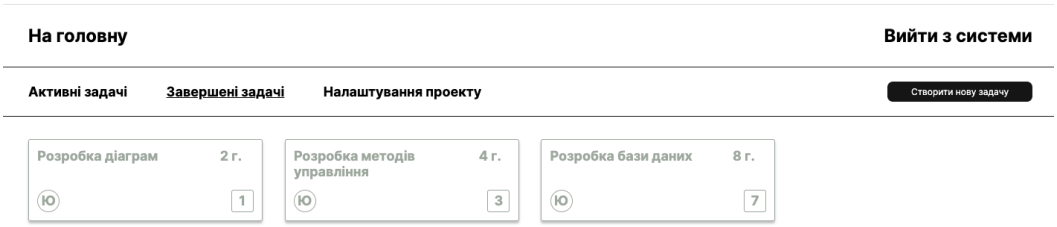

Рисунок 3.14 – Сторінка завершених задач проєкту

3.5. Розробка модуля алгоритму управління проєктами та задачами в командному середовищі

Для програмної реалізації алгоритму управління проєктами та задачами в командному середовищі з використанням технологій штучного інтелекту використаємо інструмент TensforFlow.js.

TensorFlow.js - це бібліотека для машинного навчання, яка дозволяє розробникам використовувати можливості машинного навчання та нейронних мереж у веб-браузері або на сервері за допомогою JavaScript або Node.js [36].

За допомогою TensorFlow.js можна виконувати такі завдання, як класифікація зображень, обробка природної мови, генерація тексту та багато інших. Ця бібліотека дозволяє розробникам тренувати та використовувати моделі машинного навчання безпосередньо в браузері.

TensorFlow.js має такі ключові особливості:

— Гнучкість: Здатність працювати на різних пристроях, включаючи мобільні пристрої та IoT-пристрої;

— Відкритість та розширюваність: Відкритий код і активна спільнота, яка розширює можливості бібліотеки;

— Підтримка WebGL: Використання відкритого веб-стандарту для виконання обчислень на відображенні, що прискорює роботу;

TensorFlow.js надає інструменти для розробки, навчання та використання моделей машинного навчання у веб-додатках, що робить його

потужним інструментом для реалізації проєктів з інтелектуальним аналізом даних у веб-середовищі.

Для початку використання TensforFlow варто створити нову модель. Моделі ML — це алгоритми, які приймають вхідні дані та створюють вихідні дані. При використанні нейронних мереж алгоритм являє собою набір шарів нейронів із «вагами» (числами), які керують їхнім виходом. У процесі тренування вчаться ідеальні значення для цих ваг.

Створимо функцію createModel. Лістинг функції зображено на рисунку 3.15:

```
function createModel() {
  const model = tf.sequential();

  model.add(tf.layers.dense({ inputShape: [1], units: 1, useBias: true }));
  model.add(tf.layers.dense({ units: 1, useBias: true }));

  return model;
}
```

Рисунок 3.15 – Лістинг функції створення нової моделі

Щоб отримати переваги у продуктивності TensorFlow.js, які роблять навчання моделям машинного навчання практичним, потрібно перетворити дані на тензори. Також виконаємо низку перетворень наших даних, які є найкращими методами, а саме перетасування та нормалізацію.

Спершу рандомізуємо порядок прикладів, які будуть передаватися в навчальний алгоритм. Перемішування важливе, оскільки зазвичай під час навчання набір даних розбивається на менші підмножини, які називаються пакетами, на яких навчається модель. Перетасування допомагає кожному пакету мати різноманітні дані з усього розподілу даних.

Далі створюємо два масиви: один для вхідних прикладів (пріоритетність задачі), а інший для справжніх вихідних значень (необхідний час для виконання).

Наступним чином перетворюємо дані кожного масиву в двовимірний тензор. Тензор матиме форму [num_examples, num_features_per_example]. Тут маємо приклади inputs.length, і кожен приклад має 1 вхідну функцію.

Вкінці розроблюється ще одна найкраща практика для навчання машинному навчанню - нормалізація даних. Тут нормалізуються дані в числовому діапазоні 0-1 за допомогою мінімально-максимального масштабування. Нормалізація важлива, оскільки внутрішня частина багатьох моделей машинного навчання, які будуються за допомогою tensorflow.js, розроблена для роботи з не надто великими числами. Загальні діапазони для нормалізації даних включають 0 до 1 або від -1 до 1.

Завдяки створеному екземпляру моделі та представленню даних у вигляді тензорів, можна розпочати процес навчання. Для цього створимо функцію trainModel. Ми повинні «скомпілювати» модель перед тим, як навчати її. Для цього ми маємо визначити кілька дуже важливих речей:

— Оптимізатор. Це алгоритм, що відповідає за керування оновленням моделі, враховуючи представлені приклади. У TensorFlow.js доступний ряд оптимізаторів, і ми обрали adam через його високу ефективність на практиці та відсутність потреби в складних налаштуваннях.

— Втрата. Це метрика, яка інформує модель про її ефективність у вивченні кожної з партій, або підгруп, даних, які вона використовує. У даному випадку ми використовуємо meanSquaredError для порівняння прогнозів, зроблених моделлю, із справжніми значеннями.

```
async function trainModel(model, inputs, labels) {
  model.compile({
    optimizer: tf.train.adam(),
    loss: tf.losses.meanSquaredError,
    metrics: ["mse"],
  });

  const batchSize = 32;
  const epochs = 50;

  return await model.fit(inputs, labels, {
    batchSize,
    epochs,
    shuffle: true,
  });
}
```

Рисунок 3.16 – Лістинг функції тренування моделі

Тепер, коли модель навчена, можна зробити деякі прогнози, для цього створимо функції testModel. Лістинг функції зображено на сторінці 3.17:

```
function testModel(model, inputData, normalizationData) {
  const { inputMax, inputMin, labelMin, labelMax } = normalizationData;

  const [xs, preds] = tf.tidy(() => {
    const xsNorm = tf.linspace(0, 1, 100);
    const predictions = model.predict(xsNorm.reshape([100, 1]));

    const unNormXs = xsNorm.mul(inputMax.sub(inputMin)).add(inputMin);

    const unNormPreds = predictions.mul(labelMax.sub(labelMin)).add(labelMin);

    return [unNormXs.dataSync(), unNormPreds.dataSync()];
  });

  const predictedPoints = Array.from(xs).map((val, i) => {
    return { priority: val, timeToComplete: preds[i] };
  });

  console.log(predictedPoints);
}
```

Рисунок 3.17 – Лістинг функції тестування моделі

Метод буде приймати у себе створену та навчену модель, введені користувачем дані та нормалізовані дані. Усередині функції створюється нова змінна xsNorm, у якій створюєтсья 100 нових «прикладів» для введення в модель. Model.predict — це те, ці приклади подаються в модель. Щоб повернути дані до початкового діапазону (а не 0-1), ми використовуємо значення, які ми обчислили під час нормалізації, але просто інвертуємо операції.

Наступним чином використовується метод dataSync. Це метод, який ми можемо використовувати для отримання масиву типів значень, що зберігаються в тензорі. Це дозволяє нам обробляти ці значення у звичайному JavaScript. Це синхронна версія методу .data(), якій зазвичай надають перевагу.

3.6. Висновок

Отже, у цьому розділі виконано аналіз існуючих засобів для реалізації програмного забезпечення. Розглянуті BaaS-інструменти, такі як Kinvey, AWS Amplify та Firebase. Після уважного аналізу обрано Firebase для впровадження

бекенд-частини застосунку, оскільки вона має кілька ключових переваг, включаючи широкий функціонал і безкоштовний план для початкового розвитку. Також проведено аналіз інструментів для фронтенд-реалізації застосунку, в результаті вибрано бібліотеку React для створення клієнтської частини додатку.

Далі, був проведений аналіз середовищ розробки. Оскільки вибір для реалізації клієнтської частини додатку падав на React, виникла потреба вибору основного середовища розробки серед Visual Studio Code, Webstorm та Sublime Text. Після ретельного порівняння середовищ прийнято рішення вибрати Visual Studio Code, що виявився найбільш підходящим для розробки. Його вибір обумовлений рядом переваг, таких як простий користувацький інтерфейс та широкі можливості налаштувань.

Після аналізу існуючих інструментів та середовищ розробки розпочалась реалізація програмного застосунку. Ініційовано розробку модулів авторизації та реєстрації, а також користувацького інтерфейсу та навігації. Наступним кроком було впровадження логіки для кожної окремої сторінки.

Після створення користувацького інтерфейсу було здійснено програмну реалізацію алгоритму управління проєктами та задачами в командному середовищі з використанням технологій штучного інтелекту. Для реалізації алгоритму використовувалась бібліотека для машинного навчання TensorFlow.

Таким чином було здійснено програмну реалізацію методів та засобів управління проєктами та задачами в командному середовищі у вигляді веб-застосунку. Після завершення цього етапу необхідно приступити до тестування створеного застосунку.

4. ТЕСТУВАННЯ РОБОТИ ПРОГРАМНОГО ЗАСОБУ

4.1. Аналіз методів тестування програмного забезпечення

Тестування програмного забезпечення є важливою складовою процесу розробки і має кілька ключових переваг. По-перше, воно допомагає виявити та виправити помилки та невідповідності вимогам на ранніх етапах розробки, що значно зменшує витрати на виправлення проблем у майбутньому. Тестування також підвищує ефективність розробки, оскільки автоматизовані тести дозволяють швидше виконувати тести та прискорюють випуск нових функцій.

Тестування сприяє ефективному використанню коду через автоматизацію повторюваних завдань та забезпечення надійної документації. Це полегшує підтримку та розширення системи в майбутньому. Крім того, впевненість у якості продукту через тестування допомагає підвищити довіру користувачів та клієнтів.

Існує кілька методик тестування веб-застосунків, які можна використовувати в різних етапах розробки:

— Модульне тестування, зосереджене на перевірці окремих модулів чи функцій програмного коду на правильність виконання завдань.

— Інтеграційне тестування, перевіряє взаємодію між різними модулями системи, переконуючись, що вони працюють коректно разом;

— Функціональне тестування - перевірка функціональності веб-застосунку, забезпечення того, що всі функції виконуються згідно специфікацій та вимог.

— Тестування продуктивності - визначення та аналіз продуктивності веб-застосунку, включаючи швидкість завантаження сторінок, відповідь сервера, оптимізацію ресурсів і т. д.

— Тестування сумісності - перевірка, як веб-застосунок веде себе на різних браузерах, операційних системах та пристроях.

— End-to-End (E2E) тестування - метод тестування програмного забезпечення, який використовується для перевірки функціональності відповідно до вимог користувача на всіх рівнях системи.

Зважаючи на складні бізнес-процеси у додатку, найоптимальнішим варіантом буде використання e2e тестування. Основні переваги E2E тестування включають:

— Повнота тестування: E2E тести покривають весь стек програмного забезпечення від фронтенду до бекенду, включаючи бази даних та зовнішні сервіси. Це дозволяє виявити проблеми, які можуть виникнути на будь-якому рівні системи.

— Визначення реальних проблем: Тести розробляються так, щоб відображати реальні сценарії взаємодії користувача, що допомагає виявити проблеми, які можуть виникнути під час реального використання продукту.

— Виявлення інтеграційних помилок: E2E тестування вирізняється своєю здатністю виявляти проблеми, які можуть виникнути при взаємодії різних компонентів системи.

— Підтримка автоматизації: Використання інструментів автоматизації дозволяє легко виконувати E2E тести в різних середовищах та конфігураціях.

— Покращення якості продукту: Завдяки широкому покриттю, E2E тести допомагають покращити якість продукту та забезпечити високий рівень надійності.

Для написання e2e-тестів використаємо інструмент Cypress [37]. Cypress - це потужний інструмент для автоматизованого тестування веб-додатків. Основні особливості включають в себе його легкість встановлення та використання, інтерактивну консоль для взаємодії з тестами, можливість використання синхронного коду для більшої читабельності, зручний дебаггінг завдяки вбудованій консолі та візуалізації тестів.

Один з ключових аспектів - це підтримка різних браузерів, таких як Chrome та Firefox, а також можливість виконувати тести паралельно. Важливою перевагою є підтримка Live Reload, що дозволяє переглядати результати тестів в режимі реального часу.

Cypress дозволяє легко мокувати HTTP-запити, спрощуючи тестування різних сценаріїв взаємодії з сервером. Його синтаксис простий і легко зрозуміти, а можливість використання інтерактивної консолі спрощує дебаггінг і розробку тестів.

4.2. Тестування системи управління проєктами та задачами в командному середовищі

Для того, щоб додати Cypress у веб-застосунок, необхідно виконати кілька кроків. Перш за все, необхідно встановити залежність Cypress за допомогою команди npm install --save-dev cypress. Далі необхідно запустити Cypress за допомогою команди npx cypress open.

Створимо перший автоматизований тест авторизації у застосунку. Лістинг функції тестування зображено на рисунку 4.1:

```
describe("Authorize", () => {
  before(() => {
    cy.visit("http://localhost:3000/");
    cy.contains("Вийти з системи").should("not.exist");
  });

  it("authorizes correctly", () => {
    cy.get('[data-cy="email"]').clear().type("test123@gmail.com");
    cy.get('[data-cy="password"]').clear().type("12345678");

    cy.get('[data-cy="submit"]').click();

    cy.wait(3000);

    cy.contains("Вийти з системи").should("be.visible");
  });
});
```

Рисунок 4.1 – Лістинг функції тестування авторизації

Для виконання підготовчих дій перед запуском тесту використовується метод before. У ньому використовується командна cy.visit, яка відкриває застосунок та перевіряє, чи не присутня на сторінці кнопки виходу з

застосунку, тобто чи користувач не авторизований. Далі вводимо необхідні дані для входу та натискаємо кнопку у війти, після чого, при правильний авторизації, повинна з’явитись навігацію по застосунку.

