МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри, канд. техн. наук,
доцент _____ Є. О. Давиденко
«___« _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

СИСТЕМА КЕРУВАННЯ ПРОЄКТАМИ НА ОСНОВІ ЗАСОБІВ ШТУЧНОГО ІНТЕЛЕКТУ

Спеціальність «Інженерія програмного забезпечення»

121 – КРМ.1 – 608м.21810904

Здобувачка вищої освіти _____ С. В. Бондаренко
«___«_____2024 р.

Керівник завідувач кафедри ІПЗ, канд. техн. наук, доцент _____ Є. О. Давиденко
«___«_____2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ
Зав. кафедри
_____ Є. О. Давиденко
« __ » _____ 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи магістра

Видано студентці групи 608м факультету комп'ютерних наук

Бондаренко Стефанії Віталіївні .
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи:

Система керування проєктами на основі засобів штучного інтелекту

Затверджена наказом по ЧНУ ім. П. Могили від «10» листопада 2023 р. № 234

2. Строк представлення кваліфікаційної роботи: « __ » лютого 2024 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом є система керування проєктами на основі засобі штучного інтелекту.

4. Перелік питань, що підлягають розробці:

- вивчення та аналіз фахової області та існуючих аналогів;
- формування специфікації вимог до програмного забезпечення;
- визначення архітектурних принципів для проєктування програмного забезпечення;
- моделювання та проєктування програмного забезпечення;
- розробка програмного забезпечення відповідно до визначених вимог;
- проведення тестування програмного забезпечення для перевірки його роботи та якості;
- проведення аналізу результатів розробки для визначення відповідності поставленим вимогам і пошук можливих покращень.

5. Перелік графічних матеріалів:

презентація.

Керівник роботи канд. техн. наук, доцент Давиденко Євген Олександрович
(посада, прізвище, ім'я, по батькові)

_____
(підпис)

Завдання прийнято до виконання

Бондаренко Стефанія Віталіївна

(прізвище, ім'я, по батькові студента)

_____
(підпис)

Дата видачі завдання « ___ » _____ 202__ р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема кваліфікаційної роботи:

Система керування проєктами на основі засобів штучного інтелекту

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРМ	03.09.2023 р.	04.09.2023 р	*виконано*
2.	Огляд літератури за темою роботи	06.09.2023 р.	08.09.2023 р.	*виконано*
3.	Складання календарного плану КРМ	09.09.2023 р.	10.09.2023 р.	*виконано*
4.	Аналіз предметної області	15.09.2023 р.	15.09.2023 р.	*виконано*
5.	Розробка проєктних рішень	03.10.2023 р.	04.10.2023 р.	*виконано*
6.	Моделювання та конструювання ПЗ	05.10.2023 р.	10.10.2023 р.	*виконано*
7.	Кодування, тестування розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	16.10.2023 р.	22.11.2023 р.	*виконано*
8.	Розробка спеціальної частини з охорони праці	23.11.2023 р.	28.11.2023 р.	*виконано*
9.	Оформлення КРМ та презентації	01.12.2023 р.	05.12.2023 р.	*виконано*
10.	Відгук керівника КРМ	06.12.2024 р.	07.12.2023 р.	*виконано*
11.	Попередній захист	08.12.2023 р.	08.12.2023 р.	*виконано*
12.	Завершення оформлення КРМ та презентації	15.12.2023 р.	17.12.2023 р.	*виконано*
13.	Рецензування	16.01.2024 р.	19.01.2024 р.	*виконано*
14.	Захист кваліфікаційної роботи	26.02.2024 р.	28.02.2024 р.	

Розробила студентка Бондаренко Стефанія Віталіївна _____
(прізвище, ім'я, по батькові) *(підпис)*

« ___ » _____ 2023 р.

Керівник роботи канд. техн. наук, доцент Давиденко Є. О. _____
(посада, прізвище, ім'я, по батькові) *(підпис)*

« ___ » _____ 2023 р.

АНОТАЦІЯ

до кваліфікаційної роботи магістра

«Система керування проєктами на основі засобів штучного інтелекту»

Студентка 608 гр.: Бондаренко Стефанія Віталіївна

Керівник: канд. техн. наук, доцент Давиденко Є. О.

Робота присвячена оптимізації процесу керування проєктами у різних галузях діяльності через розробку системи керування проєктами на основі штучного інтелекту.

Об’єкт роботи: процес керування проєктами, який охоплює всі аспекти планування, виконання, моніторингу та контролю проєктів у різних галузях діяльності.

Предмет роботи: методи та засоби створення системи керування проєктами на основі штучного інтелекту.

Мета: оптимізація процесу керування проєктами в різних галузях діяльності через розробку системи керування проєктами на основі штучного інтелекту.

Під час виконання передбачається вирішення наступних завдань:

– огляд існуючих методологій керування проєктами;

– аналіз сучасних систем керування проєктами та аналіз системи, що розробляється, опис структури та функціоналу;

– огляд методів ШІ та їх застосування в керуванні проєктами;

– формування специфікації вимог до програмного забезпечення;

– опис етапів реалізації проєкту та моделювання системи з графічним представленням моделей даних;

– опис та візуалізація алгоритмів роботи програмного забезпечення;

– побудова архітектури програмного комплексу;

– створення моделей системи з використанням UML-діаграм;

– огляд дизайну вебзастосунку, включаючи UI/UX;

– реалізація програмних компонентів бізнес-логіки, тестування та впровадження вебзастосунку.

Кваліфікаційна робота магістра складається з вступу, чотирьох розділів, висновків та переліку джерел посилання.

У вступі визначається актуальність теми, що приймається за мету та невеликий огляд поставленої задачі, предмет дослідження та об’єкт роботи.

У першому розділі проводиться аналіз існуючих підходів до систем керування проєктами на основі штучного інтелекту, включаючи огляд функціоналу, переваг та недоліків програмного забезпечення. Також в цьому розділі обґрунтовується план виконання завдання. Наступною частиною розділу є формування та опис специфікації вимог до розроблюваного програмного забезпечення.

У другому розділі детально описується процес розробки проєктних рішень, необхідних для виконання специфікації вимог до програмного забезпечення. Що включає моделювання об’єкту та предмету дослідження, а також розробку функціональних та інформаційних моделей програмного забезпечення. У результаті цього процесу складається детальний алгоритм вирішення поставлених завдань.

У третьому розділі розкриваються результати виконаної роботи з конструювання та моделювання програмного забезпечення для системи керування проєктами на основі штучного інтелекту. Це включає вибір технологій, мов програмування, компонентів застосунку, розробку UML-діаграм та опис інтерфейсів програмного забезпечення.

У четвертому розділі демонструється робота з кодування та тестування розробленого програмного забезпечення для системи керування проєктами на основі штучного інтелекту. Також проводиться аналіз отриманих результатів під час тестування програмного забезпечення.

У висновках проводиться аналіз роботи та отриманих результатів.

Кваліфікаційна робота магістра викладена на 93 сторінок, вона містить 4 розділи, 51 ілюстрацій, 18 таблиць, 23 джерел в переліку посилань.

Ключові слова: *система керування проєктами, штучний інтелект, оптимізація процесів, аналіз існуючих систем, специфікація вимог, управління проєктами.*

ABSTRACT

of the Master's Thesis

"Project management system based on artificial intelligence"

Student of group 608m: Bondarenko Stefaniia Vitaliivna

Supervisor: Candidate of Technical Sciences (Ph. D.), Associate Professor

Davydenko Y. O.

The work is devoted to the optimization of the project management process in various fields of activity through the development of a project management system based on artificial intelligence.

Scope of work: the project management process, which covers all aspects of planning, execution, monitoring, and control of projects in various fields of activity.

Subject of work: methods and means of creating a project management system based on artificial intelligence.

Goal: optimization of the project management process in various fields of activity through the development of a project management system based on artificial intelligence.

During implementation, the following tasks are expected to be solved:

– review of existing project management methodologies;

– analysis of modern project management systems and analysis of the system under development, description of structure and functionality;

– overview of AI methods and their application in project management;

– forming a specification of software requirements;

– description of project implementation stages and system modeling with graphical representation of data models;

– description and visualization of software algorithms;

– construction of the architecture of the software complex;

– creation of system models using UML diagrams;

– web application design review, including UI/UX;

– implementation of software components of business logic, testing, and implementation of the web application.

The master’s thesis consists of an introduction, four chapters, conclusions, and a list of reference sources.

The introduction determines the relevance of the topic, which is taken as the goal and a brief overview of the task, the subject of research, and the object of the work.

The first chapter analyzes existing approaches to project management systems based on artificial intelligence, including an overview of the functionality, advantages, and disadvantages of the software. Also, this section substantiates the task execution plan. The next part of the chapter is the formation and description of the specification of requirements for the developed software.

The second section describes in detail the process of developing project solutions necessary to fulfill the specification of software requirements. Which includes the modeling of the object and subject of research, as well as the development of functional and informational software models. As a result of this process, a detailed algorithm for solving the tasks is created.

The third chapter reveals the results of the work done on the design and modeling of software for the project management system based on artificial intelligence. This includes the choice of technologies, programming languages, application components, development

UML diagrams and description of software interfaces.

The fourth chapter demonstrates the work of coding and testing the developed software for the project management system based on artificial intelligence. The results obtained during software testing are also analyzed.

In the conclusions, an analysis of the work and the obtained results is carried out.

The master’s thesis is laid out on 93 pages, it contains 4 chapters, 51 illustrations, 18 tables, and 23 sources in the list of references.

Keywords: *project management system, artificial intelligence, process optimization, analysis of existing systems, specification of requirements, project management.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД	–	база даних
КРМ	–	кваліфікаційна робота магістра
МАІ	–	метод аналізу ієрархій
ОС	–	операційна система
ПЗ	–	програмне забезпечення
ПК	–	персональний комп’ютер
СКБД	–	система керування базами даних
ЦА	–	цільова авдиторія
API	–	application programming interface
CRUD	–	create, read, update, delete
CSS	–	cascading style sheets
ETL	–	Extract, Transform, Load
HTML	–	hypertext markup language
JS	–	JavaScript
JWT	–	JSON Web Tokens
MIT	–	Massachusetts Institute of Technology
MVC	–	model-view-controller
PHP	–	hypertext preprocessor
QR code	–	quick response code
RPA	–	Robotic process automation
SQL	–	structured query language
UI	–	user interface
UML	–	unified modeling language
UX	–	user experience
WAP	–	wireless application protocol
Web app	–	web application

ВСТУП

Розробка системи керування проєктами, що базується на штучному інтелекті, є необхідністю в контексті сучасного бізнес-середовища. Висхідна глобальна конкуренція в кожній сфері вимагає від організацій швидко адаптуватися та оптимізувати свою діяльність. У цьому контексті, керування проєктами виступає як стратегічний інструмент, спрямований на ефективне впровадження проєктів та ініціатив.

Застосування штучного інтелекту в галузі керування проєктами володіє потенціалом змінити парадигму управління проєктами. Наявні підходи та методи часто недостатньо ефективні у вирішенні складних та динамічних викликів, що виникають у життєвому циклі проєктів. Використання штучного інтелекту дозволяє здійснювати точні прогнози, аналізувати великі об’єми даних в режимі реального часу, автоматизувати рутинні завдання та прискорювати процес прийняття управлінських рішень.

Розроблена система, побудована на основі штучного інтелекту, може знаходити застосування в широкому спектрі сфер. Наприклад, в інформаційних технологіях для оптимізації розробки програмного забезпечення, у виробництві для управління виробничими процесами та ланцюгом постачання, в медицині для керування клінічними випробуваннями, у фінансовому секторі для ризик-менеджменту та прогнозування фінансових ринків, і в інших областях.

Об’єктом кваліфікаційної роботи є процес керування проєктами, який охоплює всі аспекти планування, виконання, моніторингу та контролю проєктів у різних галузях діяльності.

Предметом кваліфікаційної роботи є методи та засоби створення системи керування проєктами на основі штучного інтелекту.

Метою роботи є удосконалення процесу керування проєктами в різних галузях діяльності за рахунок розробки системи керування проєктами на основі

штучного інтелекту.

Під час виконання кваліфікаційної роботи магістра **передбачається вирішення наступних завдань**:

– огляд існуючих методологій керування проєктами;

– аналіз сучасних систем керування проєктами, що охоплює вивчення існуючих підходів систем до керування проєктами, їх переваги і недоліки;

– аналіз системи, що розробляється, опис структури та функціоналу;

– огляд методів ШІ та їх застосування в керуванні проєктами;

– формування специфікації вимог до програмного забезпечення;

– опис етапів реалізації проєкту;

– моделювання системи з графічним представленням моделей даних;

– опис та візуалізація алгоритмів роботи програмного забезпечення;

– побудова архітектури програмного комплексу;

– створення моделей системи, які деталізують структуру та функціонал системи керування проєктами, зокрема, використанням UML-діаграм;

– огляд дизайну вебзастосунку, включаючи UI/UX;

– реалізація програмних компонентів бізнес-логіки;

– тестування та впровадження вебзастосунку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз методологій керування проєктами

Методології управління проєктами – це систематичні підходи та набір практик, які використовуються для планування, виконання, контролю та завершення проєктів. Вони встановлюють стандартні процедури, ролі та відповідальності учасників проєкту з метою досягнення його цілей у межах визначених обмежень, таких як обсяг, час та бюджет. Методології управління проєктами важливі з декількох причин:

– Структурують робочий процес. Вони надають чіткий та систематизований план дій для виконання проєкту, що сприяє організації робочих процесів та уникненню хаосу.

– Забезпечують ефективне використання ресурсів. Методології управління проєктами допомагають оптимізувати використання часу, бюджету, матеріальних та людських ресурсів шляхом планування та контролю виконання завдань.

– Зменшують ризики та невдачі. Вони допомагають ідентифікувати потенційні ризики та приймати заходи щодо їх управління, що дозволяє зменшити ймовірність виникнення проблем та невдач.

– Підвищують комунікацію та співпрацю. Методології управління проєктами забезпечують рамки для ефективної комунікації та співпраці між учасниками проєкту, що сприяє зменшенню недорозумінь та конфліктів.