Результат роботи тесту зображено на рисунку 4.2:

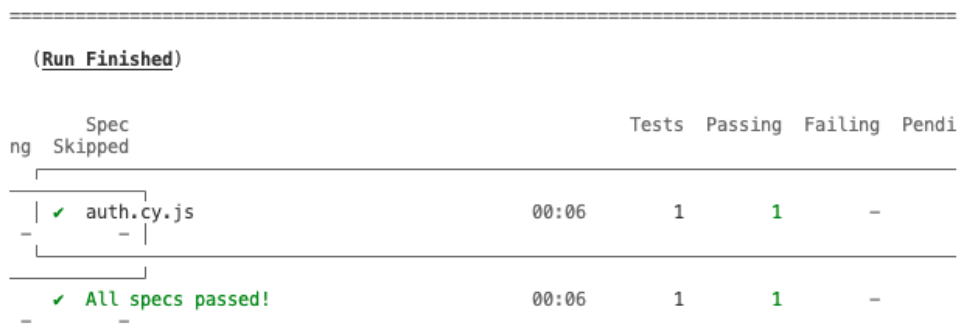

Рисунок 4.2 – Результат роботи тестування авторизації

Аналогічним чином створимо тест реєстрації в застосунку. Лістинг функції тестування реєстрації зображено на рисунку 4.3:

```
function getRandomInt(max) {
  return Math.floor(Math.random() * max);
}

describe("Register", () => {
  before(() => {
    cy.visit("http://localhost:3000/register");
    cy.contains("Вийти з системи").should("not.exist");
  });

  it("registers correctly", () => {
    cy.get('[data-cy="email"]')
      .clear()
      .type(`test${getRandomInt(99999)}@gmail.com`);
    cy.get('[data-cy="password"]').clear().type("12345678");
    cy.get('[data-cy="passwordRepeat"]').clear().type("12345678");
    cy.get('[data-cy="name"]').clear().type("Test");

    cy.get('[data-cy="submit"]').click();

    cy.wait(3000);

    cy.contains("Вийти з системи").should("be.visible");
  });
});
```

Рисунок 4.3 – Лістинг функції тестування реєстрації

Схожим чином, перед початком тесту, виконується вхід у застосунок, на сторінку реєстрації. Після цього, вводяться дані нового «користувача», а саме рандомізована електронна пошта, паролі та ім’я, потім натискається кнопка

реєстрації. Далі проводиться перевірка, чи з'явилась навігація застосунку, тобто чи новий користувач автоматично авторизувався після реєстрації. Результат роботи тесту зображено на рисунку 4.4:

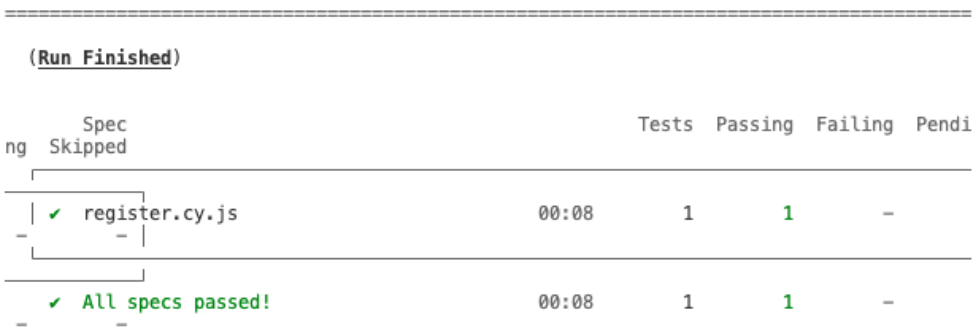

Рисунок 4.4 – Результат роботи тестування реєстрації

Нарешті, створимо тест по створенню нового проєкту. Лістинг створеної функції зображено на рисунку 4.5:

```
describe("Create Project", () => {
  before(() => {
    cy.visit("http://localhost:3000/create-project");
    cy.wait(3000);
  });

  it("creates project", () => {
    cy.get('[data-cy="name"]').clear().type(`newProjectName`);

    cy.get('[data-cy="submit"]').click();

    cy.wait(3000);

    cy.contains("newProjectName").should("be.visible");
  });
});
```

Рисунок 4.5 – Результат роботи тестування створення нового проєкту

Результати роботи застосунку зображено на рисунку 4.6:

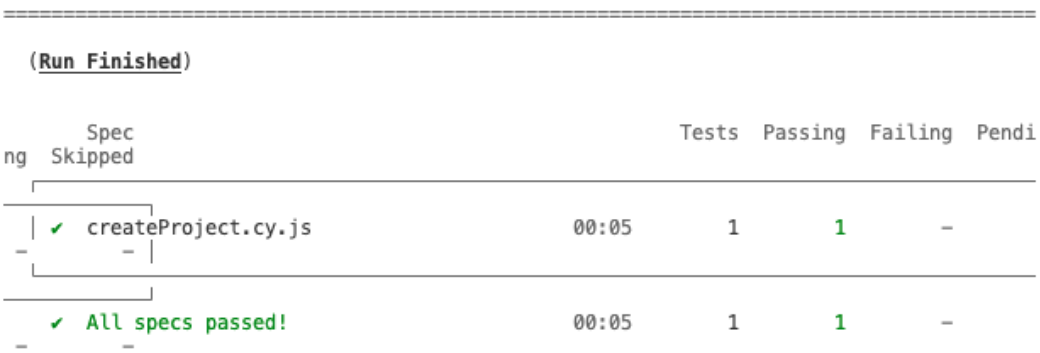

Рисунок 4.6 – Результат роботи тестування створення проєкту

4.3. Тестування роботи алгоритму управління проєктами та задачами в командному середовищі

Для тестування роботи алгоритму управління проєктами та задачами в командному середовищі створимо нову змінну data, що буде містити у собі дані для тренування нейронної мережі. Лістинг коду змінної зображено на рисунку 4.7:

```
export const data = [
  { priority: 1, timeToComplete: 5 },
  { priority: 3, timeToComplete: 10 },
  { priority: 2, timeToComplete: 7 },
  { priority: 5, timeToComplete: 12 },
  { priority: 7, timeToComplete: 10 },
  { priority: 9, timeToComplete: 9 },
  { priority: 3, timeToComplete: 5 },
  { priority: 4, timeToComplete: 2 },
  { priority: 5, timeToComplete: 7 },
  { priority: 6, timeToComplete: 5 },
];
```

Рисунок 4.7 – Лістинг змінної з даними для тренування

Наступним чином, необхідно натренувати модель використовуючи ці дані. Для цього створимо нову функцію, лістинг якої зображено на рисунку 4.8:

```
export const run = async () => {
  const model = createModel();
  const tensorData = convertToTensor(data);
  const { inputs, labels } = tensorData;
  await trainModel(model, inputs, labels);
  testModel(model, data, tensorData);
};
```

Рисунок 4.8 – Лістинг функції тренування моделі тестовими даними

Спочатку створюється нова модель. Далі створені дані переворюються у тензори, після чого створена модель тренується на створених даних. Результат тестування алгоритму зображено на рисунку 4.9:

Рисунок 4.9 – Результат тестування алгоритму управління проєктами та задачами в командному середовищі

Першим значенням у даному масив є ймовірна пріоритетність задачі, друге значення – час, необхідний на виконання задачі на основі переданих даних.

4.4. Висновок

Після проведення тестування можна прийти до висновку, що головна мета проєкту успішно досягнута. Розроблений застосунок дозволяє користувачам ефективно управляти своїми проєктами, створювати нові та приєднуватися до існуючих. Програмне забезпечення працює стабільно та виконує свої функції відповідно до зазначених функціональних та нефункціональних вимог.

5. ЕКОНОМІЧНА ЧАСТИНА

5.1. Оцінювання комерційного потенціалу розробки

Метою проведення комерційного та технологічного аудиту є оцінювання комерційного потенціалу розробки методів і програмних засобів управління проєктами та задачами в командному середовищі.

Для проведення технологічного аудиту було залучено 3-х незалежних експертів з Вінницького національного технічного університету: Ракитянська Ганна Борисівна (к.т.н., доц. кафедри ПЗ ВНТУ), Нестерук Віталій Андрійович (студент кафедри ПЗ ВНТУ), Поліщук Микола Олегович (студент кафедри ПЗ ВНТУ). Для проведення технологічного аудиту було використано таблицю 5.1 в якій за п’ятибальною шкалою використовуючи 12 критеріїв здійснено оцінку комерційного потенціалу.

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
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 на великому ринку
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 ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
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	Експлуатаційні витрати значно нижчі, ніж в аналогів

Продовження таблиці 5.1

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 позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів 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 реалізації ідеї
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 фінансуван ня 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 давно використовуються у виробництві
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 окупності інвестицій менше 3-х років
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 пові- домлення відповідним органам про виро- бництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Таблиця 5.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

В таблиці 5.3 наведено результати оцінювання експертами комерційного потенціалу розробки:

Таблиця 5.3 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Ракитянська Г.Б.	Нестерук В.А.	Поліщук М.О.
	Бали, виставлені експертами:		
1	4	4	3
2	3	3	3
3	3	4	4
4	3	3	3
5	4	4	4
6	3	3	3
7	3	3	3
8	4	3	4
9	3	3	3
10	3	3	3
11	4	4	4
12	3	3	3
Сума балів	$СБ_1$=40	$СБ_2$=40	$СБ_3$=40
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{40+40+40}{3} = 40$		

Середньоарифметична сума балів, розрахована на основі висновків експертів склала 40 балів, що згідно таблиці 5.2 вважається, що рівень комерційного потенціалу проведених досліджень є вищим середнього.

Методи і програмі засоби управління проєктами та задачами в командному середовищі, що розробляються в рамках магістерської кваліфікаційної роботи будуть цікаві інвесторам як потенційно прибутковий стартап.

Порівняємо нову розробку, що розробляється в магістерській кваліфікаційній роботі з аналогом, який існує на ринку. Аналогом у даному випадку було обрано додаток Jira.

Проведемо оцінку якості і конкурентоспроможності нової розробки порівняно з аналогом. В таблиці 5.4 наведені основні техніко-економічні показники аналога і нової розробки.

Таблиця 5.4 – Основні параметри нової розробки та товару-конкурента

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
Швидкість завантаження проєкту	5	10	2	40%
Швидкість створення задачі	4	4	1	30%
Використання ресурсів Інтернет-з’єднання, %	60	20	3	15%
Використання ресурсів пристрою, %	30	10	3	15%

Проведемо оцінку якості продукції, яка є найефективнішим засобом забезпечення вимог споживачів та порівняємо її з аналогом.

Визначимо відносні одиничні показники якості по кожному параметру за формулами (5.1) та (5.2) і занесемо їх у відповідну колонку табл. 5.5.

$$q_i = \frac{P_{\text{Б}_\text{i}}}{P_{\text{Н}_\text{i}}} \tag{5.1}$$

або

$$q_i = \frac{P_{\text{Н}_\text{i}}}{P_{\text{Б}_\text{i}}} \tag{5.2}$$

де $\text{Р}_{\text{Н}_\text{i}}$, $\text{Р}_{\text{Б}_\text{i}}$ – числові значення *i*-го параметру відповідно нового і базового виробів.

$$q_1 = \frac{10}{5} = 2$$

$$q_2 = \frac{4}{4} = 1$$

$$q_3 = \frac{60}{20} = 3$$

$$q_4 = \frac{30}{10} = 3$$

Відносний рівень якості нової розробки визначаємо за формулою:

$$\text{К}_{\text{я.в.}} = \sum_{i=1}^{n} q_i \cdot \alpha_i \tag{5.3}$$

$$\text{К}_{\text{я.в.}} = 2 \cdot 0{,}4 + 1 \cdot 0{,}3 + 3 \cdot 0{,}15 + 3 \cdot 0{,}15 = 2$$

Загальний показник конкурентоспроможності інноваційного рішення (К) з урахуванням вищезазначених груп показників можна визначити за формулою:

$$\text{К} = \frac{\text{I}_{\text{т.п.}}}{\text{I}_{\text{е.п.}}} \tag{5.4}$$

де $I_{т.п.}$ – індекст технічних параметрів; $I_{е.п.}$ – індекст економічних параметрів.

Індекс технічних параметрів є відносним рівнем якості інноваційного рішення. Індекс економічних параметрів визначається за формулою (5.5)

$$I_{е.п.} = \frac{\sum_{i=1}^{n} P_{Нei}}{\sum_{i=1}^{n} P_{Бei}} \quad (5.5)$$

де $P_{Нei}$, $P_{Бei}$ – економічні параметри (ціна придбання та споживання товару) відповідно нового та базового товарів.

$$I_{е.п.} = \frac{10}{9,6} = 0,96$$

$$К = \frac{2}{0,96} = 2,08$$

Зважаючи на розрахунки, можна зробити висновок, що нова розробка буде більш конкурентоспроможною, ніж конкурентний товар.

5.2. Прогнозування витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи групуються за такими статтями: витрати на оплату праці, витрати на соціальні заходи, матеріали, паливо та енергія для науково-виробничих цілей, витрати на службові відрядження, програмне забезпечення для наукових робіт, інші витрати, накладні витрати.

1. Основна заробітна плата кожного із дослідників ЗО , якщо вони працюють в наукових установах бюджетної сфери визначається за формулою:

$$З_О = \frac{М}{Т_Р} \cdot t \text{ (грн)} \quad (5.6)$$

де М – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

$Т_Р$ – число робочих днів в місяці; приблизно $Т_Р \approx 21 \ldots 23$ дні;

t – число робочих днів роботи дослідника.

$$З_0 = \frac{16000}{22} \cdot 66 = 48000 \text{ (грн)}$$

$$З_0 = \frac{13000}{22} \cdot 66 = 39000 \text{ (грн)}$$

Зведемо сумарні розрахунки до таблиці 5.5.

Таблиця 5.5 – Заробітна плата дослідника в науковій установі бюджетної сфери

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату грн.
Керівник	16000	727,2	66	48000
Програміст	13000	590,9	66	39000
Всього				87000

2. Розрахунок додаткової заробтіної плати робітників

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які приймали участь в розробці нового технічного рішення розраховується як 10 - 12 % від основної заробітної плати робітників.

На даному підприємстві додаткова заробітна плата начисляється в розмірі 11% від основної заробітної плати.

$$З_д = (З_0 + З_р) \cdot \frac{Н_{дод}}{100\%} \quad (5.7)$$

$$З_д = 0{,}11 \cdot 87000 = 9570 \text{ (грн)}$$

3. Нарахування на заробітну плату $Н_{ЗП}$ дослідників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою (5.8):

$$Н_{ЗП} = (З_0 + З_д) \cdot \frac{\beta}{100\%} \quad (5.8)$$

де $З_0$ – основна заробітна плата розробників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

β – ставка єдиного внеску на загальнобов'язкове державне соціальне страхування, %;

Дана діяльність відноситься до бюджетної сфери, тому ставка єдиного внеску на загальнообов'язкове державне соціальне страхування буде складати 22%, тоді:

$$Н_{ЗП} = (87000 + 9570) \cdot \frac{22}{100} = 21245{,}4 \text{ (грн)}$$

4. Витрати на матеріали М та комплектуючі К, що були використані під час виконання даного етапу роботи, розраховуються по кожному виду матеріалів за формулою:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \qquad (5.9)$$

де $Н_i$ – витрати матеріалу *i*-го найменування, кг;

$Ц_i$ – вартість матеріалу і-го найменування, грн./кг.;

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

$В_i$ – маса відходів матеріалу і-го найменування, кг;

$Ц_в$ – ціна відходів матеріалу і-го найменування, грн/кг;

n – кількість видів матеріалів.

Таблиця 5.6 – Матеріали, що використані на розробку

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено	Вартість витраченого матеріалу, грн
Папір	200	1	200
Ручка	10	1	10
Флешка	250	1	250
Всього			460
З урахування коефіцієнта транспортування			500

5. Програмне забезпечення для наукової роботи включає витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення необхідного для проведення дослідження. Для нової розробки використовувались безкоштовні програмні засоби, такі як Figma для розробки дизайн-макету застосунку, Visual Studio Code як програмне середовище розробки, Cypress для написання автоматичних тестів для створеного застосунку.

6. Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час виконання даного етапу роботи

Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

$$А = \frac{Ц \cdot Т}{Т_{кор} \cdot 12} \text{ [грн]} \quad (5.10)$$

де Ц – балансова вартість даного виду обладнання (приміщень), грн.;

$Т_{кор}$ – час користування;

Т – термін використання обладнання (приміщень), цілі місяці.

Згідно пункту 137.3.3 Податкового кодексу амортизація нараховується на основні засоби вартістю понад 25000 грн. В нашому випадку для написання магістерської роботи використовувався ноутбук вартістю 33000 грн.

$$А = \frac{33000 \cdot 3}{3 \cdot 12} = 2750$$

7. До статті «Паливо та енергія для науково-виробничих цілей» відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень.

$$В_е = \sum_{i=1}^{n} \frac{w_{yt} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_і} \quad (5.11)$$

де W_{yt} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1кВт-години електроенергії, грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

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

Для написання магістерської роботи використовувався ноутбук для якого розрахуємо витрати на електроенергію.

$$В_е = \frac{0{,}03 \cdot 510 \cdot 7{,}5 \cdot 0{,}5}{0{,}85} = 67{,}5$$

Витрати на службові відрядження, витрати на роботи, які виконують сторонні підприємства, установи, організації та інші витрати в нашому досліджені не враховуються оскільки їх не було.

Накладні (загальновиробничі) витрати $В_{нзв}$ охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Накладні (загальновиробничі) витрати $В_{нзв}$ можна прийняти як (100...150)% від суми основної заробітної плати розробників та робітників, які виконували дану МКНР, тобто:

$$В_{нзв} = (З_О + З_р) \cdot \frac{Н_{нзв}}{100\%} \qquad (5.12)$$

де $Н_{нзв}$ – норма нарахування за статтею «Інші витрати».

$$В_{нзв} = 87000 \cdot \frac{100}{100\%} = 87000 \text{ грн}$$

Сума всіх попередніх статей витрат дає витрати, які безпосередньо стосуються даного розділу МКНР:

$$В = 87000 + 9570 + 21245{,}4 + 500 + 2750 + 67{,}5 + 87000 = 208132{,}9$$

Прогнозування загальних втрат ЗВ на виконання та впровадження результатів виконаної МКНР здійснюється за формулою:

$$ЗВ = \frac{В}{\eta'} \quad (5.13)$$

де η – коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки, робота знаходиться на стадії науково-дослідних робіт, то коефіцієнт β = 0,9.

$$ЗВ = \frac{208132,9}{0,9} = 231258,8 \text{ (грн)}$$

5.3. Прогнозування комерційних ефектів від реалізації результатів розробки

У даному підрозділі кількісно спрогнозуємо, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. Розрахуємо збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, за формулою

$$\Delta П_i = \sum_1^n (\Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\upsilon}{100}\right) \quad (5.14)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році.

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт λ = 0,8333.

ρ – коефіцієнт, який враховує рентабельність продукту. р = 0,25;

υ – ставка податку на прибуток. У 2023 році – 18%.

Припустимо, що при впроваджені результатів наукової розробки покращується якість програмного продукту. Припустимо, що дохід від одного користувача зросте на 250 грн. Кількість одиниць реалізованої продукції (в даному випадку кількість користувачів) також збільшиться: протягом першого року на 1500, протягом другого року – на 2500, протягом третього року на 5000. Реалізація продукції до впровадження розробки складала 1000 шт., а дохід від одного користувача складає 1000 грн. Розрахуємо прибуток, яке отримає підприємство протягом трьох років.

$$\Delta П_1 = [250 \cdot 1000 + (1000 + 250) \cdot 1500] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 - \frac{18}{100}\right) = 362875{,}6 \text{ грн}$$

$$\Delta П_2 = [250 \cdot 1000 + (1000 + 250) \cdot 4000] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 - \frac{18}{100}\right) = 896516{,}2 \text{ грн}$$

$$\Delta П_3 = [250 \cdot 1000 + (1000 + 250) \cdot 9000] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 - \frac{18}{100}\right) = 1963797{,}5 \text{ грн}$$

Таким чином, розрахунки показують, що відповідно прогнозуванню комерційний ефект від впровадження розробки виражається у значному збільшенні чистого прибутку підприємства.

5.4. Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахуємо основні показники, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахуємо величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки.

$$PV = k_{\text{інв}} \cdot \text{ЗВ} \quad (5.15)$$

$k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження наково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщен, розробку технологій, навчання персоналу, маркетингові заходи тощо ($k_{інв}$ = 2...5).

$$PV = 2 \cdot 231258{,}8 = 462517{,}6$$

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$ згідно наступної формули:

$$\text{Е}_{\text{абс}} = (\text{ПП} - PV) \quad (5.16)$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової розробки, грн.;

$$\text{ПП} = \sum_{1}^{T} \frac{\Delta\text{П}_\text{i}}{(1+\tau)^t} \quad (5.17)$$

де $\Delta\text{П}_\text{i}$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

Т – період часу, протягом якою виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2;

t – період часу (в роках).

$$\text{ПП} = \frac{362875{,}6}{(1+0{,}2)^1} + \frac{896516{,}2}{(1+0{,}2)^2} + \frac{1963797{,}5}{(1+0{,}2)^3}$$

$$= \frac{362875{,}6}{1{,}2} + \frac{896516{,}2}{1{,}44} + \frac{1963797{,}5}{1{,}728}$$

$$= 302396{,}3 + 622580{,}7 + 1136456{,}9 = 2061433{,}90 \text{ (грн)}$$

$$E_{абс} = (2061433{,}90 - 462517{,}6) = 1598916{,}3 \text{ (грн)}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього користуються формулою:

$$E_в = \sqrt[Т_ж]{1 + \frac{E_{абс}}{PV}} - 1 \qquad (5.18)$$

де $Т_ж$ – життєвий цикл науокової розробки, роки

$$E_в = \sqrt[3]{1 + \frac{1598916{,}3}{462517{,}6}} - 1 = \sqrt[3]{4{,}45} - 1 = 0{,}64$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f \qquad (5.19)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d = 0,16

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau_{min} = 0{,}16 + 0{,}05 = 0{,}21$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної науокові розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проєкту інвестицій за формулою:

$$Т_{ок} = \frac{1}{E_в} \qquad (5.20)$$

$$T_{ок} = \frac{1}{0{,}64} = 1{,}56 \text{ роки}$$

Так як $T_{ок} \leq 3..5$-ти років, то фінансування даної наукової розробки є доцільним.

5.5. Висновок

В даному розділі було здійснено оцінювання комерційного потенціалу розробки методів і програмних засобів управління проєктами та задачами в командному середовищі у вигляді веб-застосунку.

Проведено технологічний аудит з залученням трьох експертів. Аналіз експертних даних показав, що рівень комерційного потенціалу розробки вищий середнього. Дослідження комерційного потенціалу розробки показав, що програмний продукт за своїми характеристиками випереджає аналогічні програмні продукти і є перспективною розробкою. Він має кращі функціональні показники, а тому є конкурентоспроможним товаром на ринку.

Прогнозування витрат на виконання науково-дослідної роботи по кожній з статей витрат складає 208132,9 грн. Загальна ж величина витрат на виконання та впровадження результатів даної НДР буде складати 231258,8 грн.

Вкладені інвестиції в даний проєкт окупляться через 18 з половиною місяців при прогнозованому прибутку 2061433,90 грн. за три роки.

ВИСНОВКИ

У магістерській кваліфікаційній роботі проведено розробку методів і програмних засобів управління проєктами та задачами в командному середовищі. У результаті проведення роботи було створено веб-застосунок за допомогою JavaScript-бібліотеки React та BaaS-сервісу Firebase. Відмінністю розробленого застосунку від відомих моделей управління проєктами є покращення процесу прийняття рішень та зменшення ризиків шляхом використання штучного інтелекту.

У результаті постановки задачі було виявлено, що в сучасній сфері управління проєктами та задачами існує потреба у створенні нового методу, який допоможе пристосуватись до зростаючої складності проєктів, зростаючого обсягу інформації та різноманітних завдань.

Проведено аналіз існуючих методів управління проєктами та задачами. Був проведений ретельний аналіз існуючих методів вирішення конкретної задачі призвів до визначення оптимального напрямку. Обрана технологія адаптивного управлінського циклу, що ґрунтується на використанні методу самоорганізації, визначена як ключова для розробки високоефективної системи управління проєктами та завданнями в командному середовищі.

Для ефективної розробки програмного застосунку був проведений докладний порівняльний аналіз існуючих аналогів на ринку. Було розглянуто такі програмні засоби як Microsoft Project, Trello, Asana та Jira. В процесі здійснення порівняльного аналізу було визначено слабкі та сильні сторони аналогів та визначено основні задачі розроблюваного програмного засобу та було прийнято рішення розроблювати програмний засіб саме як веб-застосунок. Було визначені функціональні та нефункціональні вимоги застосунку.

В процесі проектування було розроблено метод управління проєктами та задачами в командному середовищі на основні аналізу даних про пріоритет задач і час на її виконання. Після цього, було розроблено архітектуру

розроблюваного програмного засобу, а саме для реалізації застосунку була обрана безсерверна архітектура з використанням BaaS-сервісів для виконання функції серверної частини застосунку. Для розуміння можливості дій користувача було розроблено діаграму використання застосунку. Для успішної реалізації та гарного користувацького досвіду була розроблена логічна структура застосунку, для організації програмного коду – фізична структура. Для розуміння організації даних у застосунку була спроектована база даних. Як кінцевий етап проектування був розроблений прототип-дизайн застосунку.

Перед початком реалізації програмного засобу був проведений аналіз найвідоміших засобів реалізації програмного продукту. Були проаналізовані такі BaaS-сервіси як Kinvey, AWS Amplify та Firebase. Враховуючи специфіку проєкту та технічні вимоги проєкту був обраний Firebase. Для реалізації клієнтської сторони застосунку були проаналізовані такі відомі JavaScript-фреймворки як Angular, Vue та React, після врахування переваг та недоліків кожного з яких було обрано React.

Обираючи середовище розробки для даної роботи, було виділено кілька ключових аспектів, щоб гарантувати оптимальні умови для реалізації поставлених завдань. Після ретельного аналізу і порівняння трьох популярних середовищ розробки, а саме Sublime Text, WebStorm та Visual Studio Code, було визначено, що Visual Studio Code відповідає вимогам роботи найкраще.

У ході розробки програмного забезпечення було реалізовано обраний функціонал. Була розроблена не лише візуальна, але й логічна частини усіх екранів, забезпечуючи користувачам зручний та інтуїтивно зрозумілий інтерфейс.

Одним з ключових етапів була програмна реалізація розробленого методу управління проєктами та задачами в командному середовищі. Для цього був обраний інструмент TensorFlow, що дозволив використовувати потужність штучного інтелекту та машинного навчання для оптимізації

управлінських процесів. Використання TensorFlow стало ключовим компонентом у розробці продукту, дозволяючи вдосконалити методи управління проєктами, забезпечуючи швидкі та точні рішення в командному взаємодії.

Такий підхід дозволяє не лише ефективно використовувати ресурси та підвищувати продуктивність, але і вносити інновації в область управління проєктами, використовуючи передові технології для покращення робочих процесів та досягнення оптимальних результатів.

Після здійснення розробки програмного забезпечення були створені автоматизовані тести за допомогою фреймоврку Cypress. Були створені тест-кейси для усіх можливих випадків, і створено відповідні шаблони тестування. Окремо було протестовано програмно розроблений метод управління проєктами та задачами. Створені тести забезпечать стабільність роботи програмного застосунку після внесення додаткових змін у програмний код.

Розрахунки економічних витрат і прибутків, визначення термінів окупності та оцінка економічного ефекту підтвердили, що в сучасних умовах розробка виявляється конкурентоспроможною на ринку інформаційних технологій.

В даній роботі важливу практичну цінність представляють розроблені модулі управління проєктами та задачами, які можуть бути використані не лише у даному проєкті, а й в інших подібних ініціативах. Крім того, варто відзначити, що цей розвинутий функціонал надає роботі високий рівень конкурентоспроможності на ринку. Аналіз показав, що у цій області маємо потребу у розробці нового методу та програмних засобів, а розроблений продукт вирізняється численними суттєвими перевагами. Такий підхід дозволяє зазначити, що дана робота вирізняється та володіє великим потенціалом на ринку аналогічних продуктів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Корольчук Ю. О. Розробка методів і програмних засобів управління проєктами та задачами в командномоу середовищі // Ю. О. Корольчук., – Збірник матеріалів Міжнародної науково-практичної Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ» – Вінниця ВНТУ, Вінницька академія неперервної освіти, 2023 – С. 130.

2. O'Connell F. The Project Management Book. Лондон : LID Publishing, 2018. 128 с.

3. Horine G. Project Management Absolute Beginner's Guide. 4th ed. Індіаполіс : Que Publishing, 2017. 448 с.

4. Portny S. E. Project Management for dummies. Hoboken : Wiley-Blackwell (an imprint of John Wiley & Sons Ltd), 2017. 464 с.

5. Success Rate Rise. Transforming the high cost of low performance [Електронний ресурс] – Режим доступу: https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017.pdf.

6. Microsoft Project [Електронний ресурс] – Режим доступу: https://www.microsoft.com/uk-ua/microsoft-365/project/project-management-software.

7. Trello [Електронний ресурс] – Режим доступу: https://trello.com/uk.

8. Asana [Електронний ресурс] – Режим доступу: https://asana.com/.

9. Jira Core [Електронний ресурс] – Режим доступу: https://www.atlassian.com/software/jira/work-management.

10. Project Management Methodologies, Governance and Success. Auerbach Publications, 2023. 288 с.

11. Rubin K. Essential Scrum: A Practical Guide to the Most Popular Agile Process. Addison-Wesley Signature Series (Cohn), 2012. 496 с.

12. Martin R. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Pearson, 2017. 432 с.

13. Bass L. Software Architecture in Practice. 3rd ed. Addison-Wesley Professional, 2015. 624 с.

14. Zambrano B. Serverless Design Patterns and Best Practices. Packt Publishing, 2018. 260 с.

15. Newman S. Building Microservices: Designing Fine-Grained Systems. O'Reilly, 2015. 280 с.

16. Si Alhir S. Learning UML. O'Reilly Media, 2003. 3004 p.

17. Вхід через Google [Електронний ресурс] – Режим доступу: https://support.google.com/accounts/answer/12849458?hl=uk.

18. Ater T. Building Progressive Web Apps: Bringing the Power of Native to the Browser. O'Reilly, 2017. 288 с.

19. Shivakumar S. K. Modern Web Performance Optimization: Methods, Tools, and Patterns to Speed Up Digital Platforms. 2020. 464 с.

20. Folder Structure for Modern Web Applications [Електронний ресурс] – Режим доступу: https://dev.to/noruwa/folder-structure-for-modern-web-applications-4d11.

21. Firebase Firestore [Електронний ресурс] – Режим доступу: https://firebase.google.com/docs/firestore.

22. Figma [Електронний ресурс] - Режим доступу: https://www.figma.com/.

23. Purewal S. Learning Web App Development. O'Reilly, 2013. 306 с.

24. Overview of Kinvey [Електронний ресурс] – Режим доступу: https://devcenter.kinvey.com/android/guides/core-overview.

25. AWS Amplify [Електронний ресурс] – Режим доступу: https://aws.amazon.com/amplify/.

26. Yahiaoui H. Firebase Cookbook: Over 70 recipes to help you create real-time web and mobile applications with Firebase. Packt Publishing, 2017. 373 с.

27. Bampakos A. Learning Angular: A no-nonsense beginner's guide to building web applications with Angular 10 and TypeScript. 3rd ed. Packt Publishing, 2020. 399 с.

28. Macrae C. Vue.js: Up and Running: Building Accessible and Performant Web Apps. O'Reilly, 2018. 164 с.

29. Porcello E. Learning React: Functional Web Development with React and Redux. O'Reilly, 2017. 238 с.

30. Getting Started with WebStorm [Електронний ресурс] - Режим доступу: https://www.jetbrains.com/help/webstorm/getting-started-with-webstorm.html.

31. Sublime Text [Електронний ресурс] - Режим доступу: https://www.sublimetext.com/.

32. Johnson B. Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers. Wiley, 2019. 247 с.

33. Створення нового React-додатку [Електронний ресурс] – Режим доступу: https://uk.legacy.reactjs.org/docs/create-a-new-react-app.html.

34. Create React App [Електронний ресурс] – Режим доступу: https://create-react-app.dev/.

35. React Router [Електронний ресурс] – Режим доступу: https://reactrouter.com/en/main.

36. TensorFlow.js [Електронний ресурс] – Режим доступу: https://www.tensorflow.org/js.

37. Cypress [Електронний ресурс] – Режим доступу: https://www.cypress.io/.

Додаток А

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ

д.т.н., професор Романюк О.Н.