– Забезпечують зручний моніторинг та звітування. Вони надають структуровані механізми для відстеження прогресу проєкту та підготовки звітів про його стан для зацікавлених сторін.

В цілому, методології управління проєктами допомагають забезпечити успішне завершення проєктів шляхом встановлення чітких процедур, контролю та комунікації всіх учасників. Вони є важливим інструментом для досягнення цілей бізнесу та успішного виконання проєктів будь-якої складності. Існує кілька

основних методологій управління проєктами, які найчастіше використовуються:

1. Каскадна (Waterfall) – це традиційний метод, де кожна фаза проєкту виконується послідовно одна за одною. Вимагає чітко визначених вимог на початку проєкту та мінімальних змін під час виконання.

2. Гнучка (Agile) – використовує ітеративний підхід до розробки, де функціонал проєкту розбивається на короткі ітерації, що дозволяє замовнику змінювати вимоги протягом процесу розробки.

3. Scrum – це конкретна методологія Agile, яка використовується для керування розробкою програмного забезпечення. У Scrum команди працюють над функціоналом у короткі ітерації (спринти), під час яких вони планують, виконують та оцінюють свою роботу.

4. Kanban – це ще одна методологія Agile, яка базується на візуальному управлінні завданнями за допомогою дошки з картками. Картки представляють завдання та переміщуються по дошці зі стовпця в стовпець від початку роботи до завершення.

Методології Agile, такі як Scrum та Kanban, найчастіше потребують використання систем керування проєктами на основі засобів штучного інтелекту (вебзастосунок). Основні особливості таких систем включають:

– Планування та призначення завдань: система повинна дозволяти створювати списки завдань, призначати їх командам, встановлювати терміни виконання та пріоритезувати їх.

– Відстеження прогресу: можливість відстежувати прогрес виконання завдань у реальному часі та здійснювати необхідні корективи.

– Спільна робота та комунікація: забезпечення можливості спільної роботи команд та ефективної комунікації між учасниками проєкту.

– Аналіз та звітність: надання засобів для аналізу продуктивності команди, швидкого створення звітів та візуалізація даних про процеси.

– Інтеграція з іншими інструментами: можливість інтеграції з іншими інструментами розробки, такими як системи контролю версій, інструменти для автоматизації тестування тощо.

Функціонал, який може бути корисним для методологій Agile, включає:

– Спрощене планування: швидкий та простий спосіб створення та призначення завдань на основі методології Scrum або Kanban.

– Гнучкі засоби звітності: можливість швидко генерувати звіти про прогрес проєкту, обсяги роботи, завдання, які ще не завершені тощо.

– Спрощений спосіб співпраці та зворотного зв’язку: можливість командам легко спілкуватися, обмінюватися ідеями, висловлювати побажання та отримувати зворотний зв’язок замовника або продуктового власника.

– Швидка зміна пріоритетів та завдань: можливість швидко змінювати пріоритети та завдання відповідно до змін у вимогах замовника чи обставин.

1.2 Огляд застосунків-аналогів управління проєктами

Аналіз застосунків-аналогів для системи керування проєктами на основі засобів штучного інтелекту є ключовою складовою для визначення основних функціональних вимог та для усунення недоліків конкуруючих систем. Конкурентоспроможний застосунок повинен не лише містити аналогічний функціонал інших систем, але і розширювати його для кращого охоплення користувачів чи поліпшення зручності використання. Для цього було обрано аналоги: Jira (табл. 1.1), Trello (табл. 1.2), Asana (табл. 1.3).

Jira

Jira є високофункціональним інструментом для управління проєктами та задачами [1]. Забезпечує зручний та деталізований інтерфейс (рис. 1.1) для планування, відстеження та виконання завдань. Однак, основним недоліком Jira є значний обсяг функціоналу, що може бути варто переваги для великих команд та проєктів, але для менших проєктів може виглядати перебільшеним. При цьому, інтеграція з іншими інструментами та процесами управління проєктами

може вимагати деякої конфігурації для оптимального використання.

Таблиця 1.1 – Опис застосунку

Назва	**Jira**
Архітектура	Система на базі трьохрівневої архітектури.
Виробник	Корпорація Atlassian.
Мова реалізації	Java
Функції	1. Можливості створення та відстеження завдань, багів, історій користувачів. 2. Інтеграція з іншими інструментами для розробки. 3. Гнучкість налаштувань робочого процесу.
Переваги	1. Можливості широкої кастомізації та конфігурації. 2. Велика спільнота користувачів та розширень.
Недоліки	1. Складність для новачків. 2. Високі витрати на ліцензію для великих команд.
Вебсайт	https://www.atlassian.com/software/jira

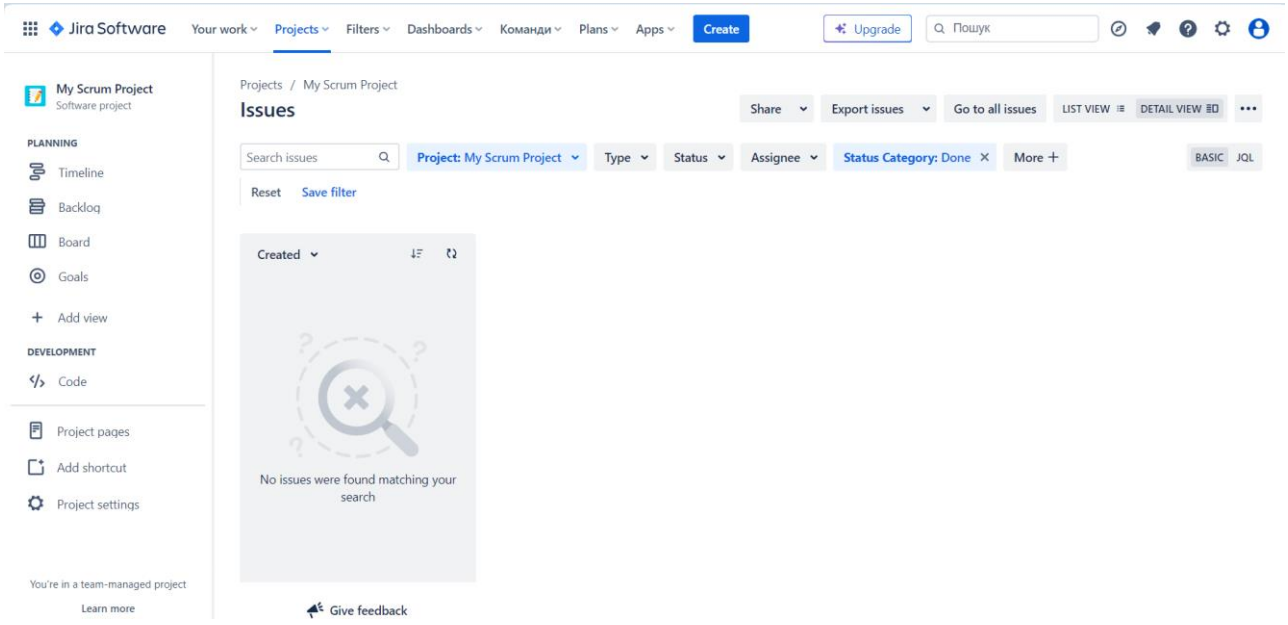

Рисунок 1.1 – Вигляд головної сторінки Jira

Trello

Trello – проста та легка платформа для управління проєктами з інтуїтивним інтерфейсом [2]. Відмінна для менших проєктів, але обмежена функціональністю порівняно з Jira. Забезпечує ефективну комунікацію та відстеження завдань, але може виявитися недостатньою для великих проєктів. Інтеграція з іншими інструментами може вимагати певної конфігурації.

Таблиця 1.2 – Опис застосунку

Назва	**Trello**
Архітектура	Система на базі трьохрівневої архітектури.
Виробник	Корпорації Atlassian та Glitch.
Мова реалізації	Java
Функції	1. Робота з дошками та картками для завдань. 2. Можливість прикріплення файлів, коментування та співпраці.
Переваги	1. Простий та інтуїтивно зрозумілий інтерфейс. 2. Безкоштовний базовий функціонал.
Недоліки	1. Обмежені можливості для складних проєктів. 2. Відсутність розширеного функціоналу для управління завданнями.
Вебсайт	https://trello.com/

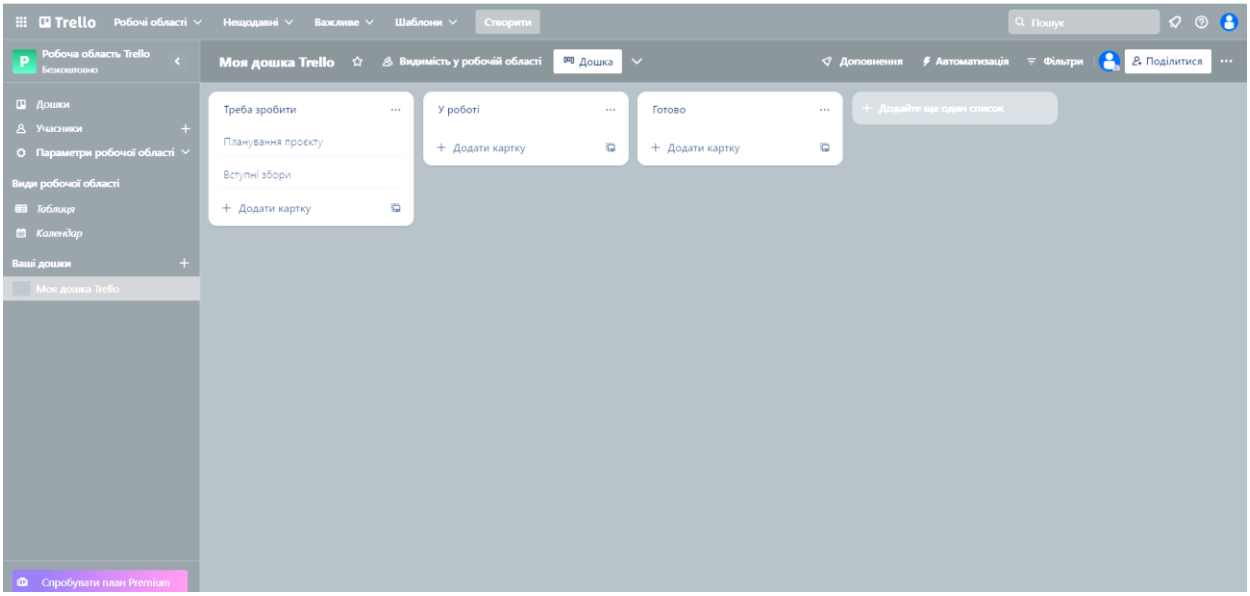

Рисунок 1.2 – Вигляд головної сторінки Trello

Asana

Asana є високоефективним інструментом для управління проєктами, який структурує завдання в точно визначені підсистеми [3]. Шляхом розкриття потенціалу через деталізацію, ця платформа надає можливість точно налаштовувати робочий процес.

Однак для менших проєктів використання всіх можливостей Asana може виглядати трошки складним, оскільки платформа надає більше функціональностей, ніж може знадобитися у подібних випадках.

Таблиця 1.3 – Опис застосунку

Назва	**Asana**
Архітектура	Система на базі трьохрівневої архітектури.
Виробник	Компанія Asana.
Мова реалізації	Python
Функції	1. Створення та відстеження завдань, проєктів, та завдань в межах проєктів. 2. Вбудовані можливості обміну файлами та комунікації.
Переваги	1. Інтуїтивно зрозумілий та легкий у використанні інтерфейс. 2. Багатофункціональність та інтеграція з іншими сервісами.
Недоліки	1. Застосунок стає платним для розширення функціоналу. 2. Обмежені можливості для великих та складних проєктів. 3. Відсутня українська версія сайту.
Вебсайт	https://asana.com

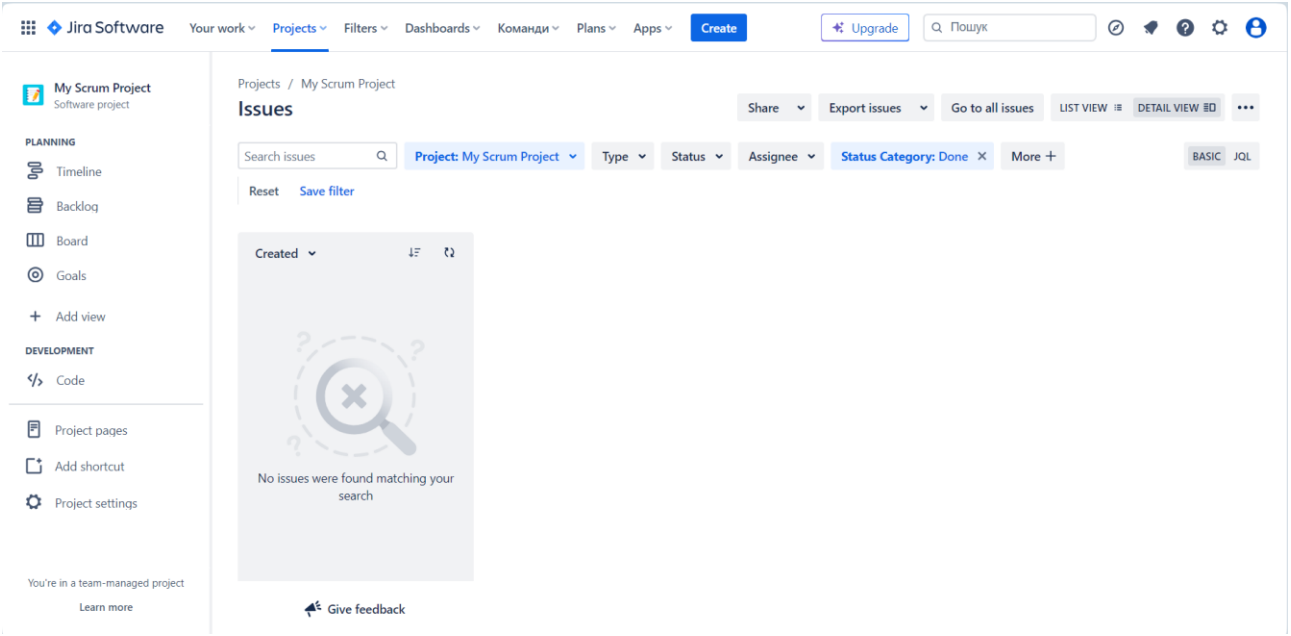

Рисунок 1.3 – Вигляд головної сторінки Asana

Вказані аналоги надають широкий функціонал управління проєктами, проте відрізняються за складністю використання, можливостями кастомізації та вартістю. Аналіз цих систем дозволяє визначити основні пріоритети для розробки системи керування проєктами на основі штучного інтелекту.