"19" вересня 2023 р.

Технічне завдання

на магістерську кваліфікаційну роботу «Розробка методів і програмних засобів управління проєктами та задачами в командному середовищі» за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

к.т.н., доц. Г.Б.Ракитянська

"19" вересня 2023 р.

Виконав:

студент гр.1ПІ-22м Ю.О. Корольчук

"19" вересня 2023 р.

Вінниця – 2023 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методів і програмних засобів управління проєктами та задачами в командному середовищі».

Галузь застосування – веб-застосунки.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № 247 від 18 вересеня 2023 р. ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення ефективності процесу управління проєктами та задачами в командному середовищі за рахунок запропонування нового методу, що передбачає використання штучного інтелекту в даних процесах.

Призначення роботи – розробка методів і програмних засобів управління проєктами та задачами в командному середовищі.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Корольчук Ю. О. Розробка методів і програмних засобів управління проєктами та задачами в командномоу середовищі // Ю. О. Корольчук., – Збірник матеріалів Міжнародної науково-практичної Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ» – Вінниця ВНТУ, Вінницька академія неперервної освіти, 2023.

5. Технічні вимоги

— Підтримка браузерів версії Internet Explorer 11+;

— з’єднання з мережею Інтернет.

6. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Аналіз методів і програмних засобів управління проєктами та задачами в командному середовищі	20.09.2023 – 28.09.2023
2	Розробка методу та моделі системи управління проєктами та задачами в командному середовищі	29.09.2023 – 25.10.2023
3	Реалізація програмного засобу управління проєктами та задачами в командному середовищі	26.10.2023 – 10.11.2023
4	Тестування роботи програмного засобу	11.11.2023 – 19.11.2023
5	Економічна частина	20.11.2023 – 01.12.2023

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

Додаток Б

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: **Розробка методів і програмних засобів управління проєктами та задачами в командному середовищі**

Тип роботи: магістерська кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, 1ПІ – 22м

Науковий керівник: к.т.н., доц. Ракитянська Г.Б.

Unicheck	
Оригінальність	**97.7**
Схожість	2.3

Аналіз звіту подібності

■ **Запозичення, виявлені 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 автора. Роботу направити на доопрацювання.

□ 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 _____ Черноволик Г. О.

Опис прийнятого рішення: **допустити до захисту**

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Корольчук Ю.О.

Керівник роботи _____ Ракитянська Г.Б.

Додаток В

Лістинг файлу auth.cy.js

```
describe("Authorize", () => {
  before(() => {
    cy.visit("http://localhost:3000/");
    cy.contains("Вийти з системи").should("not.exist");
  });

  it("authorizes correctly", () => {
    cy.get('[data-cy="email"]').clear().type("test123@gmail.com");
    cy.get('[data-cy="password"]').clear().type("12345678");

    cy.get('[data-cy="submit"]').click();

    cy.wait(3000);

    cy.contains("Вийти з системи").should("be.visible");
  });
});
```

Лістинг файлу createProject.cy.js

```
describe("Create Project", () => {
  before(() => {
    cy.visit("http://localhost:3000/create-project");
    cy.wait(3000);
  });

  it("creates project", () => {
    cy.get('[data-cy="name"]').clear().type(`newProjectName`);

    cy.get('[data-cy="submit"]').click();

    cy.wait(3000);

    cy.contains("newProjectName").should("be.visible");
  });
});
```

Лістинг файлу register.cy.js

```
function getRandomInt(max) {
  return Math.floor(Math.random() * max);
```

```
}

describe("Register", () => {
  before(() => {
    cy.visit("http://localhost:3000/register");
    cy.contains("Вийти з системи").should("not.exist");
  });

  it("registers correctly", () => {
    cy.get('[data-cy="email"]')
      .clear()
      .type(`test${getRandomInt(99999)}@gmail.com`);
    cy.get('[data-cy="password"]').clear().type("12345678");
    cy.get('[data-cy="passwordRepeat"]').clear().type("12345678");
    cy.get('[data-cy="name"]').clear().type("Test");

    cy.get('[data-cy="submit"]').click();

    cy.wait(3000);

    cy.contains("Вийти з системи").should("be.visible");
  });
});
```

Лістинг файлу commands.js

```
import firebase from "firebase/compat/app";
import "firebase/compat/auth";
import "firebase/compat/database";
import "firebase/compat/firestore";
import { attachCustomCommands } from "cypress-firebase";

const firebaseConfig = {
  apiKey: "AIzaSyB9E1DxgjhQgmrYGE01e_zOZRIAtY4HvAk",
  authDomain: "magisters-e573b.firebaseapp.com",
  projectId: "magisters-e573b",
  storageBucket: "magisters-e573b.appspot.com",
  messagingSenderId: "627244814751",
  appId: "1:627244814751:web:ab3a1afe615111958aadaa",
};

firebase.initializeApp(firebaseConfig);

attachCustomCommands({ Cypress, cy, firebase });
```

Лістинг файлу index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <link rel="preconnect" href="https://fonts.googleapis.com" />
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
    <link

href="https://fonts.googleapis.com/css2?family=Inter:wght@400;700&display=sw
ap"
      rel="stylesheet"
    />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>Projects</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

Лістинг файлу Nav.js

```
import React, { useContext } from "react";
import { Container } from "@ui";
import { UserContext } from "@context";
import { useNavigate } from "react-router";

import s from "./Nav.module.css";

export const Nav = () => {
  const navigate = useNavigate();
  const { auth } = useContext(UserContext);
```

```
  const onSignOut = () => {
    auth.signOut();
    navigate("/");
  };

  return (
    <div className={s.root}>
      <Container className={s.container}>
        <button onClick={() => navigate("/")}>На головну</button>
        <button onClick={onSignOut}>Вийти з системи</button>
      </Container>
    </div>
  );
};
```

Лістинг файлу Nav.module.css

```
.root {
  width: 100%;
  padding: 36px 0;

  border-bottom: 1px solid #151515;
}

.container {
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.root button {
  font-weight: bold;
  font-size: 32px;
  border: none;
  background: none;
  cursor: pointer;
}

.root button:hover {
  text-decoration: underline;
}
```

Лістинг файлу CreateProject.js

```
import React, { useContext, useState } from "react";
import { Input, Button } from "@ui";
import { projectsColRef } from "@constants/firebase";
import { setDoc } from "firebase/firestore";
import { UserContext, ProjectsContext } from "@context";
import { useNavigate } from "react-router-dom";
import { v4 as uuid } from "uuid";

import s from "./CreateProject.module.css";

export const CreateProject = () => {
  const navigate = useNavigate();
  const { user } = useContext(UserContext);
  const { projects, invalidate } = useContext(ProjectsContext);
  const [newProjName, setNewProjName] = useState("");

  const onCreate = () => {
    if (newProjName) {
      setDoc(projectsColRef(user.uid), {
        projects: [...projects, { name: newProjName, id: uuid() }],
      })
        .then(() => {
          invalidate();
          navigate("/");
        })
        .catch((e) => console.error(e));
    }
  };

  return (
    <div className={s.root}>
      <div className={s.container}>
        <Input
          placeholder="Назва проекту"
          value={newProjName}
          setValue={setNewProjName}
          data-cy="name"
        />

        <div className={s.buttons}>
          <Button onClick={onCreate} data-cy="submit">
```

```
          Створити
        </Button>
        <Button>Назад до проектів</Button>
      </div>
    </div>
  </div>
 );
};
```

Лістинг файлу CreateProject.module.css

```
.root {
 display: flex;
 justify-content: center;
 align-items: center;
 width: 100%;
 height: 100%;
}

.container {
 display: flex;
 flex-direction: column;
 align-items: center;
 width: 306px;
 transform: scale(1.5);
}

.buttons {
 display: flex;
 flex-direction: column;
 gap: 10px;
 width: 100%;
 margin-top: 17px;
}
```

Лістинг файлу ProjectNav.js

```
import React, { useState, useEffect } from "react";
import { Container, Button, Modal, Input } from "@ui";
import { tasksColRef } from "@constants/firebase";
import { getDoc } from "firebase/firestore";
import { useParams, useLocation, useNavigate } from "react-router-dom";
import { matchPath } from "react-router";
```

```
import cx from "classnames";
import { run } from "./utils";

import s from "./ProjectNav.module.css";

export const ProjectNav = ({ setTasks }) => {
  const location = useLocation();
  const navigate = useNavigate();

  const { projectId } = useParams();
  const [isCreating, setIsCreating] = useState(false);
  const [inv, setInv] = useState(false);

  const [newTaskName, setNewTaskName] = useState("");
  const [newTaskDuration, setNewTaskDuration] = useState("");
  const [newTaskPriority, setNewTaskPriority] = useState("");

  const isActivePage = matchPath(
    { path: location.pathname },
    `/project/${projectId}/active`,
  );
  const isInactivePage = matchPath(
    { path: location.pathname },
    `/project/${projectId}/inactive`,
  );
  const isSettingsPage = matchPath(
    { path: location.pathname },
    `/project/${projectId}/settings`,
  );

  useEffect(() => {
    async function fetchData() {
      try {
        const docSnap = await getDoc(tasksColRef(projectId));
        if (docSnap.exists() && docSnap.data().tasks && setTasks) {
          setTasks(
            docSnap
              .data()
              .projects.filter(
                (proj) =>
                  proj.status === (isActivePage ? "active" : "inactive"),
              ),
          );
```

```
      }
    } catch (error) {
      console.log(error);
    }
  }

  if (projectId) {
    fetchData();
  }
}, [projectId, inv]);

useEffect(() => {
  run();
}, []);

return (
  <div className={s.nav}>
    <Container className={s.navInner}>
      <div className={s.navInnerLinks}>
        <span
          className={cx(s.navInnerLink, isActivePage && s.active)}
          onClick={() => navigate(`/project/${projectId}/active`)}
        >
          Активні задачі
        </span>
        <span
          className={cx(s.navInnerLink, isInactivePage && s.active)}
          onClick={() => navigate(`/project/${projectId}/inactive`)}
        >
          Завершені задачі
        </span>
        <span
          className={cx(s.navInnerLink, isSettingsPage && s.active)}
          onClick={() => navigate(`/project/${projectId}/settings`)}
        >
          Налаштування проекту
        </span>
      </div>
      <Button className={s.navButton} onClick={() => setIsCreating(true)}>
        Створити нову задачу
      </Button>
    </Container>
    {isCreating && (
```

```
      <Modal onClose={() => setIsCreating(false)}>
        <div className={s.modalInner}>
          <Input
            value={newTaskName}
            setValue={setNewTaskName}
            className={s.input}
          />
          <div className={s.modalInnerItem}>
            <span>Визначте тривалість:</span>
            <Input
              value={newTaskDuration}
              setValue={setNewTaskDuration}
              className={s.inputSmall}
            />
          </div>
          <div className={s.modalInnerItem}>
            <span>Призначте відповідальних:</span>
            <Input
              value={newTaskPriority}
              setValue={setNewTaskPriority}
              className={s.inputSmall}
            />
          </div>
          <div className={s.modalInnerItem}>
            <span>Визначте приорітетність:</span>
            <Input
              value={newTaskPriority}
              setValue={setNewTaskPriority}
              className={s.inputSmall}
            />
          </div>
          <div className={s.modalInnerSubmit}>
            <Button>Створити</Button>
          </div>
        </div>
      </Modal>
    )}
  </div>
 );
};
```

Лістинг файлу ProjectNav.module.css

```
.nav {
  width: 100%;
  padding: 25px 0;
  border-bottom: 1px solid #151515;
}

.navInner {
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.navInnerLinks {
  display: flex;
  gap: 70px;
  font-size: 24px;
  font-weight: 700;
}

.navInnerLink {
  cursor: pointer;
}

.navInnerLink:hover,
.active {
  text-decoration: underline;
}

.navButton {
  max-width: 260px;
  border-radius: 10px;
}

.modalInner {
  display: flex;
  flex-direction: column;
  gap: 28px;
  color: #151515;
  font-size: 20px;
  font-weight: bold;
}
```

```
.input {
  border: none;
  box-shadow: none;
  border-bottom: 2px solid #151515;
  color: #151515;
  font-size: 24px;
  font-weight: bold;
  padding: 0;
}

.input:focus {
  outline: none;
}

.inputSmall {
  width: 40px;
  height: 40px;
  border-radius: 8px;
  border: 1px solid #151515;
  font-size: 24px;
  font-weight: bold;
  color: #151515;
}

.modalInnerItem {
  display: flex;
  align-items: center;
  gap: 10px;
}

.modalInnerSubmit {
  display: flex;
  justify-content: flex-end;
}

.modalInnerSubmit button {
  max-width: 168px;
}
```

Лістинг файлу ProjectTasks.js

```
import React, { useState } from "react";
import { Container, Task } from "@ui";
```

```
import s from "./ProjectTasks.module.css";
import { ProjectNav } from "../nav/ProjectNav";

export const ProjectTasks = () => {
  const [tasks, setTasks] = useState([]);

  return (
    <>
      <div className={s.root}>
        <ProjectNav setTasks={setTasks} />
        <Container className={s.content}>
          {tasks.map((task) => (
            <Task
              key={task.id}
              name={task.name}
              timeLeft={task.timeLeft}
              priority={task.priority}
              responsible={task.responsible}
            />
          ))}
        </Container>
      </div>
    </>
  );
};
```

Лістинг файлу ProjectTasks.module.css

```
.root {
  display: flex;
  flex-direction: column;
}

.content {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr 1fr;
  grid-auto-flow: row dense;
  gap: 38px;
  margin-top: 40px;
}
```

Лістинг файлу Projects.js

```
import React, { useContext } from "react";
import { Container, Button, Project } from "@ui";
import { ReactComponent as IconPlus } from "./icons/iconPlus.svg";
import { ProjectsContext } from "@context";
import { useNavigate, Link } from "react-router-dom";

import s from "./Projects.module.css";

export const Projects = () => {
  const navigate = useNavigate();
  const { projects } = useContext(ProjectsContext);

  return (
    <Container className={s.root}>
      <div className={s.buttons}>
        <Button
          className={s.button}
          onClick={() => navigate("/create-project")}
        >
          <div className={s.buttonInner}>
            <span>Створити новий проект</span>
            <IconPlus />
          </div>
        </Button>
        <Button className={s.button}>
          <div className={s.buttonInner}>
            <span>Приєднатись до існуючого проекту</span>
            <IconPlus />
          </div>
        </Button>
      </div>
      {projects.map((proj) => (
        <Link to={`/project/${proj.id}/active`} key={proj.id}>
          <Project name={proj.name} />
        </Link>
      ))}
    </Container>
  );
};
```

Лістинг файлу Projects.module.css

```
.root {
  display: flex;
  flex-direction: column;
  gap: 65px;
  margin-top: 65px;
  font-size: 36px;
  font-weight: bold;
  cursor: pointer;
}

.root a {
  color: #151515;
  text-decoration: none;
}

.buttons {
  display: flex;
  justify-content: space-between;
  align-items: center;
  gap: 25px;
}

.button {
  padding: 24px 45px;
  border-radius: 10px;
  transition: 0.2s ease-in-out 0s;
}

.button:hover {
  transform: scale(1.02);
}

a:hover {
  text-decoration: underline;
}

.buttonInner {
  display: flex;
  justify-content: space-between;
  align-items: center;
  font-size: 36px;
  font-weight: bold;
}
```