1.3 Аналіз системи, що розробляється

Розробка призначена для оптимізації управління проєктами шляхом використання системи штучного інтелекту (ШІ). Аналіз системи включає в себе огляд поточних проблем у сфері управління проєктами та визначення ключових вимог, які мають бути враховані при розробці.

Таблиця 1.4 – Опис системи, що розробляється

Функції	1. Реєстрація користувачів – можливість користувачів реєструвати свої облікові записи в системі. 2. Авторизація користувачів – забезпечення безпечного та контрольованого доступу до системи для авторизованих користувачів. 3. Створення проєкту – можливість користувачів створювати нові проєкти для управління та визначення їх ключових характеристик.

Кінець таблиці 1.4

Функції	4. Аналіз та візуалізація даних – надання інструментів для аналізу та візуалізації даних про стан проєктів з використанням штучного інтелекту. 5. Завдання та етапи проєкту – визначення конкретних завдань та етапів для кожного проєкту з можливістю їхнього відстеження. 6. Планування ресурсів – автоматизоване планування та розподіл ресурсів для ефективного виконання завдань у проєкті. 7. Спільна робота та комунікація – забезпечення можливостей спільної роботи та комунікації між учасниками проєкту. 8. Автоматизована звітність – створення автоматизованих звітів та аналітичної інформації на основі зібраних даних.
Користувачі	1. адміністратор; 2. керівництво проєкту; 3. проєктний менеджер; 4. розробники та учасники проєкту.
Сценарії роботи	1. Керівник проєкту створює новий проєкт та назначає учасників. 2. Розробники виконують завдання та відмічають прогрес. 3. Проєктний менеджер аналізує дані та приймає рішення на основі розуміння ситуації у системі. 4. Комунікація та обмін ідеями між учасниками проєкту через систему. 5. Автоматизоване створення та відправлення звітів за визначеними періодами. 6. Система надає рекомендації щодо оптимізації робочих процесів на основі аналізу даних.

Першочерговим завданням є визначення потреб та очікувань користувачів системи, а також виявлення недоліків у існуючих методах управління проєктами. Додатково проводиться аналіз сучасних тенденцій у сфері ШІ та їх застосування в управлінні проєктами З метою ефективного впровадження системи керування проєктами на базі ШІ, виконується оцінка потенційних переваг і ризиків, а також врахування аспектів масштабованості та інтеграції з існуючими ІТ-структурами компанії. В результаті аналізу визначаються основні цілі та завдання системи, які

враховують потреби користувачів та сприяють підвищенню продуктивності управління проєктами.

1.4 Аналіз методів ШІ в контексті керування проєктами

Машинне Навчання (Machine Learning)

Машинне навчання в керуванні проєктами використовується для автоматизації прийняття рішень на основі аналізу великих обсягів даних. Перевагами є можливість виявлення складних патернів та предиктивний аналіз. Однак, для ефективного застосування потрібні великі обсяги даних, а недоліками є потреба в складних алгоритмах та складність інтерпретації результатів.

Аналіз даних та візуалізація (Data Analysis and Visualization)

Метод аналізу даних та візуалізації є потужним інструментом для управління проєктами. Забезпечуючи інтерактивні засоби для аналізу, Даний метод надає змогу керівникам проєктів отримувати чіткий образ ситуації. Перевагами є реалізація швидкого та глибокого аналізу даних, а також зручність сприйняття інформації. Але, для ефективності, потрібна добре підготовлена БД.

Роботизований процес ввтоматизації (RPA)

RPA спрямований на автоматизацію рутинних операцій. Він забезпечує значну економію часу та ресурсів. Перевагами є швидка імплементація та ефективність у виконанні монотонних завдань. Однак, RPA не завжди ефективний для складних операцій та вимагає регулярного оновлення.Розвиток сучасних технологій в галузі штучного інтелекту привів до розширення можливостей управління проєктами. Один із важливих аспектів цього напрямку – використання різних методів штучного інтелекту для покращення процесів управління та прийняття рішень. У цьому контексті, аналіз методів, таких як машинне навчання, аналіз даних та візуалізація, і роботизований процес автоматизації (RPA) надає можливість зрозуміти їхні переваги та обмеження. Порівняльна таблиця методів штучного інтелекту в керуванні проєктами (табл. 1.5) надає огляд ключових характеристик кожного

підходу. Що спрощує вибір оптимального методу для конкретного проєкту, враховуючи його потреби та характеристики.

Таблиця 1.5 – Порівняльна таблиця методів ШІ в керуванні проєктами

Критерії	**Машинне навчання**	**Аналіз даних та візуалізація**	**Роботизований процес автоматизації**
Можливість предикції	Так	Ні	Ні
Кількість потрібних паних	Велика	Середня	Мала
Складність реалізації	Висока	Середня	Низька
Інтерпретація результатів	Складна	Легка	Легка
Час розгортання	Довгий	Короткий	Короткий
Ефективність в рутинних завданнях	Середня	Найвища	Висока

Хоча всі три методи ШІ мають свої переваги та недоліки, аналіз даних та візуалізація виявляється найбільш ефективним для управління проєктами [4]. Цей метод не лише забезпечує глибокий аналіз даних, але й надає зручний та зрозумілий інструмент для прийняття рішень, перевершуючи інші методи за значущими критеріями.

1.5 Специфікація вимог системи

ПРИЗНАЧЕННЯ ТА МЕЖІ ПРОЄКТУ

Призначення системи (застосунку), для якої розробляється програмне забезпечення

Система спрямована на автоматизацію процесів планування, виконання та моніторингу проєктів, забезпечуючи ефективне управління ресурсами та використання аналітичних інструментів для прийняття обґрунтованих рішень.

Погодження, що ухвалені в програмній документації

Ухвалені в програмній документації засади включають в себе визначення основних функціональностей системи, методів ШІ для аналізу та візуалізації даних, а також вимог до інтерфейсу користувача. Погоджено механізми автоматизованого керування проєктами та використання розроблених

аналітичних інструментів.

Межі проєкту ПЗ

Крайня дата завершення роботи над ПЗ – 10.05.2024 р.

ЗАГАЛЬНИЙ ОПИС

Сфера застосування

Сфера застосування розробленої системи охоплює управління проєктами в різних галузях, включаючи інформаційні технології, будівництво, маркетинг та інші сектори. Система спрямована на поліпшення ефективності та точності управлінських рішень завдяки використанню штучного інтелекту та аналітичних методів.

Характеристики користувачів

Користувачі системи включають менеджерів проєктів, команди розробників та аналітиків. Їхні основні потреби – ефективне планування, моніторинг та аналіз проєктів, а також доступ до інтуїтивно зрозумілих інструментів для взаємодії з системою. Для взаємодії та роботи з системою потребують: персональні комп’ютери або ноутбуки з веббраузером та доступом до Інтернету, через що важливо, щоб система була оптимізована для різних типів пристроїв та забезпечувала зручний інтерфейс користувача.

Загальна структура і склад системи

Загальна структура і склад системи включає наступні компоненти: сервер для обробки запитів, базу даних для зберігання інформації про проєкти, вебзастосунок для взаємодії з користувачами та API для інтеграції з іншими системами.

Загальні обмеження

Загальні обмеження для роботи з ПЗ включають наявність високошвидкісного інтернет-з’єднання, сумісність з сучасними веббраузерами.

ФУНКЦІЇ СИСТЕМИ КЕРУВАННЯ ПРОЄКТАМИ НА ОСНОВІ ЗАСОБІВ ШТУЧНОГО ІНТЕЛЕКТУ

Функція створення звітів про стан проєкту

Опис функції

Забезпечує можливість генерації детальних звітів щодо стану проєкту, включаючи інформацію про завдання, використання ресурсів та інші ключові показники.

Вхідна і вихідна інформація

Вхідна інформація – дані про стан проєкту;

вихідна інформація – сформовані звіти та статистика.

Функціональні вимоги

Забезпечення можливості користувачам генерувати різноманітні звіти для оцінки та аналізу продуктивності проєкту.

Функція візуалізації прогресу проєкту

Опис функції

Надає засоби візуалізації даних щодо прогресу виконання проєкту, використовуючи методи штучного інтелекту для аналізу та підсумкового представлення інформації.

Вхідна і вихідна інформація

Вхідна інформація – дані про стан завдань та проєкту;

вихідна інформація – графічні візуалізації прогресу.

Функціональні вимоги

Застосування алгоритмів візуалізації та аналізу даних для надання користувачам зрозумілої та інформативної картини стану проєкту.

Функція створення та збереження завдань проєкту

Опис функції

Надає можливість користувачам легко створювати нові завдання для проєкту та зберігати їхні основні параметри, такі як назва, термін виконання та опис.

Вхідна і вихідна інформація

Вхідна інформація – дані про нові завдання;

вихідна інформація – збережені параметри завдань.

Функціональні вимоги

Забезпечення можливості швидкого та зручного створення нових завдань і їхнього збереження у системі.

ВИМОГИ ДО ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

Джерела і зміст вхідної інформації (даних)

В даній системі керування проєктами на основі штучного інтелекту, основним джерелом вхідної інформації є: користувачі – мають можливість вручну вводити дані щодо параметрів проєктів, визначення завдань, термінів виконання та інших важливих деталей; історія проєктів – попередні дані проєктів, включаючи завдання, терміни виконання та результати, можуть бути використані для аналізу та покращення стратегій управління проєктами.

Нормативно-довідкова інформація (класифікатори, довідники тощо)

Вимоги до даного пункту відсутні.

Вимоги до способів організації, збереження та ведення інформації

Всі дані взаємодії зберігаються та обробляються за допомогою RESTful API для забезпечення стандартизованого обміну даними між різними компонентами системи. Для збереження та управління інформацією використовується база даних PostgreSQL, що забезпечує надійність, швидкодію та підтримку транзакцій для ефективного ведення журналу подій та забезпечення консистентності даних.

ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

Вимоги до технічного забезпечення включають підтримку операційних систем Windows 10, MacOS або Linux. Для ефективної обробки даних та виконання алгоритмів штучного інтелекту потрібен чотирьохядерний процесор з підтримкою SSE2. Мінімальний обсяг оперативної пам’яті повинен становити 8 ГБ. Система має підтримувати сучасні веббраузери (Google Chrome, Mozilla Firefox, Safari) для відображення та взаємодії з вебзастосунком. Стабільне інтернет-з’єднання необхідне для роботи з вебсервісами та спільної роботи над проєктами в реальному часі. Роздільна здатність екрану повинна бути не менше

1280x800 пікселів для зручного відображення інтерфейсу та інформації.

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура програмної системи

Архітектура складається з клієнтської частини, серверної частини та БД.

Системне програмне забезпечення

Розробка front-end частини реалізована з використанням бібліотек та інструментів, таких як React, React Query, React Router, React Redux, Redux Toolkit та Tailwind CSS. Для back-end частини використано мову програмування Node.JS, а також фреймворки Express, CORS. Використання штучного інтелекту здійснено через OpenAI, яке використовується для конкретних функціональних можливостей системи. Обмін даними між клієнтом та сервером реалізований за допомогою RESTful API, забезпечуючи стандартизований та ефективний обмін інформацією. У якості системи керування базами даних (СКБД) обрано PostgreSQL, що гарантує надійність, швидкодію та підтримку транзакцій для ефективного зберігання та управління даними.

Мережеве програмне забезпечення

Для створення мережевого програмного забезпечення використано операційну систему Windows 11. У якості редактора коду вибрано Atom для зручності розробки та редагування програмного коду. Для перегляду вебсторінок та забезпечення зручного тестування використовуються веббраузери Opera та Google Chrome. Додатково встановлено розширення для Google Chrome, такі як React DevTools та відлагоджувач Node.js, що дозволяють використовувати інструменти для розробки та відлагодження на платформі Node.js прямо в браузері.

Програмне забезпечення ведення інформаційної бази

Управління даними здійснюється через систему керування базами даних PostgreSQL, яка забезпечує високий рівень надійності та швидкодії. Використання PostgreSQL дозволяє ефективно виконувати операції зчитування, запису, оновлення та видалення даних (CRUD-операції) у контексті

інформаційної бази застосунку. Завдяки цьому, забезпечується стійкість та консистентність даних, а також ефективність взаємодії з інформаційною базою.

Мова і технологія розробки ПЗ

Для розробки програмного забезпечення використовується мова програмування JavaScript та фреймворк Node.js з використанням Express. JavaScript є основною мовою розробки, що забезпечує гнучкість та широкі можливості для вивчення та впровадження нових функцій. Фреймворк Node.js дозволяє ефективно створювати серверні застосунки, а використання Express спрощує розробку вебдодатків та API, забезпечуючи стандартизований та ефективний підхід до створення серверної частини програмного забезпечення.

ВИМОГИ ДО ЗОВНІШНІХ ІНТЕРФЕЙСІВ

Інтерфейс користувача

Система повинна мати інтуїтивно зрозумілий інтерфейс, орієнтований на забезпечення зручності користувача. Вебзастосунок має включати дві основні частини: пошукову панель і основний контент. Пошукова панель, розташована зліва, має фільтри для ефективного пошуку. Основний контент динамічно змінюється відповідно до встановлених фільтрів.

Апаратний інтерфейс

Система має бути доступною на різних пристроях користувача, таких як ПК, смартфони та планшети. Вебзастосунок повинен адаптуватися до екранного розміру та можливостей кожного пристрою.

Програмний інтерфейс

Express – це фреймворк для розробки вебзастосунків на Node.js. Він надає виразний синтаксис та ефективні засоби для обробки HTTP-запитів, маршрутизації, підтримки middleware і створення REST-контролерів. Завдяки цьому, Express є потужним інструментом для швидкої та зручної розробки серверної частини вебзастосунків.