Лістинг файлу Register.js

```
import React, { useContext, useState } from "react";
import { Input, Button } from "@ui";
import { Link, useNavigate } from "react-router-dom";
import { createUserWithEmailAndPassword } from "firebase/auth";
import { UserContext } from "@context";

import s from "./Register.module.css";

export const Register = () => {
  const navigate = useNavigate();
  const { auth } = useContext(UserContext);

  const [email, setEmail] = useState("");
  const [pass, setPass] = useState("");
  const [passRepeat, setPassRepeat] = useState("");
  const [name, setName] = useState("");

  const onReg = () => {
    if (email && pass && pass === passRepeat && name) {
      createUserWithEmailAndPassword(auth, email, pass).then(() => {
        navigate("/");
      });
    }
  };

  return (
    <div className={s.root}>
      <div className={s.container}>
        <Input
          placeholder="Ел.пошта"
          type="email"
          required
          value={email}
          setValue={setEmail}
          data-cy="email"
        />
        <Input
          placeholder="Пароль"
          type="password"
          value={pass}
```

```
          setValue={setPass}
          data-cy="password"
        />
        <Input
          placeholder="Повторіть пароль"
          type="password"
          value={passRepeat}
          setValue={setPassRepeat}
          data-cy="passwordRepeat"
        />
        <Input
          placeholder="Ваше ім'я"
          value={name}
          setValue={setName}
          data-cy="name"
        />

        <div className={s.buttons}>
          <Button onClick={onReg} data-cy="submit">
            Зареєструватись
          </Button>
          <Link to="/">
            <Button>Назад до авторизації</Button>
          </Link>
        </div>
      </div>
    </div>
  );
};
```

Лістинг файлу Register.js

```
.root {
  display: flex;
  justify-content: center;
  align-items: center;
  width: 100%;
  height: 100%;
}

.container {
  display: flex;
```

```
  flex-direction: column;
  align-items: center;
  gap: 15px;
  width: 306px;
  transform: scale(1.5);
}

.buttons {
  display: flex;
  flex-direction: column;
  gap: 15px;
  width: 100%;
  margin-top: 17px;
}
```

Лістинг файлу SignIn.js

```
import React, { useState, useContext } from "react";
import { Input, Button } from "@ui";
import { Link } from "react-router-dom";
import { signInWithEmailAndPassword } from "firebase/auth";
import { UserContext } from "@context";

import s from "./SignIn.module.css";
import { SignInWithGoogle } from "./components";

export const SignIn = () => {
  const [email, setEmail] = useState("");
  const [pass, setPass] = useState("");
  const { auth } = useContext(UserContext);

  const onLogin = () => {
    if (email && pass) {
      signInWithEmailAndPassword(auth, email, pass);
    }
  };

  return (
    <div className={s.root}>
      <div className={s.container}>
        <Input
          placeholder="Ел.пошта"
```

```
          value={email}
          setValue={setEmail}
          data-cy="email"
        />
        <Input
          placeholder="Пароль"
          value={pass}
          setValue={setPass}
          type="password"
          data-cy="password"
        />

        <div className={s.buttons}>
          <Button onClick={onLogin} data-cy="submit">
            Увійти
          </Button>
          <Link to="register">
            <Button>Зареєструватись</Button>
          </Link>
          <SignInWithGoogle />
        </div>
      </div>
    </div>
  );
};
```

Лістинг файлу SignIn.module.css

```
.root {
  display: flex;
  justify-content: center;
  align-items: center;
  width: 100%;
  height: 100%;
}

.container {
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 28px;
  width: 306px;
  transform: scale(1.5);
```

```
}

.buttons {
  display: flex;
  flex-direction: column;
  gap: 12px;
  width: 100%;
}
```

Лістинг файлу SignInWithGoogle.js

```
import React from "react";
import cx from "classnames";
import { ReactComponent as GoogleLogo } from "./icons/google.svg";

import s from "./SignInWithGoogle.module.css";
import { getAuth, signInWithPopup } from "firebase/auth";
import { app, googleAuthProvider } from "@constants/firebase";

export const SignInWithGoogle = ({ className, ...props }) => {
  const auth = getAuth(app);
  const onButtonClick = () =>
    signInWithPopup(auth, googleAuthProvider).catch((error) => {
      console.log("Caught error Popup closed");
    });

  return (
    <button
      className={cx(s.root, className)}
      onClick={onButtonClick}
      {...props}
    >
      <GoogleLogo />
      <span>Увійти за допомогою Google</span>
    </button>
  );
};
```

Лістинг файлу SignInWithGoogle.module.css

```
.root {
  display: flex;
  align-items: center;
```

```
  gap: 32px;
  padding: 10px;
  color: #3c4043;
  line-height: 17px;
  font-size: 14px;
  border-radius: 4px;
  border: 1px solid #dadce0;
  background: #fff;
  cursor: pointer;
  height: 36px;
  transition: 0.2s ease-in;
}

.root:hover {
  transform: scale(1.03);
}
```

Лістинг файлу firebase.js

```
import { initializeApp } from "firebase/app";
import { GoogleAuthProvider } from "firebase/auth";
import { getFirestore, doc } from "firebase/firestore";

const firebaseConfig = {
  apiKey: "AIzaSyB9E1DxgjhQgmrYGE01e_zOZRIAtY4HvAk",
  authDomain: "magisters-e573b.firebaseapp.com",
  projectId: "magisters-e573b",
  storageBucket: "magisters-e573b.appspot.com",
  messagingSenderId: "627244814751",
  appId: "1:627244814751:web:ab3a1afe615111958aadaa",
};

export const app = initializeApp(firebaseConfig);
export const googleAuthProvider = new GoogleAuthProvider();
const db = getFirestore();
export const projectsColRef = (userId) => doc(db, "projects", userId);
export const tasksColRef = (projectId) => doc(db, "tasks", projectId);
```

Лістинг файлу ProjectContextProvider.js

```
import React, { useContext, useState, useEffect } from "react";
import { projectsColRef } from "@constants/firebase";
```

```
import { getDoc } from "firebase/firestore";
import { UserContext } from "../index";
import { ProjectsContext } from "./ProjectsContext";

export const ProjectsContextProvider = ({ children }) => {
  const { user } = useContext(UserContext);
  const [projects, setProjects] = useState([]);
  const [inv, setInv] = useState(false);

  useEffect(() => {
    async function fetchData() {
      try {
        const docSnap = await getDoc(projectsColRef(user.uid));
        if (docSnap.exists() && docSnap.data().projects) {
          setProjects(docSnap.data().projects);
        }
      } catch (error) {
        console.log(error);
      }
    }

    if (user) {
      fetchData();
    }
  }, [user, inv]);

  const invalidate = () => setInv((s) => !s);

  return (
    <ProjectsContext.Provider value={{ projects, invalidate }}>
      {children}
    </ProjectsContext.Provider>
  );
};
```

Лістинг файлу UserContextProvider.js

```
import { getAuth } from "firebase/auth";
import { app } from "@constants/firebase";
import { useEffect, useState } from "react";
import { UserContext } from "./userContext";

export const UserContextProvider = ({ children }) => {
```

```
  const auth = getAuth(app);

  const [user, setUser] = useState(auth.currentUser);

  useEffect(() => {
    auth.onAuthStateChanged((maybeUser) => {
      setUser(maybeUser);
    });
  }, []);

  return (
    <UserContext.Provider value={{ auth, user, setUser }}>
      {children}
    </UserContext.Provider>
  );
};
```

Лістинг файлу Button.js

```
import React from "react";
import cx from "classnames";

import s from "./Button.module.css";

export const Button = ({ className, ...props }) => (
  <button className={cx(s.root, className)} {...props}>
    {props.children}
  </button>
);
```

Лістинг файлу Button.module.css

```
.root {
  padding: 8px;
  width: 100%;
  background-color: #151515;
  border: none;
  color: #fff;
  font-size: 16px;
  line-height: 20px;
  border-radius: 4px;
  cursor: pointer;
```

```
  transition: 0.2s ease-in-out 0s;
}

.root:hover {
  transform: scale(1.02);
}
```

Лістинг файлу Container.js

```
import React from "react";
import cx from "classnames";

import s from "./Container.module.css";

export const Container = ({ children, className, ...props }) => (
  <div className={cx(s.root, className)} {...props}>
    {children}
  </div>
);
```

Лістинг файлу Container.module.css

```
.root {
  width: 100%;
  max-width: 1920px;
  margin: 0 auto;
  padding: 0 60px;
}
```

Лістинг файлу Input.js

```
import React from "react";
import cx from "classnames";

import s from "./Input.module.css";

export const Input = ({ setValue, className, ...props }) => (
  <input
    onChange={(e) => setValue(e.target.value)}
    className={cx(s.root, className)}
    {...props}
  />
);
```

Лістинг файлу Input.module.css

```
.root {
  width: 100%;
  padding: 8px;
  box-shadow: 0px 1px 2px 0px rgba(55, 65, 81, 0.08);
  border-radius: 4px;
  border: 1px solid #e3e8ee;
  font-size: 14px;
  color: #697386;
  line-height: 20px;
}
```

Лістинг файлу Modal.js

```
import React from "react";
import { ReactComponent as Close } from "./icons/cross.svg";

import s from "./Modal.module.css";

export const Modal = ({ children, onClose }) => (
  <>
    <div className={s.backdrop} onClick={onClose}></div>
    <div className={s.root}>
      <div className={s.rootClose}>
        <button onClick={onClose}>
          <Close />
        </button>
      </div>
      {children}
    </div>
  </>
);
```

Лістинг файлу Modal.module.css

```
.backdrop {
  display: flex;
  justify-content: center;
  align-items: center;
  position: absolute;
  top: 0;
```

```
  left: 0;
  width: 100vw;
  height: 100vh;
  background: rgba(0, 0, 0, 0.5);
  z-index: 500;
}

.root {
  display: flex;
  flex-direction: column;
  background-color: #fff;
  width: 600px;
  border-radius: 12px;
  z-index: 501;
  position: absolute;
  left: 50%;
  top: 50%;
  transform: translate(-50%, -50%);
  padding: 24px;
}

.rootClose {
  display: flex;
  justify-content: flex-end;
  width: 100%;
  margin-bottom: 5px;
}

.rootClose button {
  background: none;
  border: none;
  cursor: pointer;
}
```

Лістинг файлу Project.js

```
import React from "react";
import cx from "classnames";
import { ReactComponent as ArrowRight } from "./icons/arrowRight.svg";

import s from "./Project.module.css";

export const Project = ({ className, name, ...props }) => (
```

```
  <div className={cx(s.root, className)} {...props}>
    <span>{name}</span>
    <ArrowRight />
  </div>
);
```

Лістинг файлу Project.module.css

```
.root {
  display: flex;
  justify-content: space-between;
  align-items: center;
  border-radius: 10px;
  border: 1px solid #151515;
  padding: 28px 40px;
  transition: 0.2s ease-in-out 0s;
}

.root svg {
  width: 40px;
  height: 40px;
}

.root:hover {
  transform: scale(1.02);
}
```

Лістинг файлу Task.js

```
import React from "react";
import { calcTaskColor } from "./utils/calcTaskColor";

import s from "./Task.module.css";

export const Task = ({ name, timeLeft, responsible, priority, isInactive }) => {
  const color = calcTaskColor(priority, timeLeft, isInactive);

  return (
    <div className={s.root} style={{ borderColor: color }}>
      <div className={s.rootContent}>
        <span style={{ color: isInactive ? "#9EA89D" : "auto" }}>{name}</span>
```

```
      <span style={{ color }}>{`${timeLeft} г.`}</span>
     </div>
     <div className={s.rootContent}>
      <div className={s.rootResponsible}>
       {responsible.map((i) => (
        <span
         key={i}
         style={{
          borderColor: isInactive ? "#9EA89D" : "auto",
          color: isInactive ? "#9EA89D" : "auto",
         }}
        >
         {i.substring(0, 1)}
        </span>
       ))}
      </div>
      <div className={s.rootPriority} style={{ borderColor: color, color }}>
       {priority}
      </div>
     </div>
    </div>
  );
};
```

Лістинг файлу Task.module.css

```
.root {
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  height: 147px;
  width: 420px;
  padding: 15px;
  font-size: 24px;
  font-weight: bold;
  border-width: 1px;
  border-style: solid;
  box-shadow: 0px 4px 4px 0px rgba(0, 0, 0, 0.25);
  border-radius: 4px;
}

.rootContent {
  display: flex;
```

```
  justify-content: space-between;
  align-items: center;
}

.rootResponsible {
  display: flex;
  align-items: center;
  gap: 10px;
}

.rootResponsible > * {
  display: flex;
  justify-content: center;
  align-items: center;
  width: 40px;
  height: 40px;
  border: 1px solid #151515;
  border-radius: 50%;
}

.rootContent span:nth-of-type(2) {
  align-self: baseline;
  flex-basis: 60px;
}

.rootPriority {
  display: flex;
  justify-content: center;
  align-items: center;
  width: 40px;
  height: 40px;
  border-width: 1px;
  border-style: solid;
  border-radius: 4px;
}
```

Лістинг файлу calcTaskColor.js

```
export const calcTaskColor = (priority, timeLeft, isInactive) => {
  if (isInactive) return "#9EA89D";
  if (+priority === 1 || timeLeft < 3) return "#B84242";
  if (+priority === 2) return "#B3B96E";
```

```
  return "#4BB842";
};
```

Лістинг файлу index.js

```
import React, { useContext } from "react";
import {
  SignIn,
  Register,
  CreateProject,
  Projects,
  ProjectTasks,
  ProjectSettings,
} from "@pages";
import {
  BrowserRouter as Router,
  Routes,
  Route,
  Navigate,
} from "react-router-dom";
import { Nav } from "./components";

import { UserContext } from "@context";

export const Magisters = () => {
  const { user } = useContext(UserContext);

  if (!user) {
    return (
      <Router>
        <Routes>
          <Route path={"/"} element={<SignIn />} />
          <Route path={"/register"} element={<Register />} />
        </Routes>
      </Router>
    );
  }

  return (
    <Router>
      <Nav />
      <Routes>
        <Route path={"/"} element={<Projects />} />
```

```
      <Route path={"/create-project"} element={<CreateProject />} />
      <Route path={"/project/:projectId/active"} element={<ProjectTasks />} />
      <Route
        path={"/project/:projectId/inactive"}
        element={<ProjectTasks />}
      />
      <Route
        path={"/project/:projectId/settings"}
        element={<ProjectSettings />}
      />
      <Route path="*" element={<Navigate to="/" />} />
    </Routes>
  </Router>
 );
};
```

Лістинг файлу App.js

```
import "./index.css";
import { UserContextProvider, ProjectsContextProvider } from "@context";
import { Magisters } from "./app/index";

function App() {
 return (
  <UserContextProvider>
   <ProjectsContextProvider>
    <Magisters />
   </ProjectsContextProvider>
  </UserContextProvider>
 );
}

export default App;
```

Додаток Г

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА МЕТОДІВ І ПРОГРАМНИХ ЗАСОБІВ УПРАВЛІННЯ ПРОЄКТАМИ ТА ЗАДАЧАМИ В КОМАНДНОМУ СЕРЕДОВИЩІ

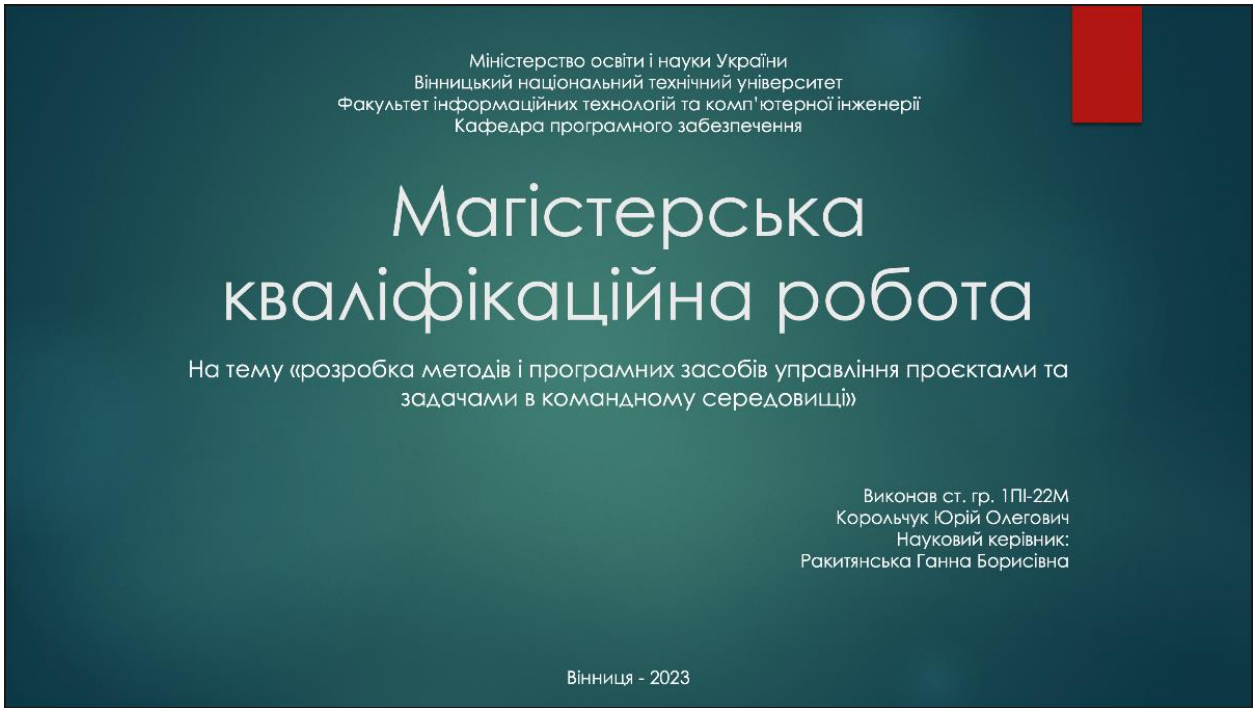

Рисунок Г.1 – Назва роботи

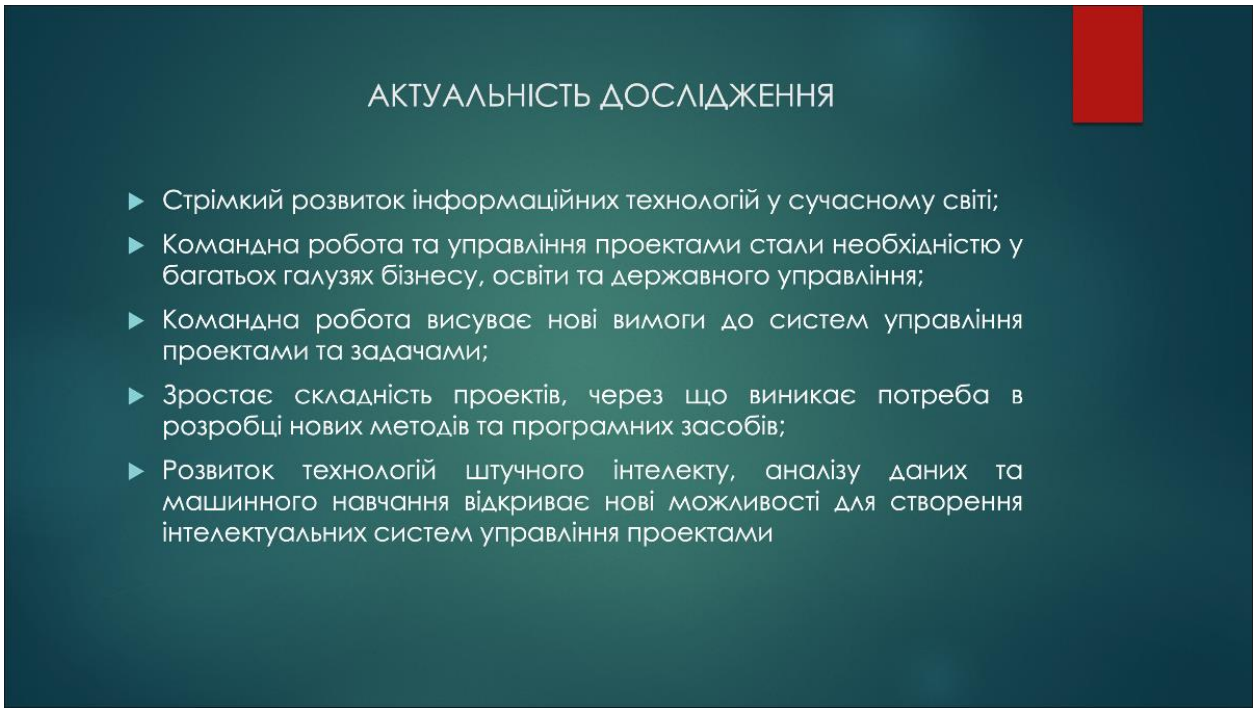

Рисунок Г.2 – Актуальність дослідження

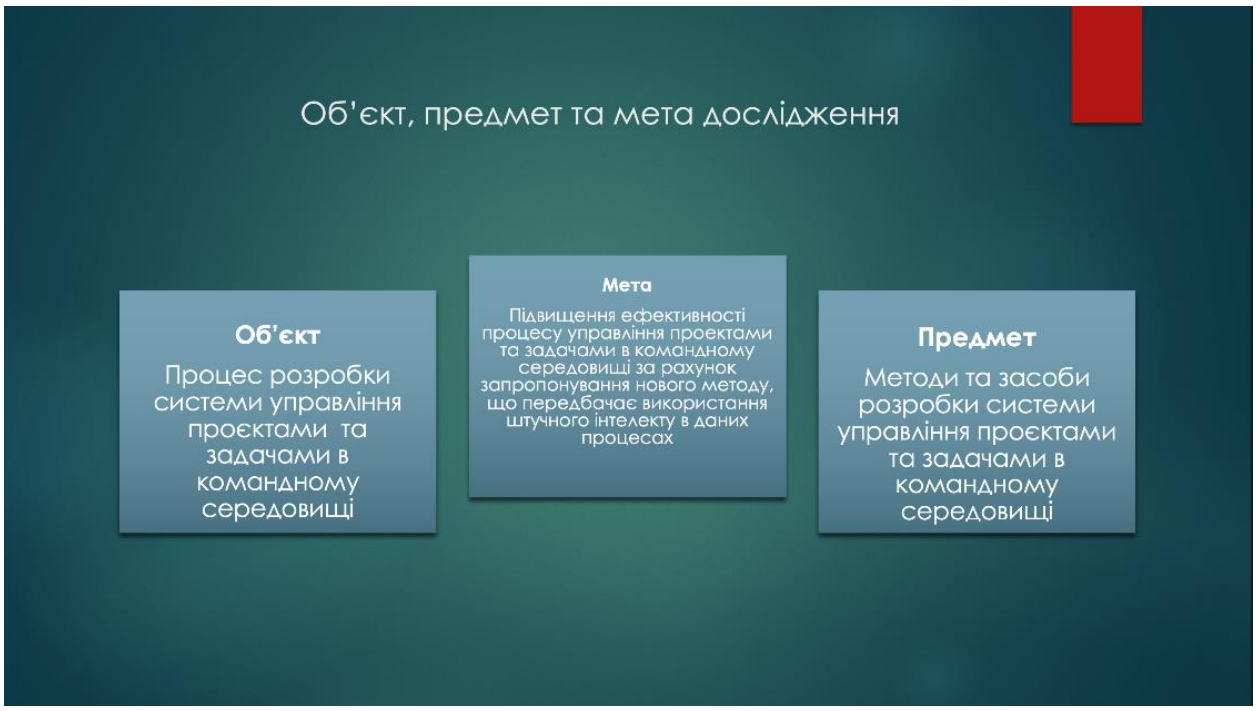

Рисунок Г.3 – Об’єкт, предмет та мета дослідження

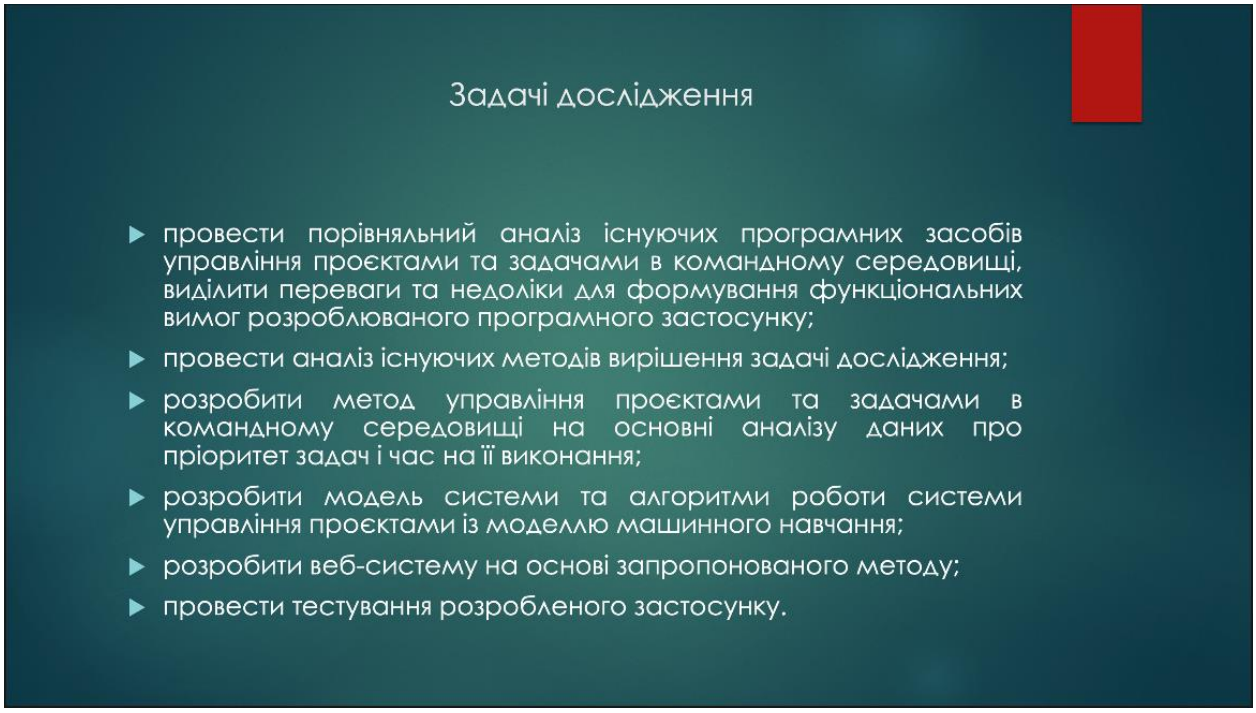

Рисунок Г.4 – Задачі дослідження

Рисунок Г.5 – Етапи розробки

Критерії	MS Project	Trello	Asana	Jira	Розроблювана система
Простий інтерфейс	-	+	+	-	+
Складний в налаштуванні	+	-	+	+	-
Необхідність інших інструментів для повного функціоналу	+	+	+	+	-
Наявність безкоштовної версії	-	+	+	-	+
Наявність засобів аналітики	+	-	-	-	+
Складний в інтеграції	+	-	+	+	-

Рисунок Г.6 – Порівняння з аналогами

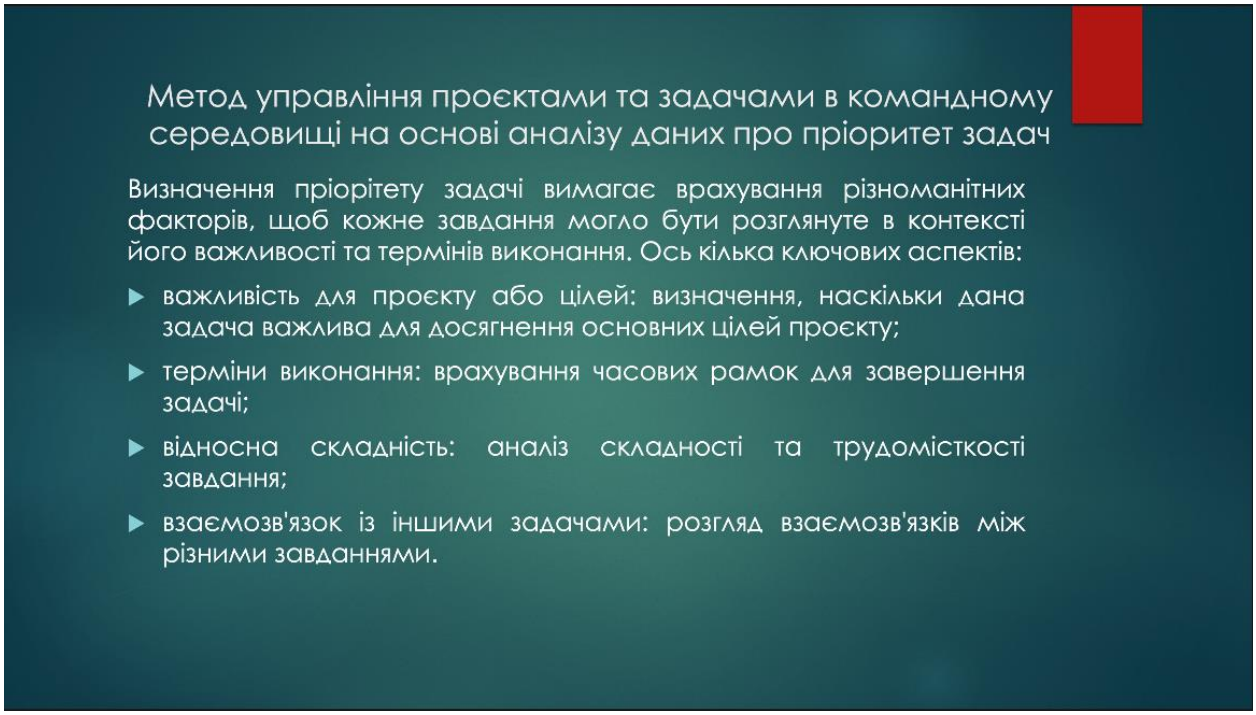

Рисунок Г.7 – Метод управління проєктами та задачами в командному середовищі на основі аналізу даних про пріоритет задач (1)

Рисунок Г.8 – Метод управління проєктами та задачами в командному середовищі на основі аналізу даних про пріоритет задач (2)