Комунікаційний протокол

Система використовує мережеві протоколи WAP для безпровідної

передачі даних та TCP/IP для забезпечення надійності та ефективності обміну інформацією.

ВЛАСТИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Доступність

Система повинна реалізувати механізм доступу, що базується на принципах вебтехнологій, забезпечуючи сумісність з усіма сучасними веббраузерами та можливість взаємодії з користувачами через HTTP-протокол. Вимагається наявність стандартизованого апаратного та мережевого інтерфейсу.

Супроводжуваність

Розроблена система повинна бути автономною та включати в себе елементи самодіагностики та системи логування для моніторингу її функціонування. Необхідно врахувати можливість визначення та виправлення невідомих станів системи без додаткового втручання оператора.

Переносимість

Розроблена система повинна відповідати стандартам кросплатформенної розробки та мати можливість запуску на різних операційних системах, зокрема Windows (версія 10 і вище), а також мобільних платформах iOS і Android (версія 7.1 і вище).

Продуктивність

Швидкість виконання запитів системи повинна бути оптимізована для мінімізації часу відповіді. Вимагається вивчення та удосконалення алгоритмів обробки даних та взаємодії з базою даних для досягнення максимальної продуктивності.

Надійність

Система повинна враховувати принципи криптографічної безпеки для забезпечення конфіденційності та цілісності користувальницької інформації. Вимагається введення системи авторизації та обмеження доступу, зокрема в межах особистих облікових записів користувачів.

Безпека

Аутентифікація користувачів реалізована з використанням принципу токенізації, що базується на стандартах безпеки та передових технологіях. У системі використовуються JWT (JSON Web Tokens) для генерації та обміну токенами. JWT забезпечує ефективний та безпечний механізм автентифікації, дозволяючи передавати інформацію про користувача між різними компонентами системи без необхідності зберігання сесій на сервері. Це підвищує безпеку взаємодії та захищає від потенційних атак, пов’язаних зі зломом та витоком конфіденційної інформації.

Висновки до розділу 1

В першому розділі кваліфікаційної роботи магістра проведено аналіз наявних систем керування проєктами, включаючи критичний огляд парадигм та методологій в їх реалізації. Здійснено огляд й аналіз переваг та недоліків різних підходів до керування проєктами, а також проаналізовано функціональні та нефункціональні аспекти наявних рішень.

Вивчено структуру та функціонал системи, що розробляється, враховуючи архітектурні аспекти та механізми взаємодії компонентів. Це охоплює визначення основних користувачів, її ключовий функціонал та деталізацію сценаріїв взаємодії.

В контексті першого розділу також здійснено науковий огляд методів штучного інтелекту та їх реалізацію в області керування проєктами. Проведено аналіз застосування ШІ для аналізу та візуалізації проєктних даних, включаючи розгляд основних алгоритмів та підходів.

Крім того, розділ містить специфікацію вимог до програмного забезпечення, що включає докладний опис функціональності, вимог до безпеки, а також визначення метрик ефективності та надійності системи. Розглянуто вимоги до архітектури системи, стандартів взаємодії та технічних характеристик.

2 МОДЕЛЮВАННЯ СИСТЕМИ КЕРУВАННЯ ПРОЄКТАМИ

2.1 Моделі системи

ER-діаграма (Entity-Relationship) відображає модель сутностей системи керування проєктами (рис. 2.1). Діаграма сутностей і зв’язків системи керування проєктами демонструє всі інструменти таблиць бази даних та зв’язки між «Співробітником», «Завданням», «Проєктом», «Помилкою» тощо. Діаграма використовує структуровані дані для визначення відносин між групами структурованих даних функціоналів системи керування проєктами.

Основні сутності та їх атрибути Системи Управління Проєктами:

- сутність проєкту: Атрибути «Project» – project_id, project_developer_id, project_tester_id, project_name, project_assign, project_last_date, project_type, project_description;
- сутність співробітника: Атрибути «Employee» – employee_id, employee_name, employee_mobile, employee_email, employee_username, employee_password, employee_address;
- сутність завдання: Атрибути «Task» – task_id, task_employee_id, task_name, task_type, task_description;
- сутність помилки: Атрибути «Bug» – bug_id, bug_developer_id, bug_tester_id, bug_title, bug_type, bug_description;
- сутність заявки (коментаря): Атрибути «Ticket» – ticket_id, ticket_bug_id, ticket_type, ticket_date, ticket_description.

Опис бази даних Системи Управління Проєктами:

- деталі проєкту зберігаються в таблицях «Project» разом із всіма іншими таблицями;
- кожна сутність містить первинний та унікальний ключі;
- сутність «Bug» пов’язані з «Project», «Employee» за допомогою зовнішнього ключа;
- доступні відносини один до одного та один до багатьох між «Bug»,

«Task», «Ticket», «Project»;

– всі сутності: «Bug», «Ticket», «Project» – нормалізовані для зменшення дублювання записів;

– реалізовано індексацію для кожної таблиці в таблицях системи керування проєктами для швидкого виконання запитів.

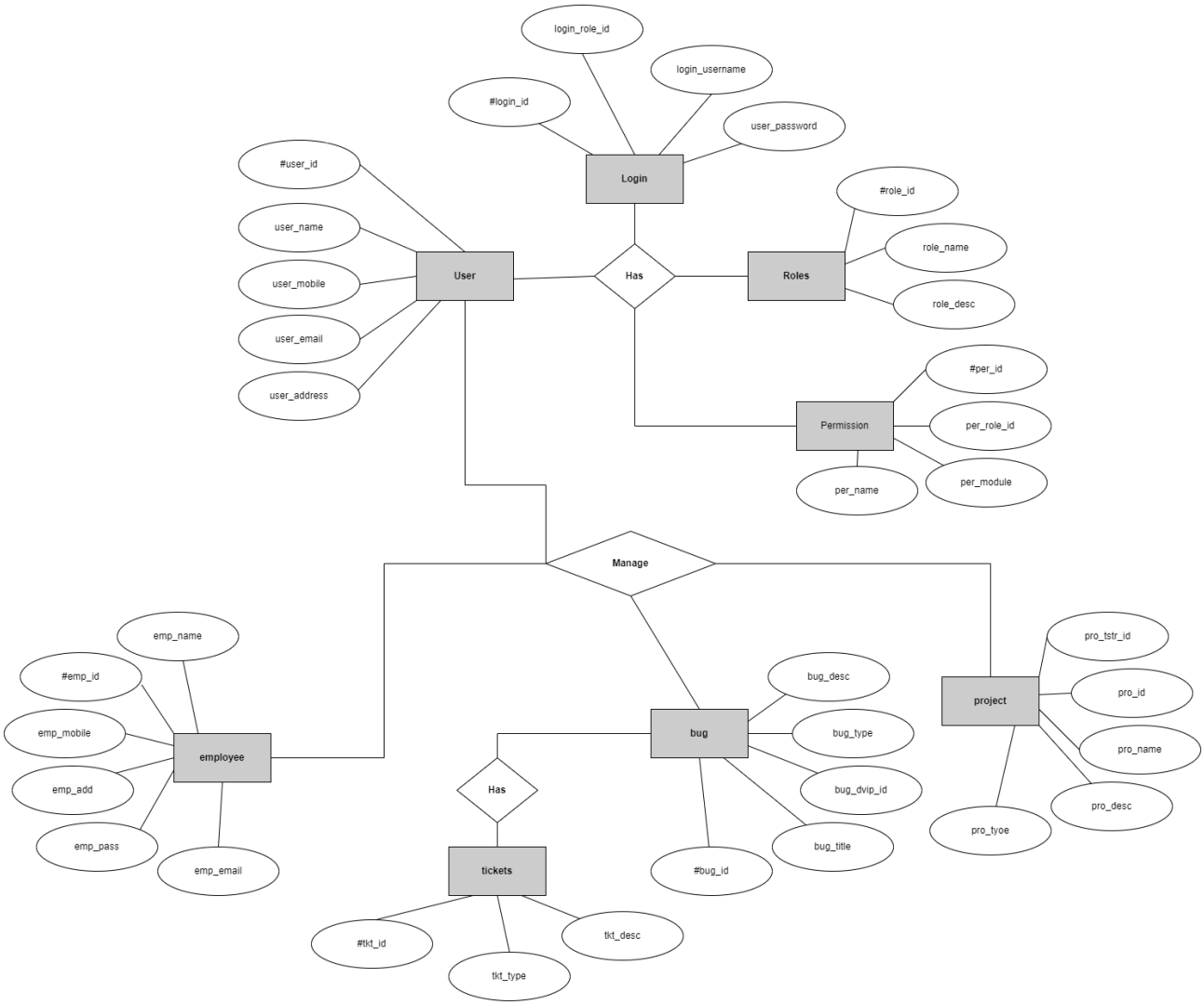

Рисунок 2.1 – ER-діаграма системи, що розробляється

Розроблена діаграма сутностей і зв’язків використовується для візуалізації та проєктування бази даних. Її створення має на меті визначення сутностей та їх взаємозв’язків, сприяючи правильному проєктуванню бази даних для відображення реальних бізнес-процесів. Діаграма допомагає уточнити вимоги до бази даних, визначає первинні та зовнішні ключі, і забезпечує цілісність даних.

Під час розробки ER-діаграма використовується для визначення таблиць бази даних, взаємодії замовників, оптимізації запитів, а також при внесенні змін та розширенні функціональності системи. Діаграма ER слугує засобом комунікації та навчання для команди розробників, допомагаючи розуміти структуру та взаємодії в системі. Наприклад, розроблена ER-діаграма дозволяє визначити, як інформація про проєкти, завдання, співробітників взаємодіє, що полегшує створення реальної бази даних для системи.

DFD (Data Flow Diagram), або діаграма потоку даних, є інструментом моделювання, який використовується для візуалізації та аналізу обміну даними та процесів у системі. Дозволяє зображати, як дані переміщаються в системі, як вони обробляються та які процеси з ними пов’язані. DFD використовується для створення абстрактних моделей систем, спрощуючи розуміння їх структури та функцій.

Зазвичай DFD використовується для наступних цілей:

– аналіз бізнес-процесів: DFD дозволяє аналізувати обмін даними та послідовності операцій в бізнес-процесах.

– документація систем: Вона служить для створення абстрактної моделі системи, що полегшує її розуміння та документування.

– проєктування систем: DFD допомагає визначити ключові компоненти та взаємозв’язки між ними, що полегшує розробку та проєктування систем.

– оптимізація процесів: Вона вказує на можливість оптимізації потоків даних та процесів в системі для підвищення ефективності.

– взаємодія з замовниками: DFD може використовуватися для узгодження та зрозуміння вимог замовників щодо функціональності системи.

Узагальнюючи, DFD – це інструмент візуалізації та аналізу системних процесів, що сприяє зрозумінню їх функцій та структури для подальшого вдосконалення та розвитку.

Створення та використання діаграм потоку даних (DFD) від 0-го до 2-го рівнів в моделюванні системи дозволяє отримати загальний огляд та подробиці

щодо обміну даними та процесів у системі. DFD є ефективним інструментом для аналізу та документування функцій системи, визначення взаємодій між компонентами та зовнішніми сутностями. Нульовий рівень дозволяє отримати високорівневий огляд, тоді як рівні 1 (рис. 2.3) і 2 (рис. 2.4) деталізують процеси та взаємозв'язки. Різниця в рівнях полягає в ступені деталізації та конкретизації взаємодій, що дозволяє отримати повнішу картину функціонування системи на різних рівнях деталізації.

Діаграма потоку даних нульового рівня (DFD 0) системи керування проєктами є базовим оглядом високорівневого процесу керування проєктами, призначений для аналізу або моделювання. На діаграмі DFD 0 (рис. 2.2) продемонстровано загальний високорівневий потік процесу.

Рисунок 2.2 – DFD нульового рівня

Перший рівень DFD (1-й рівень) системи управління проєктами показує, як система розділена на підсистеми (процеси), кожна з яких працює з одним або кількома потоками даних до чи від зовнішнього агента, і які разом забезпечують всю функціональність системи управління проєктами в цілому. Також визначаються внутрішні сховища даних, які повинні бути присутніми для того, щоб система управління проєктами могла виконувати свою роботу, і показує

потік даних між різними частинами працівника, проєкту, заявки, табеля робочого часу та помилки в системі. DFD рівня 1 надає більш детальний розклад частин першого рівня DFD. Виокремлено основні функціональності системи управління проєктами.

Основні сутності та вивід Першого рівня DFD (1-й рівень DFD):

- обробка записів про працівників та генерація звіту про всіх працівників;
- обробка записів про проєкти та генерація звіту про всі проєкти;
- обробка записів про задачі та генерація звіту про всі задачі;
- обробка записів про помилки та генерація звіту про всі помилки;
- обробка записів про коментарями, заявками та генерація звіту про всі заявки;
- обробка записів про табель робочого часу (розклад) та генерація звіту про весь табель робочого часу.

Рисунок 2.3 – DFD першого рівня

DFD рівня 2 виходить розкриває деталі системи управління проєктами. Для досягнення необхідного рівня деталей щодо функціонування системи управління проєктами може бути потрібно більше функцій управління проєктами. Перший рівень DFD (1-й рівень) системи управління проєктами показує, як система розділена на підсистеми (процеси).