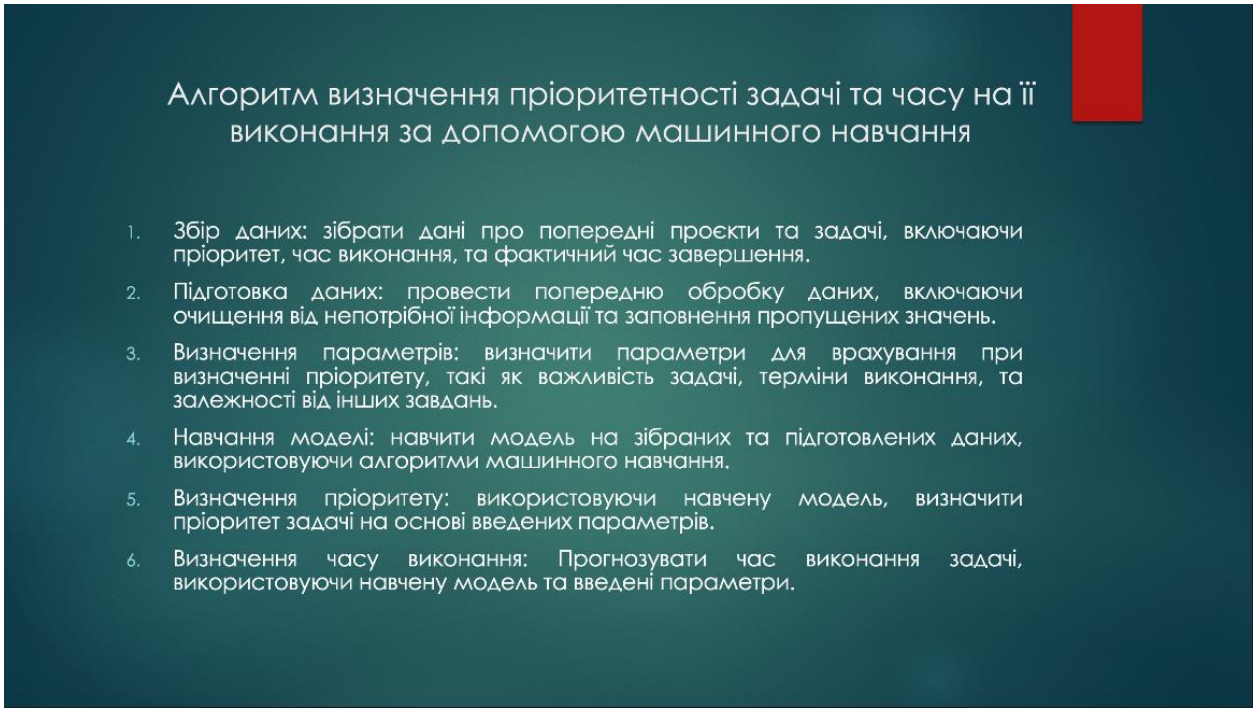

Рисунок Г.9 – Алгоритм визначення пріоритетності задачі та часу на її виконання за допомогою машинного навчання

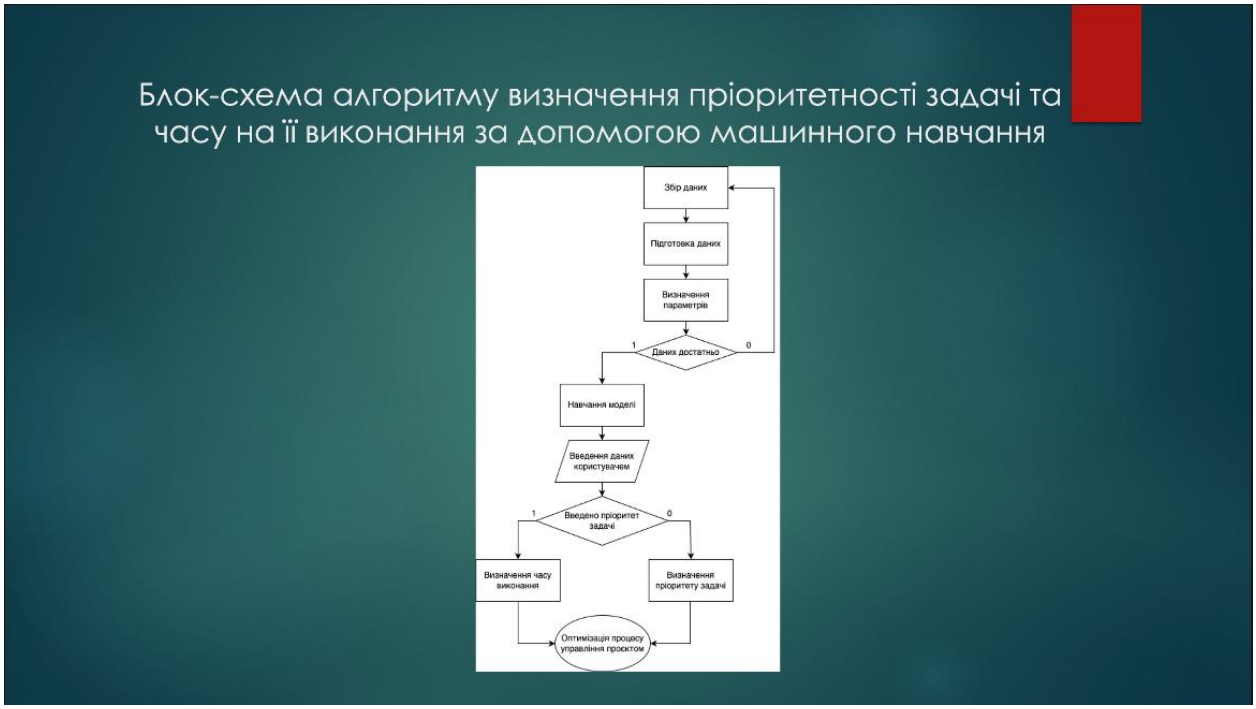

Рисунок Г.10 – Блок-схема алгоритму визначення пріоритетності задачі та часу на її виконання за допомогою машинного навчання

Рисунок Г.11 – Основні алгоритми застосунку

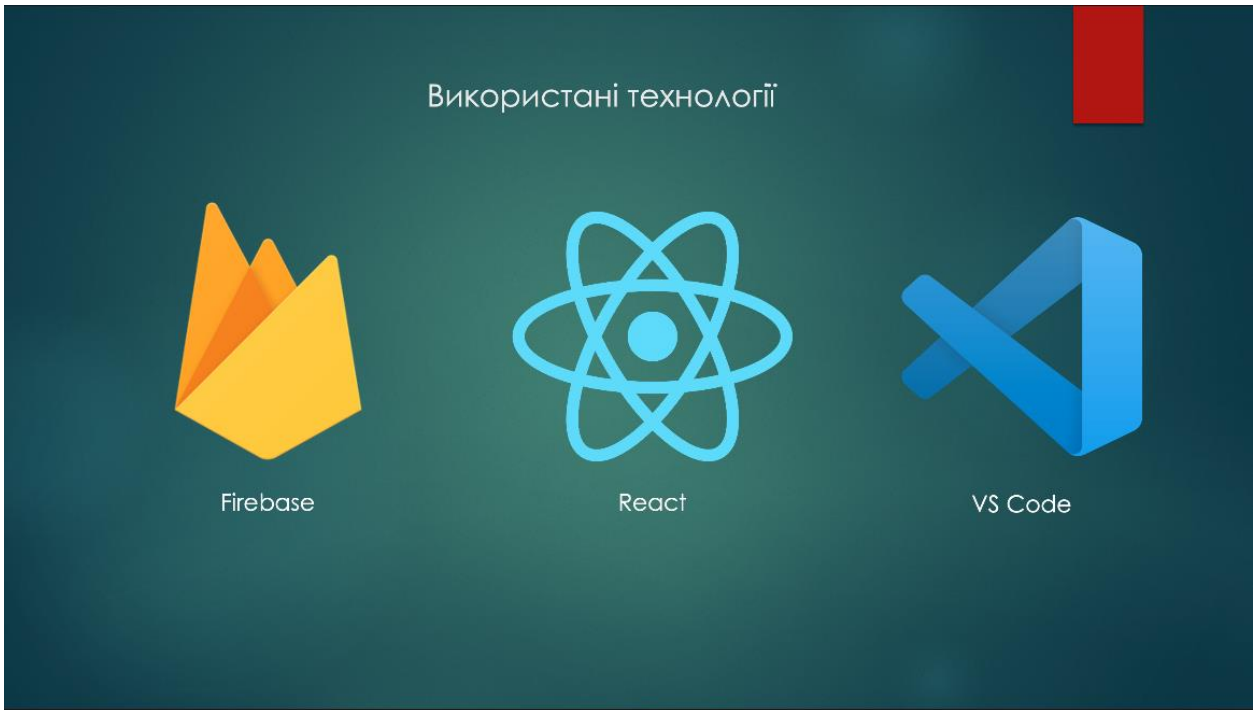

Рисунок Г.12 – Використані технології

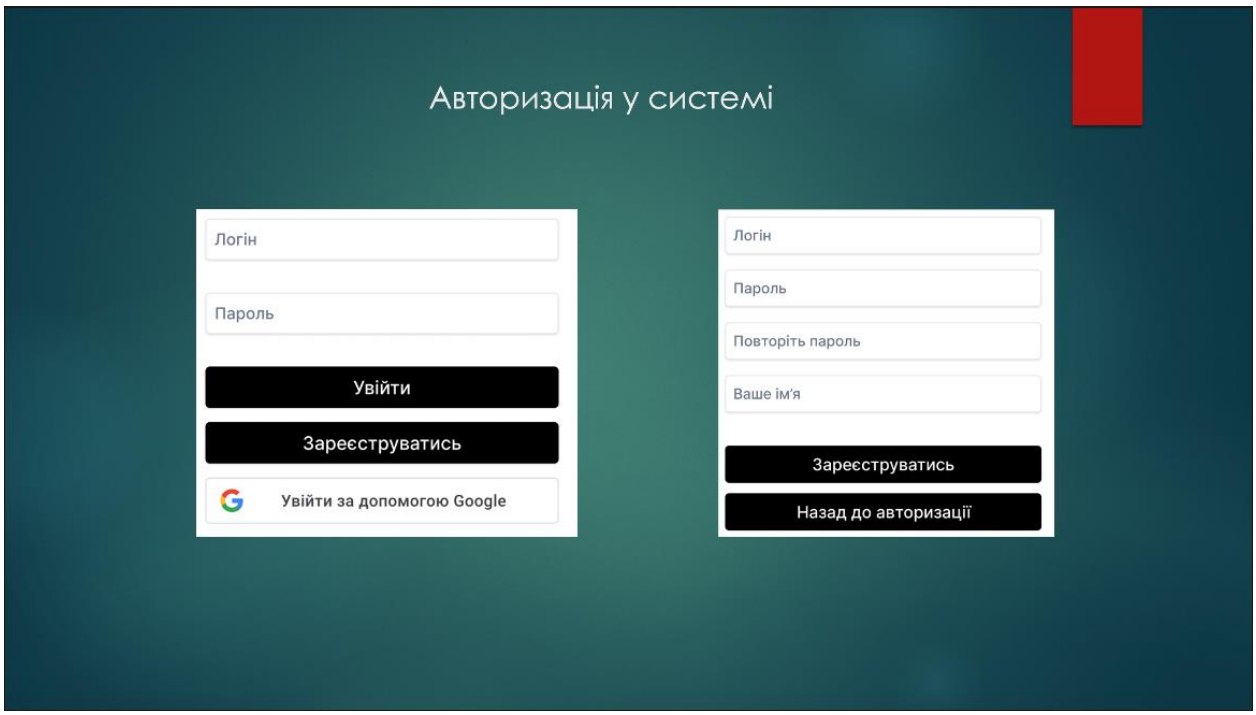

Рисунок Г.13 – Авторизація у системі

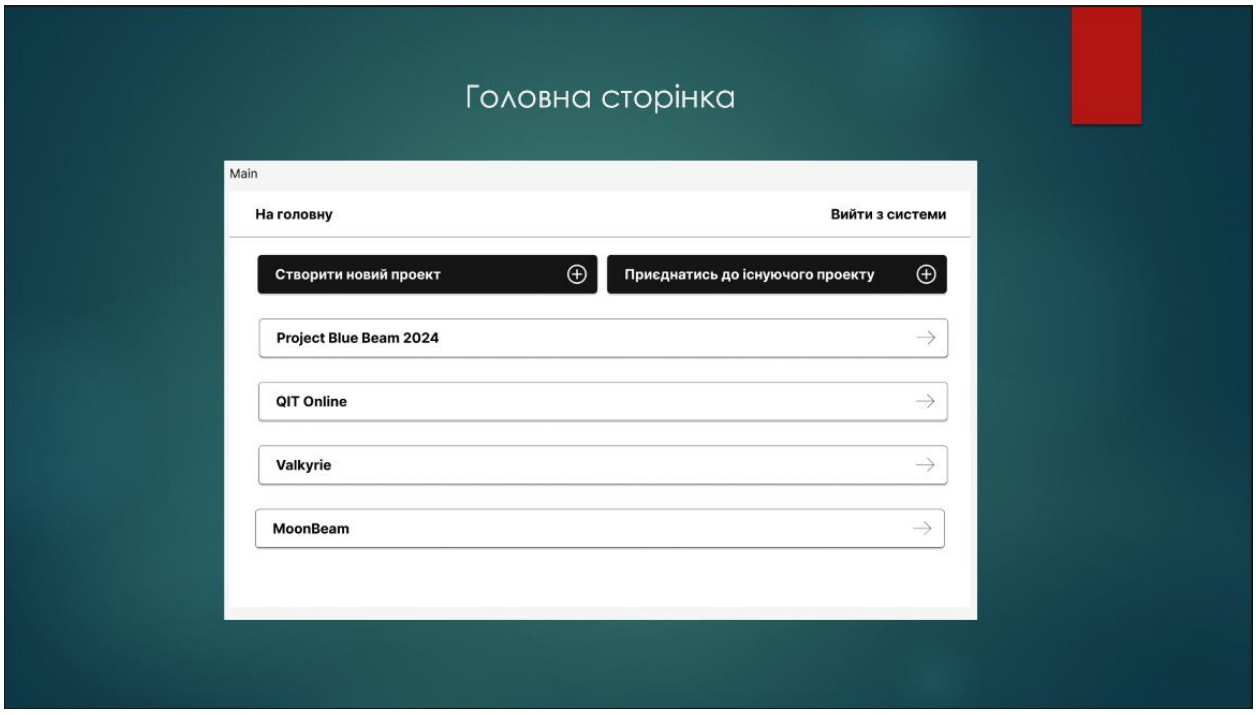

Рисунок Г.14 – Головна сторінка

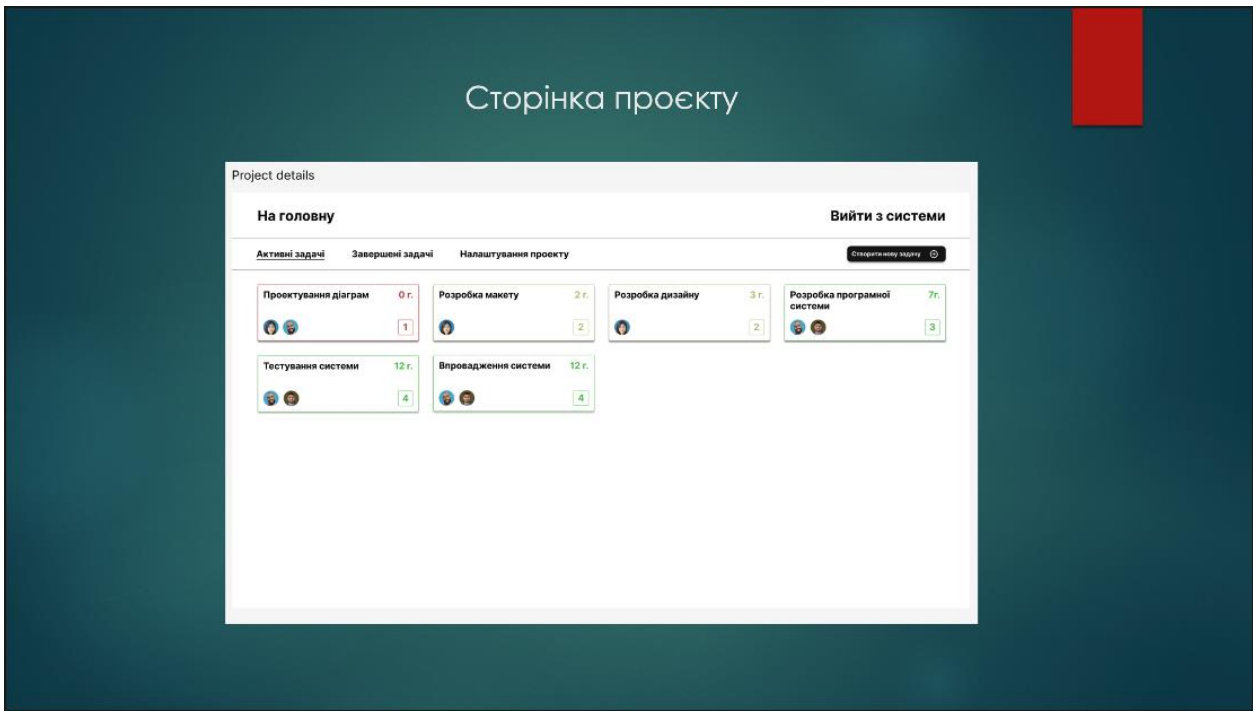

Рисунок Г.15 – Сторінка проєкту

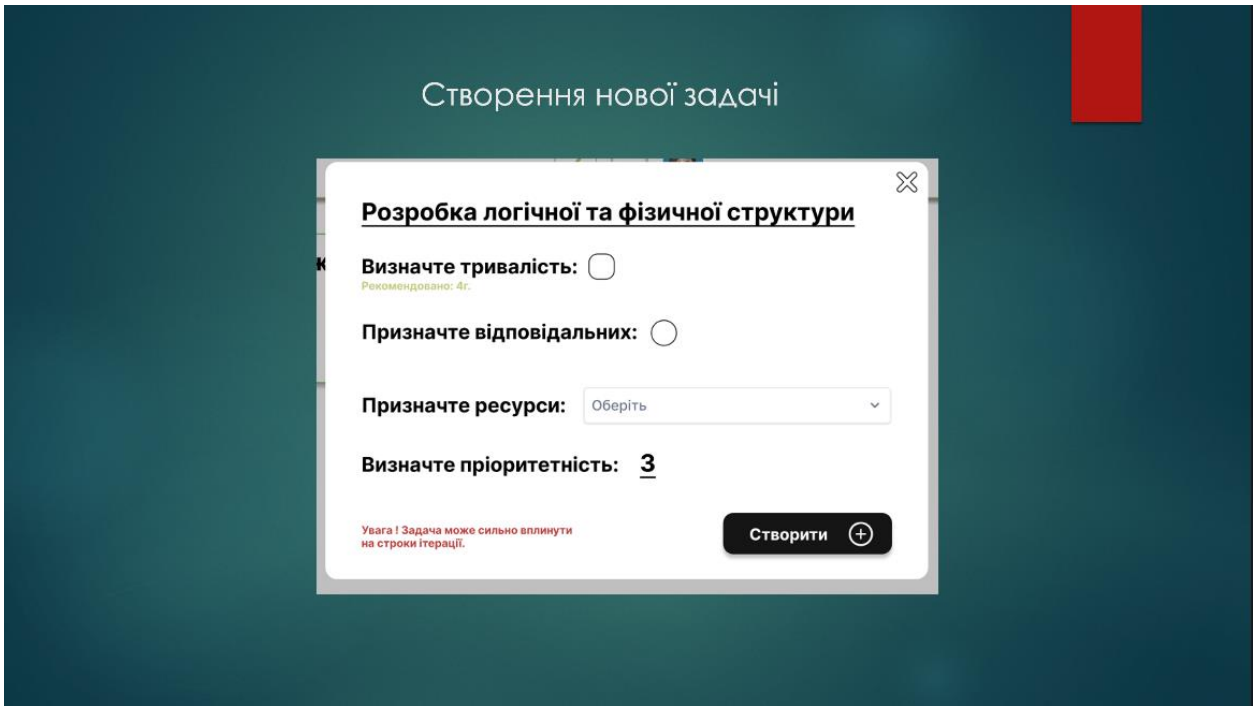

Рисунок Г.16 – Створення нової задачі з використанням алгоритму визначення пріоритетності задачі та часу на її виконання за допомогою машинного навчання

Рисунок Г.17 – Тестування

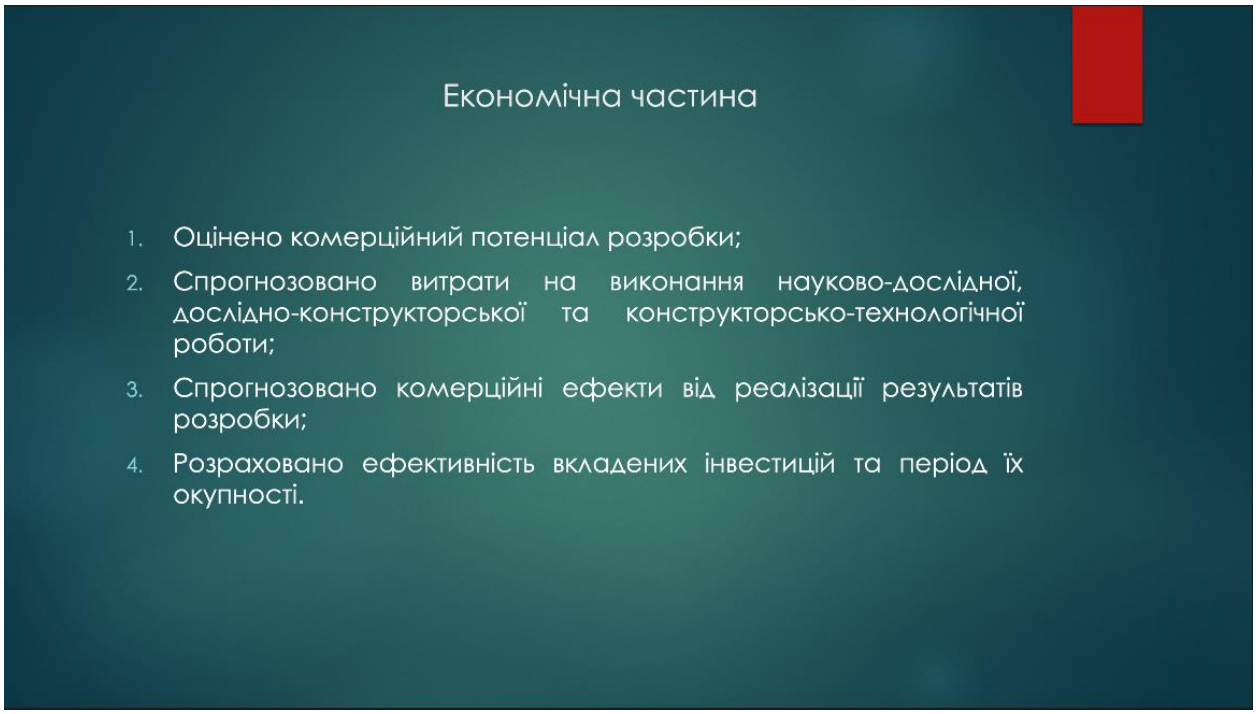

Рисунок Г.18 – Економічна частина

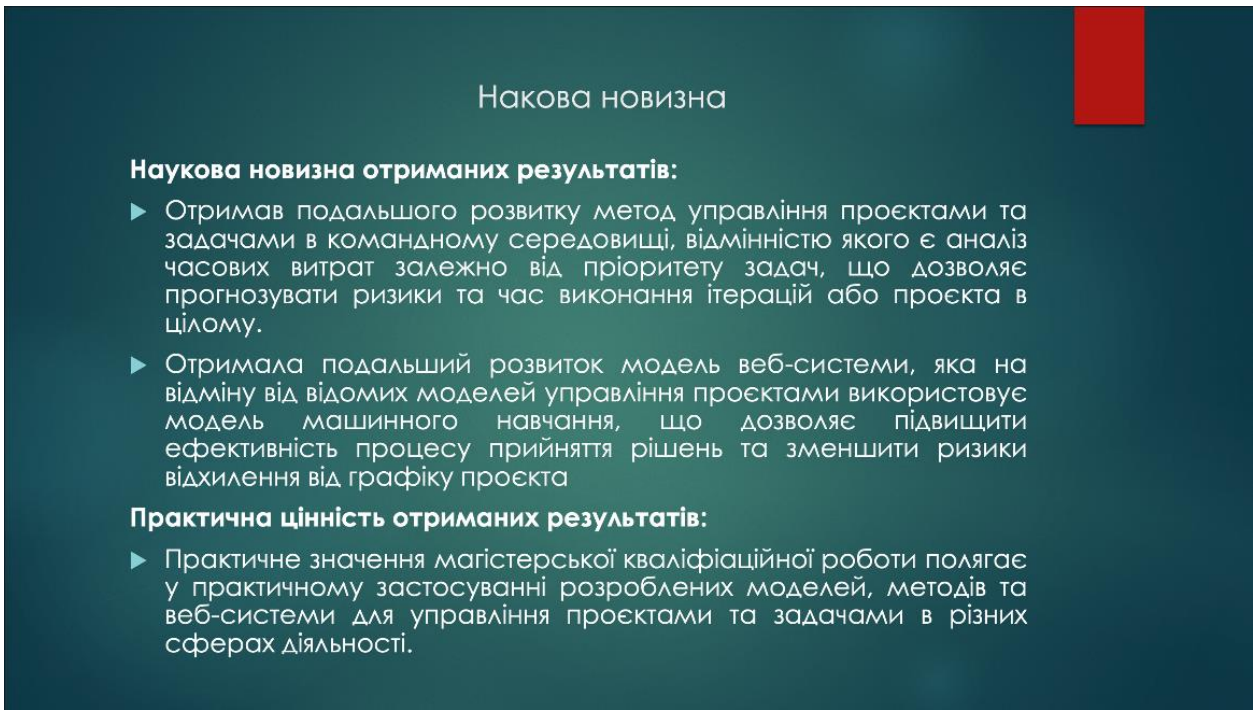

Рисунок Г.19 – Наукова новизна

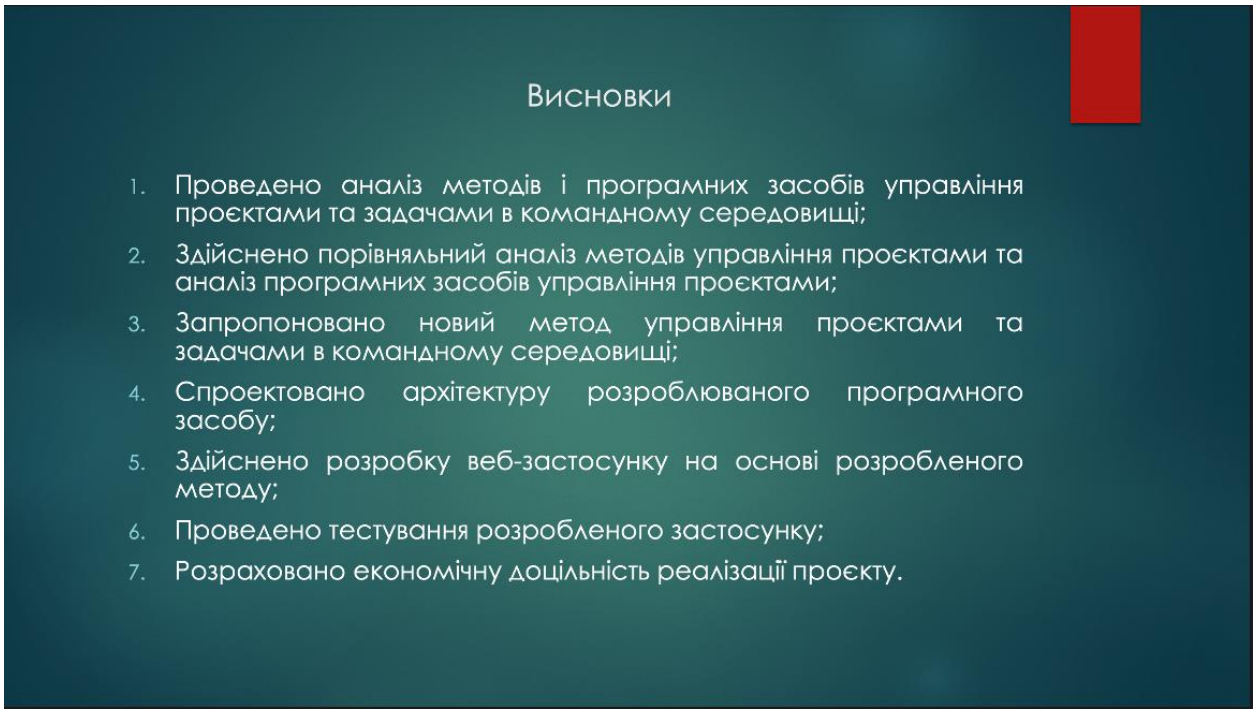

Рисунок Г.20 – Висновки

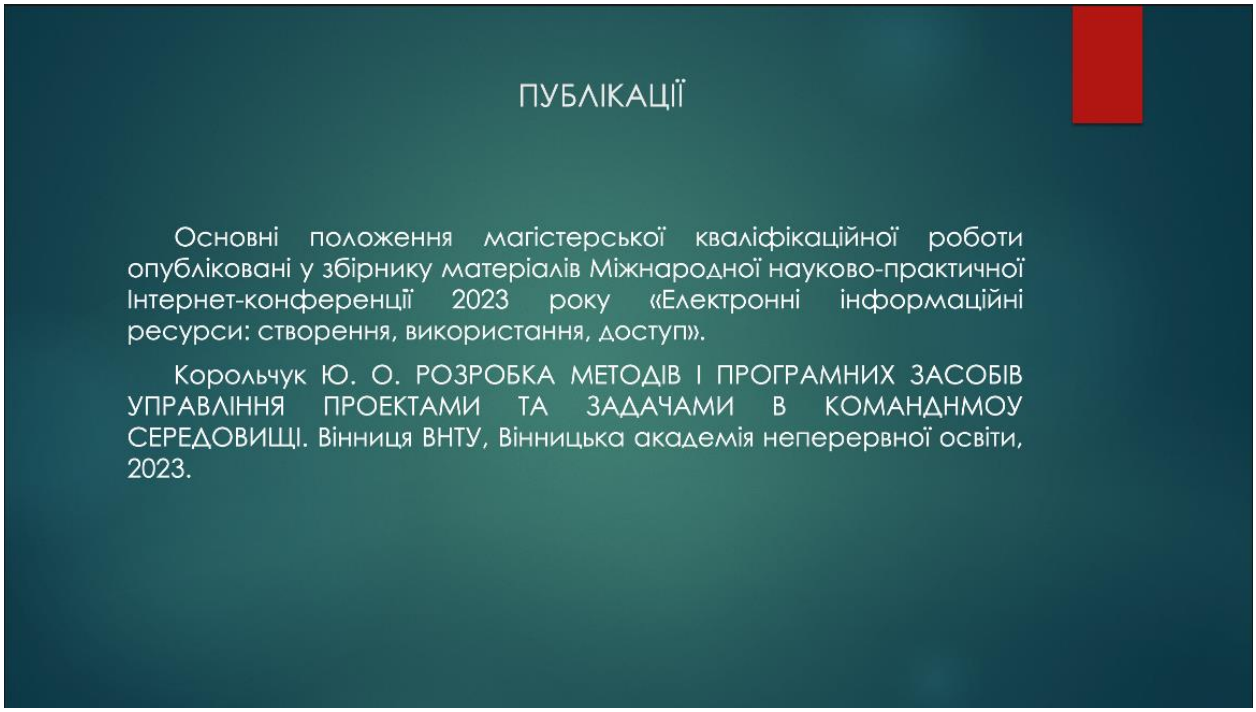

Рисунок Г.21 – Публікації