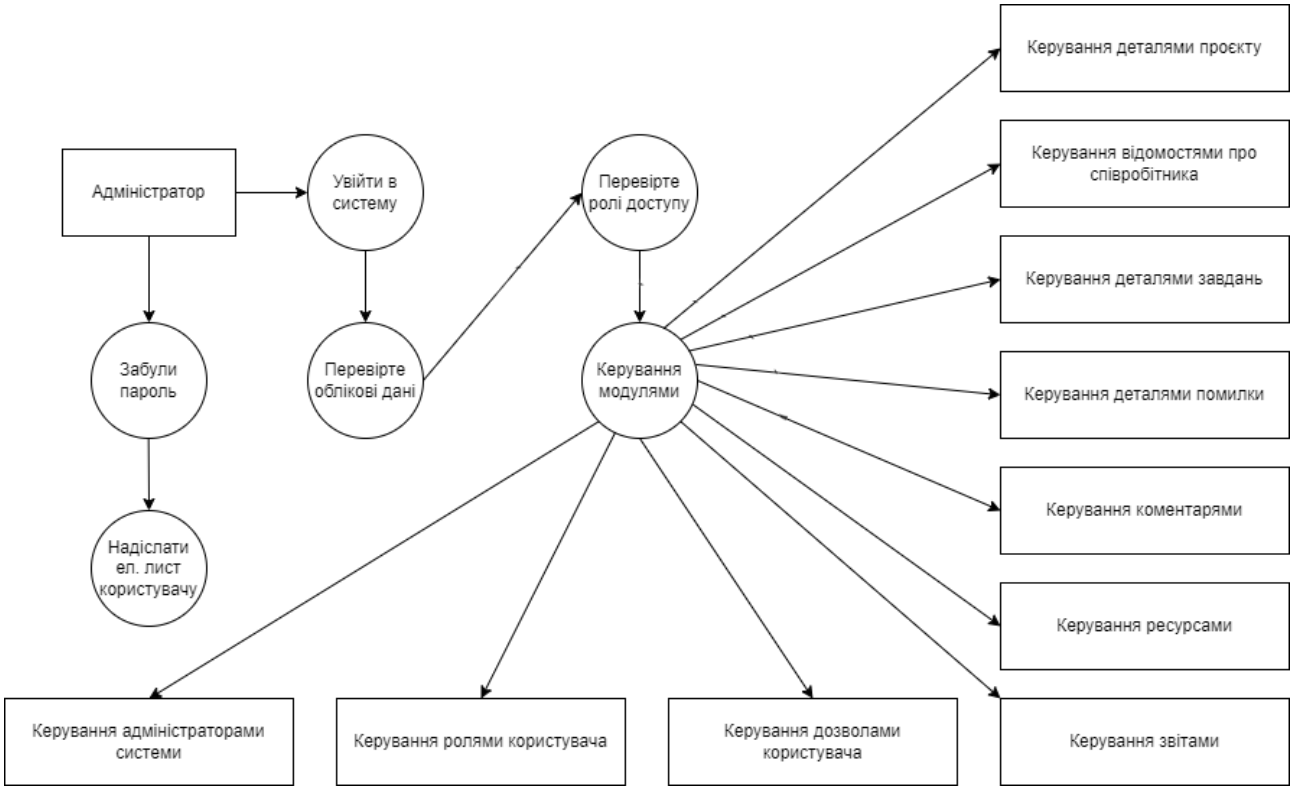

Рисунок 2.4 – DFD другого рівня

Функціональності системи управління проєктами на низькому рівні для адміністратора [5]:

- вхід до системи та керування всіма функціональними можливостями системи управління проєктами;
- має можливість додавати, редагувати, видаляти та переглядати записи про працівників, проєкти, помилки, табель робочого часу;
- має можливість керувати всіма деталями щодо зарплати працівників, завдань та заявок;
- може генерувати звіти про працівників, зарплату працівників, проєкти, завдання, помилки, заявки;

– має можливість шукати деталі щодо зарплати працівників, помилок, заявок;

– може застосовувати різні рівні фільтрів до звіту про працівників, завдання, помилки;

– має можливість відстежувати детальну інформацію щодо зарплати працівників, проєктів, завдань, помилок.

Діаграма IDEF0 (Integrated Definition for Function Modeling) є методологією моделювання бізнес-процесів та функцій у системах. Вона використовується для аналізу та документування функціональних аспектів організаційних систем. IDEF0 дозволяє визначити, які функції виконуються в системі, як вони взаємодіють між собою та як система реагує на вхідні дані.

Основна мета діаграми IDEF0 – це надати структурований погляд на функції системи та їх взаємодію. Діаграма IDEF0 складається з блоків, які представляють функції, та стрілок, які вказують на взаємодію між цими функціями.

У контексті розробки «Системи керування проєктами на основі засобів штучного інтелекту» діаграма IDEF0 може бути використана для наступних цілей:

– аналіз бізнес-процесів: діаграма IDEF0 дозволяє детально проаналізувати бізнес-процеси, які входять в систему керування проєктами. Це може допомогти ідентифікувати ефективність та слабкі сторони процесів;

– визначення функціональності: діаграма IDEF0 може допомогти визначити різні функціональні елементи системи керування проєктами та їх взаємодію;

– оптимізація процесів: аналіз діаграми IDEF0 може вказати на можливість оптимізації функціональних процесів для підвищення ефективності системи керування проєктами;

– взаємодія зі штучним інтелектом: якщо система керування проєктами використовує засоби штучного інтелекту, діаграма IDEF0 може допомогти

визначити, як саме ці технології взаємодіють з іншими функціями системи та як вони впливають на процеси.

Загалом, діаграма IDEF0 (рис. 2.5) є інструментом, який сприяє визначенню, аналізу та оптимізації функціональності в складних системах, що може бути корисним при розробці та удосконаленні систем керування проєктами, особливо при використанні штучного інтелекту.

Блок A0 «Аналіз та візуалізація даних» в діаграмі IDEF0 є вищим рівнем абстракції, який описує загальний процес аналізу та візуалізації даних у системі. Цей блок об’єднує підблоки A1, A2, A3 та A4, визначаючи загальні кроки та функції для ефективного та цільового аналізу даних. Функції блоку включають визначення стратегій аналізу, керування взаємодією між підблоками та встановлення стандартів візуалізації.

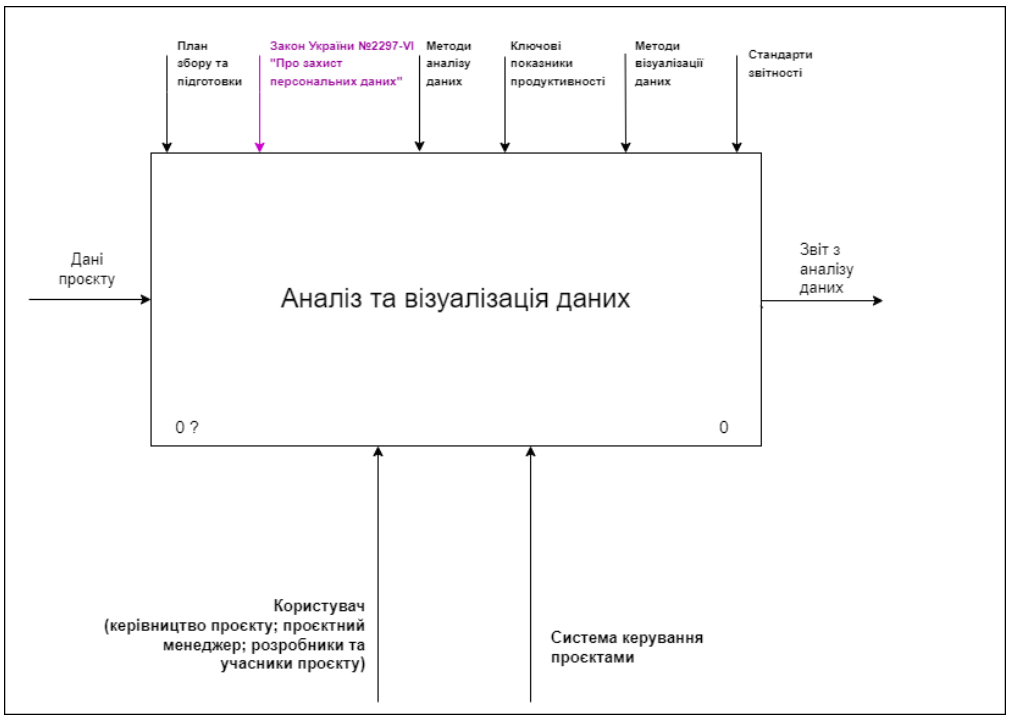

Рисунок 2.6 – Діаграма IDEF0, вузол A-0 (загальний блок)

Блок A0 взаємодіє з іншими блоками для отримання та передачі даних, забезпечуючи високорівневий огляд над аналітичним процесом. Його результати визначають основні етапи та функції, використовувані для аналізу та візуалізації даних.

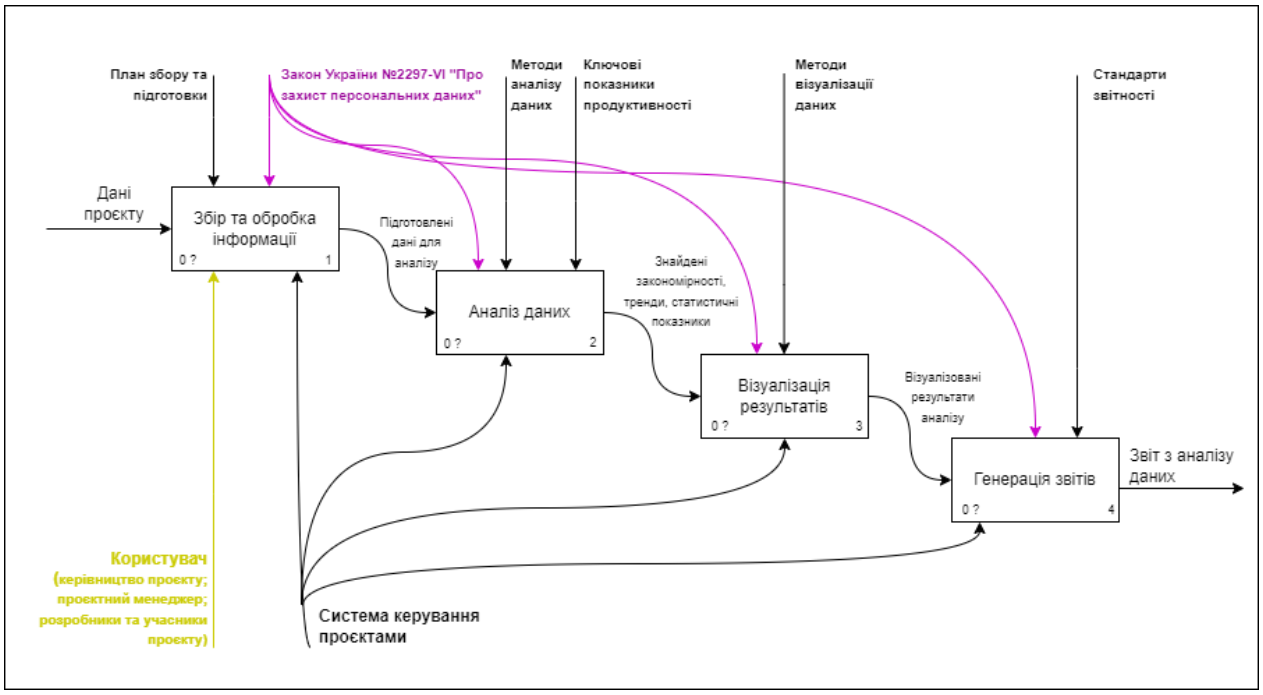

Рисунок 2.6 – Діаграма IDEF0, вузол A-0 (підблоки)

Діаграма IDEF0 для блоку A0 «Аналіз та візуалізація даних» з підблоками A1, A2, A3, та A4 може бути створена для детального аналізу процесів аналізу та візуалізації даних.

Таблиця 2.1 – Опис підблоків

Назва	**Опис**	**Функції**
A1: Збір та обробка даних	Відповідає за збір та первинну обробку даних, які подаються до системи. Це може включати в себе автоматизовані процеси отримання даних від різних джерел, таких як бази даних, зовнішні API, або інші джерела даних.	– збір та агрегація даних; – очищення та перевірка достовірності даних; – синхронізація та інтеграція даних з різних джерел.

Кінець таблиці 2.1

Назва	Опис	Функції
A2: Аналіз даних	Відповідає за виконання різноманітних аналітичних операцій над даними з метою виявлення збігів, трендів та кореляцій. Він може використовувати різні алгоритми аналізу даних та статистичні методи для отримання відомостей.	– використання аналітичних методів та алгоритмів; – виявлення збігів та аномалій у даних; – розробка моделей для прогнозування та оптимізації.
A3: Візуалізація ресурсів	Відповідає за представлення результатів аналізу даних у вигляді візуальної інформації, яка допомагає користувачам краще розуміти дані. Це може включати створення діаграм, графіків, карт та інших візуальних елементів.	– створення візуальних репрезентацій даних; – інтерактивність та можливість налаштування візуалізацій; – підтримка різних типів графіків для різноманітності інформації.
A4: Генерація звітів	Відповідає за автоматизовану генерацію детальних звітів та аналітичних документів на основі оброблених даних. Звіти можуть включати ключові відомості, графіки та рекомендації.	– автоматизована підготовка звітів на основі результатів аналізу. – персоналізація та налаштування звітів для різних аудиторій. – розподілення звітів відповідно до визначених графіків та умов.

Розроблена діаграма IDEF0 є корисним інструментом для моделювання та аналізу процесу «Аналіз та візуалізація даних». З її допомогою можна чітко визначити ролі та взаємозв'язки між підблоками, такими як збір та обробка даних, аналіз даних, візуалізація ресурсів і генерація звітів [6]. Діаграма дозволяє легко розуміти структуру процесу, його функціональність та взаємодію компонентів. Це важливо для ефективного управління та вдосконалення системи аналізу та візуалізації даних, допомагаючи забезпечити оптимальність та високу якість аналітичних результатів.

2.2 Створення Use case

Use Case (рис. 2.7) або сценарій користування представляє документований опис послідовності дій, які визначають взаємодію користувача з програмним застосунком з метою виконання конкретної операції або досягнення певного функціонального результату [7]. У такому документі детально описується, як користувач взаємодіє із системою, враховуючи можливості та обмеження додатку. Важливо зазначити, що такі сценарії зазвичай фокусуються на тому, що система виконує, а не на тому, як саме це відбувається.

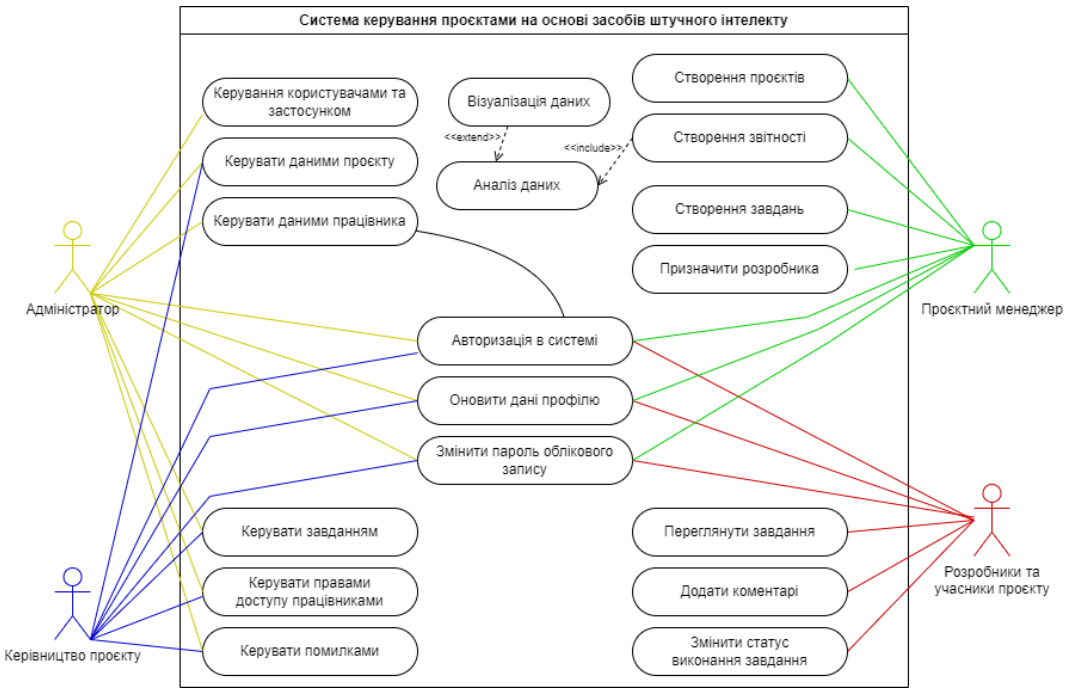

Рисунок 2.7 – Діаграма прецедентів системи, що розробляється

Короткий Use Case:

Користувач має потребу в оптимізації управління проєктами та аналізу даних. Додає проєкт до системи та використовує опції ШІ для автоматизації завдань та прогнозування ризиків. Звертається до системи для отримання рекомендацій та візуалізації даних. Має можливість шукати інформацію у вже підготованих звітах та аналітичних матеріалах. Відстежує прогрес та ефективність проєкту, взаємодіючи з системою.

Поверхневий Use Case:

Головний сценарій (успішний):

1. користувач має потребу в управлінні новим проєктом та використовує систему для його додавання;
2. використовуючи функціонал ШІ, користувач налаштовує автоматичні завдання та отримує прогноз ризиків;
3. звертається до системи для отримання рекомендацій щодо оптимізації процесів та аналізу даних;
4. використовує можливості візуалізації для кращого розуміння стану проєкту;
5. здійснює пошук інформації у вже підготованих звітах та аналітичних матеріалах системи;
6. відстежує прогрес проєкту та оцінює його ефективність.

Альтернативний сценарій:

1. рекомендації системи не допомагають користувачеві у вирішенні конкретних проблем проєкту, потрібні додаткові консультації;
2. користувач не встигає вирішити завдання у визначений час, необхідно збільшити терміни проєкту;
3. користувач вирішує закрити проєкт, не завершивши всі заплановані етапи;
4. користувач використовує функціонал ШІ неправильно, що може призвести до неочікуваних результатів.

Таблиця 2.2 – Повний usecase

Розділ	Опис
Use Case Name	Створення проєкту та проведення аналізу, візуалізації даних проєкту
Scope	Система керування проєктами на основі засобів штучного інтелекту
Level	Ціль користувача
Primary Actor	Проєктний менеджер
Stakeholders and interests	– Замовник проєкту: ціль – ефективне керування та візуалізація прогресу. – Робоча команда: ціль – отримання чіткої інструкції щодо виконання завдань
Preconditions	– Коректна реєстрація проєкту в системі; – доступ до вхідних даних проєкту.
Success guarantee	Створення проєкту та успішне проведення аналізу і візуалізації даних.
Main Success Scenario	1. проєктний менеджер входить до системи; 2. обирає опцію «Створення нового проєкту»; 3. вводить необхідну інформацію про проєкт; 4. зберігає дані і переходить до аналізу та візуалізації.
Extensions	– Якщо введена інформація не коректна, система надає повідомлення про помилку. – Якщо аналіз та візуалізація неможлива з причин технічних або даних обмежень
Special Requirements	Висока швидкість обробки та відображення даних.
Technology and Data	Використання штучного інтелекту для аналізу та візуалізації даних.
Variations List	– Можливість вибору різних типів діаграм та графіків для візуалізації. – Автоматичне оновлення даних проєкту для актуальності візуалізації.
Frequency of Occurrence	Висока (80%)
Miscellaneous	– Додаткові функціоні можливості для розширення аналізу та візуалізації. – Можливість експорту результатів аналізу

2.3 Алгоритм роботи програмного забезпечення

Алгоритм роботи програми – це конкретний набір інструкцій або кроків, які визначають порядок виконання певної задачі чи розв’язання конкретної

проблеми в рамках програмного забезпечення. Це є основою для реалізації функцій програми і визначає послідовність дій, які виконуються в процесі взаємодії програми з користувачем або іншими системними компонентами. Важливість алгоритмів роботи програм полягає в декількох ключових аспектах:

1. алгоритми дозволяють визначити логічний порядок виконання завдань, оскільки визначають як програма має реагувати на вхідні дані чи події, як обробляти інформацію та як генерувати вихідні результати;

2. коректно спроєктовані алгоритми можуть значно покращити продуктивність програми, дозволяючи вибирати оптимальні методи виконання завдань, що допомагає уникати зайвих витрат ресурсів;

3. алгоритми сприяють створенню коду, що можна використовувати в різних частинах програми чи навіть в інших програмах, це полегшує розробку та утримання програмного забезпечення;

4. якщо алгоритми роботи програми є зрозумілими та легко читаються, це полегшує розуміння логіки програми для розробників та підтримки обслуговування застосунку.

У контексті моделювання системи важливо враховувати алгоритми роботи для того, щоб правильно визначити внутрішню логіку та поведінку системи. Алгоритми служать як основа для побудови моделі та дозволяють враховувати різноманітні аспекти взаємодії компонентів системи. Правильно спроєктовані алгоритми роботи системи гарантують її ефективність, надійність та можливість розширення.

Таблиця 2.3 – Загальний алгоритм роботи програми

Крок	**Дії**
Збір вхідних даних	Збір інформації про проєкт, включаючи завдання, ресурси, терміни, витрати, ризики та інші важливі параметри.
Попередня обробка даних	Перевірка та очищення вхідних даних від помилок та неправильних значень. Сортування та категоризація даних для подальшого зручного аналізу.

Кінець таблиці 2.3

Введення даних в систему ШІ	Передача оброблених даних до системи керування проєктами на основі штучного інтелекту.
Моделювання проєкту	Використання алгоритмів для створення математичної моделі проєкту на основі вхідних даних. Урахування різних аспектів, таких як ресурси, залежності між завданнями, та інші фактори.
Аналіз та прогнозування	Використання алгоритмів аналізу даних та машинного навчання для виявлення закономірностей та тенденцій у проєкті. Прогнозування можливих ризиків та визначення стратегій їх управління.
Оптимізація ресурсів	Використання алгоритмів оптимізації для ефективного розподілу ресурсів та оптимального використання їх потенціалу.
Візуалізація результатів	Створення графічних представлень, діаграм та інших візуальних засобів для розуміння даних та результатів аналізу.
Прийняття рішень	Використання алгоритмів прийняття рішень для автоматичного або порадного вибору оптимальних шляхів дій.
Моніторинг та коригування	Забезпечення постійного моніторингу ходу проєкту. Автоматичне виявлення відхилень від плану та розробка алгоритмів для коригування стратегій.
Звітність та аналіз результатів	Генерація звітів та аналітичних матеріалів на основі аналізу даних. Представлення результатів у вигляді зрозумілих звітів для зацікавлених сторін.

Створена блок-схема керування проєктом і відображає ключові етапи та процеси проєкту. Блок-схема надає зрозуміле візуальне представлення всіх етапів та різних аспектів управління проєктом [8], що допомагає здійснювати ефективний контроль та приймати управлінські рішення на кожному етапі проєкту.

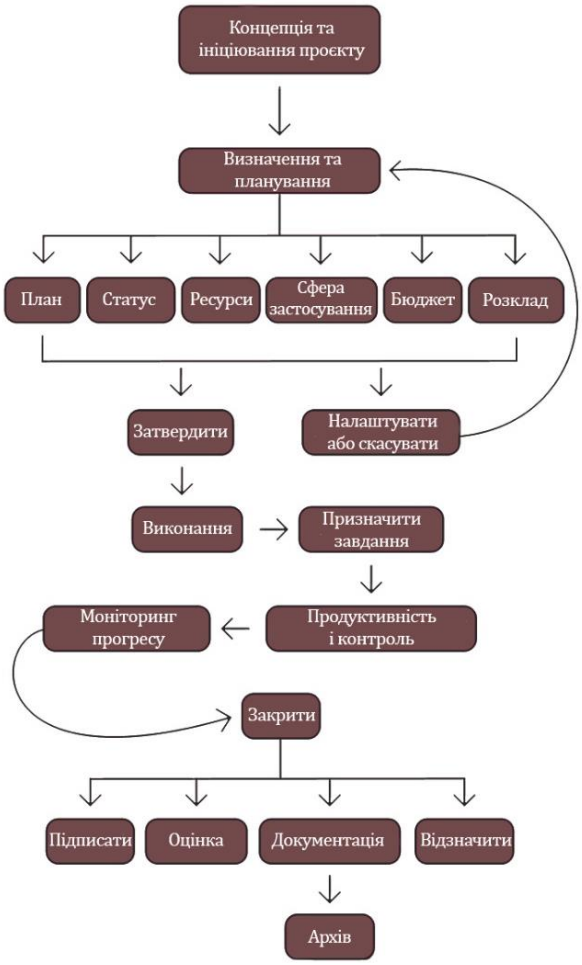

Рисунок 2.8 – Блок-схема керування проєктом

Опис етапів та процесів проєкту:

1. **Ініціювання проєкту:**
   - Починається з ініціювання та переходу до створення Статуту проєкту.
   - Можливі напрямки:
     1) *Якщо Статут проєкту не схвалений*, можливість корекцій чи скасування проєкту, що повертає до етапу ініціювання або призводить до скасування проєкту.
     2) *Якщо Статут проєкту схвалений*, продовження з процесом планування.
2. **План проєкту:**
   - Планування, збір вимог та розробка обсягу призводять до плану проєкту.

- План проєкту визначає ресурси, бюджети, графіки і тощо.
- Кожна підсистема, така як план комунікацій чи план ризиків, веде до іншої точки на діаграмі потоку, впливаючи на план змін та управління якістю.
- Затвердження плану веде до процесу виконання, а відсутність затвердження повертає нас на початок.

3. **Виконання плану:**
   - Розвиток команди проєкту, забезпечення ресурсів для забезпечення якості та розподіл інформації.
   - Можливість внесення коректив, скасування або продовження, залежно від обраного варіанту, повертає нас на початок або переводить до наступного етапу.
4. **Моніторинг та контроль проєкту:**
   - Процес моніторингу та контролю призводить до інтегрованого плану контролю змін.
   - Контроль якості включає звітність про ризики та проблеми та інші етапи.
5. **Завершення проєкту:**
   - Закриття проєкту включає перевірку та прийняття результатів проєкту та операцій.
   - Перехід до вивчених уроків і готовність до підписання відповідного документа.
6. **Весь процес проєкту:**
   - Макро-діаграма, що відображає усі етапи від створення проєкту до завершення, з урахуванням документації, завдань, зустрічей, звітів тощо.
   - Кожен елемент веде до свого власного потоку, розглядаючи різні аспекти проєкту, такі як моніторинг стану проєкту, бюджету, графіку тощо.

2.4 Розробка діаграми розгортання

Для графічного відображення системи управління проєктами, що базується на засобах штучного інтелекту, необхідно створити діаграму розгортання (рис. 2.9). Її мета полягає в представленні обчислювальних вузлів, компонентів та об’єктів, щоб змоделювати та продемонструвати фізичне розгортання артефактів системи на вузлах.

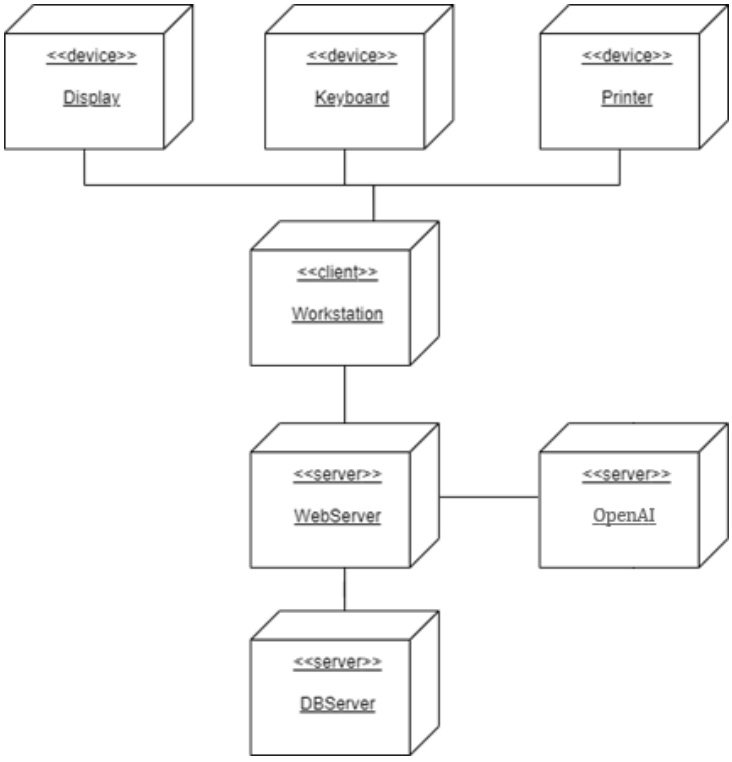

Рисунок 2.9 – Діаграма розгортання система керування проєктами на основі засобів штучного інтелекту

На діаграмі розгортання системи управління проєктами вузли відображено у вигляді прямокутних паралелепіпедів, розділених на два основних типи: обчислювальні фізичні ресурси (вузол пристрою) та програмні ресурси (вузол середовища виконання).

Вузол пристрою містить власну пам’ять та забезпечує виконання застосунку за допомогою сервісів. Таким чином, це може бути різноманітні обчислювальні пристрої, такі як смартфони, ПК, планшети і тощо. На відміну від цього, вузол середовища виконання є сервісом, який може використовувати інші

програмні елементи та функціонує саме у зовнішньому вузлі.

Мета створення діаграми розгортання полягає в раціональній організації компонентів, що є важливою частиною розробки програмного забезпечення. Ефективна організація компонентів впливає на продуктивність та безпеку системи, що розробляється. На даній діаграмі враховано комплекс трьох рівнів архітектури з метою досягнення оптимальної структури системи управління проєктами на основі засобів штучного інтелекту (табл 2.3).

Таблиця 2.4 – Опис архітектури програмного комплексу

№	Компонент	Опис
1	Клієнт – workstation.	*Робоче місце* з пристроями введення та виведення, такими як дисплей, клавіатура, принтер.
2	Сервер – webserver.	*Вебсервер,* що відповідає за *виконання основної бізнес-логіки* програмного комплексу.
3	Сервер – OpenAI	*Компонент,* який *використовує функціонал OpenAI* для розв'язання конкретних завдань.
4	Сервер – DBServer.	*Сервер*, до якого відбувається доступ з вебсервера *для взаємодії з базою даних*.

Створення діаграми розгортання для системи керування проєктами на основі засобів штучного інтелекту визначає структурні взаємозв'язки між фізичними та програмними компонентами. Ця діаграма не лише візуалізує фізичне розташування артефактів системи на різних вузлах, але й допомагає раціонально організувати їх взаємодію. З врахуванням трьошарової архітектури, система ефективно використовує обчислювальні та програмні ресурси, що сприяє підвищенню продуктивності та забезпечує безпеку управління проєктами з використанням штучного інтелекту.

Висновки до розділу 2

У другому розділі з моделювання системи керування проєктами на основі засобів штучного інтелекту, вивчені та враховані ключові аспекти розробки програмного забезпечення для системи керування проєктами на основі засобів штучного інтелекту. Моделі системи, такі як ERD, DFD на рівнях 0-2 та IDEF0,

надали глибокий уявлення про структуру та взаємодію елементів системи. Створення повного опису та діаграм прецедентів через Use case дозволило чітко визначити функціональні вимоги до системи та ідентифікувати її основні взаємодії з користувачами. Алгоритм роботи програмного забезпечення був повністю описаний, а блок-схема керування проєктом дозволила візуалізувати послідовність операцій та прийняти рішень у процесі управління проєктом.

Завершення розділу включило розробку діаграми розгортання, яка надає візуальне відображення фізичного розташування компонентів системи на різних рівнях. Відомості з другого розділу сприятимуть ефективній інтеграції та розробці програмного комплексу для системи керування проєктами на основі засобів штучного інтелекту.

3 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ТА ОГЛЯД ТЕХНОЛОГІЧНОГО СТЕКУ

3.1 Ієрархічна модель прогнозування станів системи та МАІ

Ієрархічна модель прогнозування майбутніх станів системи керування проєктами на основі засобів штучного інтелекту (рис. 3.1) є важливим інструментом для ефективного управління проєктами [9]. Основна ідея полягає в тому, щоб розділити систему керування на більш прості підсистеми і прогнозувати їхні майбутні стани, а потім інтегрувати ці прогнози для отримання загального прогнозу.

Рисунок 3.1 – Блок-схема сценарної моделі

Таблиця 3.1 – Умовні позначення блок-схеми (рис. 1)

Позначення	Опис
$a_{1–7}$	*Актори*
$O_{1–7}$	*Мета акторів*
$C_{1–7}$	*Керуючий вплив*
$S_{1–3}$	*Сценарій*

Таблиця 3.2 – Опис сценаріїв здійснення дослідження

Сценарій	Позначення	Опис	Наслідки
Песимістичний	S_1	1. Надмірні затрати на проєкти та відсутність необхідних ресурсів. 2. Затримки у виконанні проєктів через несприятливі умови та труднощі в роботі. 3. Втрата клієнтів та конкурентність компанії.	1. Збільшення витрат на проєкти, що може вплинути на фінансовий стан компанії. 2. Погіршення репутації компанії через невиконання проєктних обіцянок. 3. Зменшення мотивації команд та збільшення перерв у роботі.
Консервативний	S_2	1. Збереження поточних ресурсів та вимог до проєктів без значних змін. 2. Збереження стабільних процесів та планів. 3. Зниження ризику виконання проєктів та невеликі зміни в стратегії компанії.	1. Стабільність виконання проєктів, зниження ризику інновацій та ризиків. 2. Збереження відносно низьких витрат і фінансової стабільності. 3. Підтримка довгострокової стабільності компанії.
Оптимістичний	S_3	1. Розширення проєктної діяльності та впровадження нових можливостей. 2. Покращення продуктивності та ефективності управління. 3. Залучення інновацій та нових технологій.	1. Збільшення прибутковості проєктів та розширення бізнесу. 2. Залучення нових клієнтів та збільшення конкурентоспроможності. 3. Підвищення мотивації команд та привертання талантів.

Використання ієрархічної моделі прогнозування має декілька важливих переваг:

1. розділення системи на підсистеми дозволяє краще розуміти компоненти проєкту і їх взаємозв’язки. Це полегшує аналіз і прогнозування майбутніх станів кожної окремої частини;

2. прогнозування майбутніх станів в кожній підсистемі дозволяє здійснювати більш точні прогнози, оскільки враховується більше деталей і факторів, які впливають на кожну конкретну частину системи;

3. знання майбутніх станів допомагає керівникам проєктів приймати кращі рішення з управління, наприклад, вчасно реагувати на можливі проблеми або змінювати стратегії;

4. ієрархічна модель дозволяє легко створювати різні сценарії розвитку подій та аналізувати їхні наслідки, що допомагає знайти оптимальний шлях для реалізації проєкту;

5. використання засобів штучного інтелекту дозволяє автоматизувати процес прогнозування та аналізу даних, що робить його більш ефективним і точним.

Метод аналізу ієрархії є одним з методів багатокритеріального аналізу, який дозволяє систематично оцінювати альтернативи за допомогою ієрархічної структури критеріїв. У контексті оцінки технологій для системи керування проєктами на основі засобів штучного інтелекту, MAI може бути вкрай корисним і має наступні переваги:

1. метод аналізу ієрархії дозволяє систематизувати критерії та альтернативи, розглядаючи їх у відповідних контекстах. Це допомагає уникнути суб’єктивних оцінок та допомагає виявити ключові аспекти для порівняння технологій.

2. при розгляді систем керування проєктами на основі штучного інтелекту може бути багато різних аспектів для врахування. MAI дозволяє розбити цю складність на менші, керовані елементи, що полегшує процес прийняття рішень.

3. метод аналізу ієрархії допомагає узгоджувати оцінки між різними учасниками процесу прийняття рішень, що може бути важливим у випадках, коли думки різняться.

4. в ході проєктування системи нова інформація може впливати на оцінку технологій, тому метод аналізу ієрархії дозволяє легко вносити корективи в оцінки з урахуванням нових даних чи уточнень.

Оцінено 3 технології для системи керування проєктами на основі засобів штучного інтелекту, яка розробляється, за обраними критеріями (атрибутами) якості на основі методу аналізу ієрархії (MAI).

Обрані технології (A):

A_1 – Машинне навчання (Machine Learning): дозволяє системі аналізувати дані, розпізнавати патерни та приймати рішення на основі інформації, через що, у проєктному управлінні, система може використовувати машинне навчання для передбачення термінів завершення проєктів, виділення ризиків та рекомендацій щодо оптимізації процесів;

A_2 – Аналіз даних та візуалізація (Data Analysis and Visualization): Інструменти для аналізу даних та візуалізації можуть бути використані для створення інтерактивних панелей та звітів, які допомагають управляти проєктами, тому ШІ може використовуватися для автоматизації аналізу даних та видачі рекомендацій на основі результатів аналізу;

A_3 – Роботизований процес автоматизації (RPA): використовується для автоматизації рутинних завдань та процесів у проєктному керуванні, таких як введення даних, виконання повторюваних дій та інших операцій. Це допоможе зменшити витрати часу та ресурсів, звільняючи співробітників для більш важливих завдань.

Перелік критеріїв (K):

K_1 – Відповідність функціоналу потребам проєкту;

K_2 – Фінансова доцільність;

K_3 – Ресурсоємність технічного обладнання;

K_4 – Перспективи розвитку.

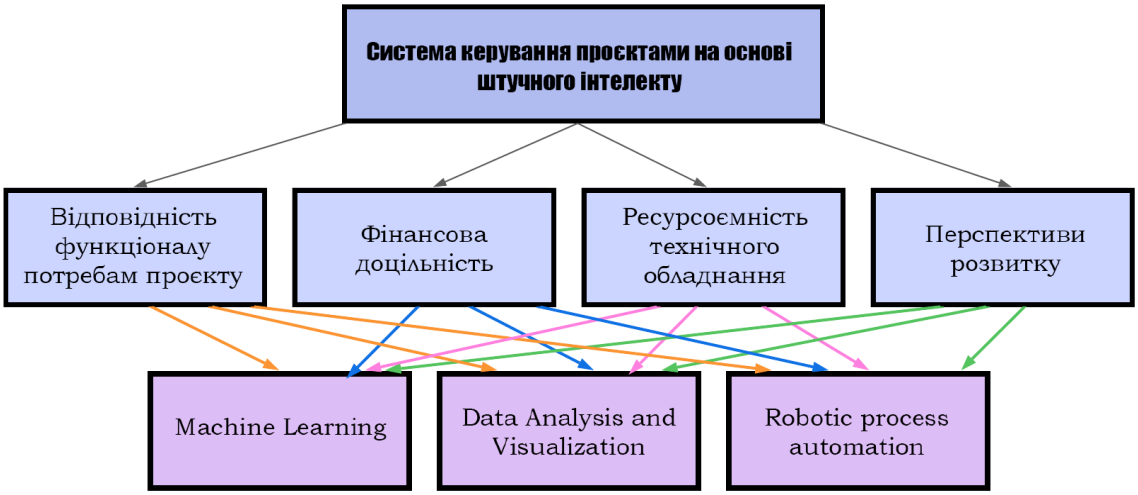

Рисунок 3.2 – Ієрархічна модель для критерію вибору технологій

Критерії оцінювались за 9-ти бальною шкалою Сааті (табл. 1), де значення 2, 4, 6, 8 є проміжними градаціями [10].

Таблиця 3.3 – Шкала оцінювання Сааті

Ступінь переваги	**Визначення**	**Коментар**
1	Рівна важливість	Дві альтернативи еквівалентні за перевагою
3	Помірне перевага одного над іншим	Одна альтернатива помірно краща за іншу
5	Істотне або сильне перевагу	Одна альтернатива істотно краща за іншу
7	Значна перевага	Одна альтернатива значно краща за іншу
9	Дуже сильне перевага	Беззаперечна переважність однієї альтернативи над іншою

Крім того при розрахунках необхідно враховувати значення випадкового індексу (RI) узгодженості для матриць різної розмірності (табл. 2). Обчислений на основі методу попарного порівнянь власний вектор прийнятний в тому випадку, якщо $CR \leq 0.10$.

Таблиця 3.4 – Значення випадкового індексу (RI) узгодженості для матриць різної розмірності

Порядок МПП	1	2	3	4	5	6	7	8	9	10
Випадкового індекс (*RI*)	0	0	0,58	0,89	1,1	1,2	1,3	1,4	1,4	1,4

Алгоритм вибору кращої альтернативи за методом MAI має наступні етапи:

1. Побудова ієрархічної структури проблеми, яка містить декілька рівнів: цілі – критерії – альтернативи. В рамках цього етапу, формується множина об'єктів експертизи (альтернатив) $A = \{A_i \mid i = \overline{1,z}\}$, та множина критеріїв $K = \{K_l \mid l = \overline{1,n}\}$ (у разі вирішення багатокритеріальної задачі вибору).

2. Проводиться процедура визначення пріоритетів між об'єктами (критеріями, альтернативами) на основі експертних вподобань. Для цього створюється матриця парних порівнянь (МПП) для кожного критерію та альтернативи за всіма заданими критеріями.

3. Розрахунок вектору пріоритетів МПП $\Omega = \{\omega_q \mid q = \overline{1,s}\}$, елементи якого відповідають умовам:

$$0 \le \omega_q \le 1, \quad \forall q = \overline{1,s}; \qquad \sum_{q=1}^{s} \omega_q = 1. \tag{1}$$

Вектор пріоритетів розраховується на основі кожної сформованої МПП, тобто для критеріїв, та для всіх альтернатив за кожним критерієм окремо.

4. Перевірка узгодженості сукупності ЕО.

5. Розрахунок кількісного показника якості кожної альтернативи на основі виразу:

$$U_i = \sum_{j=1}^{n} w_j r_{ij}\,, \tag{2}$$

де U_i – кількісний показник якості i-ї альтернативи;

w_j – вага j-го критерію;

r_{ij} – вага i-ї альтернативи для j-го критерію.

Найбільше значення U_i має краща альтернатива.

Розрахунки

Розрахунки проведено за допомогою мови програмування Python, що використовувалась для створення програмна реалізація обчислення MAI

(Додаток А). Процес розрахунку ваги критеріїв в методі аналізу ієрархій включає кілька кроків. Спочатку проводиться попарне порівняння критеріїв (рис. 3.3) щодо їх відносної важливості. На основі цих порівнянь обчислюються ваги критеріїв, які використовуються для подальшого аналізу альтернатив.

```
Ваги критеріїв (попарний розрахунок):
Відповідність функціоналу потребам проєкту    0.58
Фінансова доцільність                         0.13
Ресурсоємність технічного обладнання          0.09
Перспективи розвитку                          0.19
```

Рисунок 3.3 – Розрахунок ваги критеріїв

У наведеному коді функція **pairwise_weight_calculation** відповідає за процес попарного розрахунку ваг критеріїв. Вона обчислює вагу кожного критерію на основі матриці попарних порівнянь, що підготовлена у функції **prepare_items_for_weight_calculation**.

```
def prepare_items_for_weight_calculation(items_to_evaluate):
    unique_pairs = list()
    for comb in combinations(items_to_evaluate, 2):
        unique_pairs.append(comb)

    points = dict()
    for pair in unique_pairs:
        points[pair] = fractional_input(input(f»\033[1mОцініть {pair[0]}
відносно {pair[1]}:\033[0m «))

    data_frame = pd.DataFrame(columns=items_to_evaluate,
index=items_to_evaluate)
    for key in points:
        data_frame.at[key[0], key[1]] = points[key]
        data_frame.at[key[0], key[0]] = 1
        data_frame.at[key[1], key[1]] = 1
        data_frame.at[key[1], key[0]] = 1 / points[key]

    return data_frame

def pairwise_weight_calculation(points_data_frame):
    d = points_data_frame.product(axis=«columns»).pow(1 / 3)
    D = d.sum()
    a = d.div(D).apply(lambda x: round(x, 2))
    return a
```

Наступним кроком проводиться оцінка технологій за чотирма критеріями: «Відповідність функціоналу протребам проєкту» (рис. 3.4), «Фінансова доцільність» (рис. 3.5), «Ресурсоємність технічного обладнання» (рис. 3.6), та

«Перспективи розвитку» (рис. 3.7). Це важливий крок у виборі оптимальної технології для системи керування проєктами на основі штучного інтелекту, оскільки дозволяє врахувати різноманітні аспекти, відповідно до вимог та цілей проєкту. Кожен з цих критеріїв визначається та оцінюється з урахуванням його вагомості та впливу на результативність системи. На основі результатів оцінки можна зробити обґрунтований вибір технології, яка найкращим чином відповідає потребам та умовам конкретного проєкту.

```
Оцінка предметів за критерієм Відповідність функціоналу потребам проєкту:
Оцініть Машинне навчання (Machine Learning) відносно Аналіз даних та візуалізація (Data Analysis and Visualization): 1/5
Оцініть Машинне навчання (Machine Learning) відносно Роботизований процес автоматизації (RPA): 7
Оцініть Аналіз даних та візуалізація (Data Analysis and Visualization) відносно Роботизований процес автоматизації (RPA): 9
```

Рисунок 3.4 – Оцінка технологій за критерієм
«Відповідність функціоналу протребам проєкту»

```
Оцінка предметів за критерієм Фінансова доцільність:
Оцініть Машинне навчання (Machine Learning) відносно Аналіз даних та візуалізація (Data Analysis and Visualization): 1/5
Оцініть Машинне навчання (Machine Learning) відносно Роботизований процес автоматизації (RPA): 3
Оцініть Аналіз даних та візуалізація (Data Analysis and Visualization) відносно Роботизований процес автоматизації (RPA): 7
```

Рисунок 3.5 – Оцінка технологій за критерієм
«Фінансова доцільність»

```
Оцінка предметів за критерієм Ресурсоємність технічного обладнання:
Оцініть Машинне навчання (Machine Learning) відносно Аналіз даних та візуалізація (Data Analysis and Visualization): 1/5
Оцініть Машинне навчання (Machine Learning) відносно Роботизований процес автоматизації (RPA): 1/3
Оцініть Аналіз даних та візуалізація (Data Analysis and Visualization) відносно Роботизований процес автоматизації (RPA): 5
```

Рисунок 3.6 – Оцінка технологій за критерієм
«Ресурсоємність технічного обладнання»

```
Оцінка предметів за критерієм Перспективи розвитку:
Оцініть Машинне навчання (Machine Learning) відносно Аналіз даних та візуалізація (Data Analysis and Visualization): 3
Оцініть Машинне навчання (Machine Learning) відносно Роботизований процес автоматизації (RPA): 7
Оцініть Аналіз даних та візуалізація (Data Analysis and Visualization) відносно Роботизований процес автоматизації (RPA): 9
```

Рисунок 3.7 – Оцінка технологій за критерієм
«Перспективи розвитку»

У коді частина, що відповідає за оцінку технологій за кожним з критеріїв, міститься в основному блоку програми, який розпочинається з рядка **if __name__ == '__main__':** і закінчується перед виведенням результатів. В цьому блоку виконуються наступні дії:

1. Виведення тексту-підказки для введення критеріїв.
2. Зчитування та підготовка даних для оцінки технологій за кожним з критеріїв.
3. Виклик функції **prepare_items_for_weight_calculation**, яка готує дані для оцінки технологій за кожним критерієм.
4. Цикл, що пробігає всі критерії і викликає функцію **pairwise_weight_calculation** для розрахунку ваг кожного критерію.
5. Виведення результатів оцінки важливості кожного критерію (**criteria_w**).
6. Оцінка предметів за кожним критерієм та розрахунок значень CR (**items_cr**).
7. Виведення результатів оцінки технологій за кожним критерієм та підсумкового результату.

Обчислення CR (рис. 3.8) відбувається за допомогою функції **calculate_cr_value**:

```
def calculate_cr_value(points_data_frame, w, m, ri):
    lambda_max = points_data_frame.sum().mul(w).sum()
    ci = (lambda_max - m) / (m - 1)
    return round(ci / ri, 3)
```

```
Значення CR:
{'Відповідність функціоналу потребам проєкту': 0.178, 'Фінансова доцільність': 0.055, 'Ресурсоємність технічного обладнання': 0.126, 'Перспективи розвитку': 0.216}
```

Рисунок 3.8 – Значення CR

За результати обчислень відповідає функція **calculate_results**, яка в **main** викликається за допомогою «**result = calculate_results(items_w, criteria_w)**»:

```
def calculate_results(items_weights_list, criteria_weight):
    df = pd.DataFrame(items_weights_list)
    return df.mul(criteria_weight.to_numpy(), axis=«index»).sum().apply(lambda
x: round(x, 2))
```

```
Результат:
Машинне навчання (Machine Learning)                              0.28
Аналіз даних та візуалізація (Data Analysis and Visualization)   0.64
Роботизований процес автоматизації (RPA)                         0.07
```

Рисунок 3.9 – Результат обчислень

Відповідно до отриманих результатів (рис. 10), можна зробити висновки, що технологія, яка підходить *найкраще для системи керування проєктами є Аналіз та візуалізація*, згідно з оцінками експертів та вагою критеріїв, за якими здійснено обрахунки.

3.2 Розробка UML-діаграм

UML (Unified Modeling Language) – це стандартна мова для візуального моделювання програмних систем. UML-діаграми дозволяють графічно представити різні аспекти системи, такі як її структуру, поведінку, взаємодію між компонентами та інше.

Важливість розробки діаграм для проєктування системи керування проєктами на основі засобів штучного інтелекту полягає в наступному [11]:

1. Візуалізація концепцій. Дозволяють команді проєкту ясно розуміти концепції та взаємодію компонентів системи, що полегшує спілкування та співпрацю.

2. Аналіз та проєктування. Допомагають аналізувати вимоги до системи та проєктувати її архітектуру, допомагаючи виявити потрібні класи, взаємозв’язки та інші аспекти.

3. Документація. Слугують як ефективний інструмент для створення документації проєкту, яка допомагає зберігати та передавати знання про систему.

4. Виявлення помилок та уточнення вимог. Шляхом моделювання системи за допомогою UML-діаграм можна виявити розбіжність в вимогах або потенційні проблеми в архітектурі ще до розробки коду.

5. Підтримка розробки коду. Можуть слугувати основою для генерації коду або допомагати розробникам краще зрозуміти структуру системи під час програмування.

Отже, розробка UML-діаграм є важливою для ефективного проєктування та розробки системи керування проєктами на основі засобів штучного інтелекту,

сприяючи як взаєморозумінню у команді, так і досягненню якості та функціональності в кінцевому продукті. Activity Diagram, Class Diagram та Sequence Diagram є ключовими типами UML-діаграм (табл. 3.5), які мають важливе значення для проєктування систем, включаючи «Систему керування проєктами на основі засобів штучного інтелекту».

Таблиця 3.5 – Характеристики діаграм

Тип діаграми	**Важливість**	**Застосування**
Activity Diagram	Моделювання послідовності дій або процесів, допомагає у виявленні потенційних ризиків та удосконаленні бізнес-процесів.	Використовується для моделювання різних бізнес-процесів у системі керування проєктами.
Class Diagram	Візуалізація структури системи, виявлення класів, їх атрибутів та методів.	Допомагає зрозуміти структуру та організацію об’єктів у системі керування проєктами.
Sequence Diagram	Моделювання взаємодії між об’єктами в послідовному порядку.	Використовується для представлення послідовності взаємодії між різними модулями та компонентами у системі керування проєктами.

Activity Diagram (діаграма діяльності) – моделює послідовність дій або процесів у системі. UML-діаграма діяльності системи керування проєктами, яка показує потоки між діяльністю проєкту, завдання, зарплати, працівника, помилки. Основна діяльність, залучена в цій діаграмі діяльності UML системи управління проєктами, полягає в наступному:

- проєктна діяльність;
- завдання;
- зарплатна діяльність;
- активність співробітників;
- робота з помилками.

Особливості діаграми Activity UML системи управління проєктами:

– користувач-адміністратор може шукати проєкт, переглядати опис вибраного проєкту, додавати проєкт, оновлювати проєкт і видаляти проєкт;

– показує процес редагування, додавання та оновлення завдання;

– користувач зможе шукати та створювати звіти про зарплату, працівника, помилку;

– усі об’єкти, такі як (Проєкт, Завдання, Помилка) взаємопов’язані;

– відображає повний опис і порядок проєктів, співробітників, помилок, зарплати, завдань.

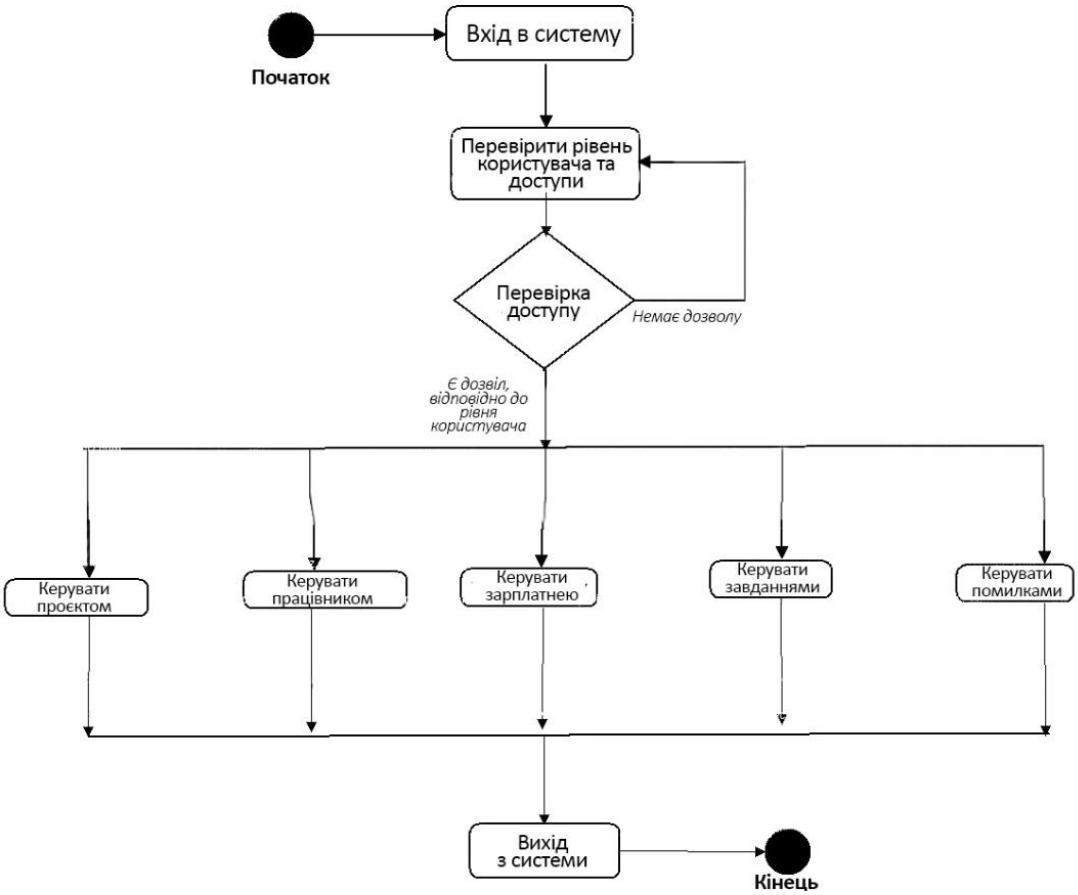

Рисунок 3.10 – Діаграма діяльності перевірки доступів

Це діаграма активності входу системи керування проєктами, яка показує потоки активності входу, де адміністратор зможе входити, використовуючи своє ім’я користувача та пароль. Після входу користувач може керувати всіма операціями щодо зарплати, проєкту, завдання, помилки, працівника. Усі

сторінки, такі як Завдання, Помилка, Співробітник, безпечні, і користувач може отримати доступ до них після входу. Наведена нижче схема допомагає продемонструвати, як працює сторінка входу в систему керування проєктами. Різні об’єкти на сторінці «Помилка», «Зарплата», «Проєкт», «Завдання» та «Співробітник» взаємодіють під час дії, і користувач не зможе отримати доступ до цієї сторінки, не підтвердивши свою особу.

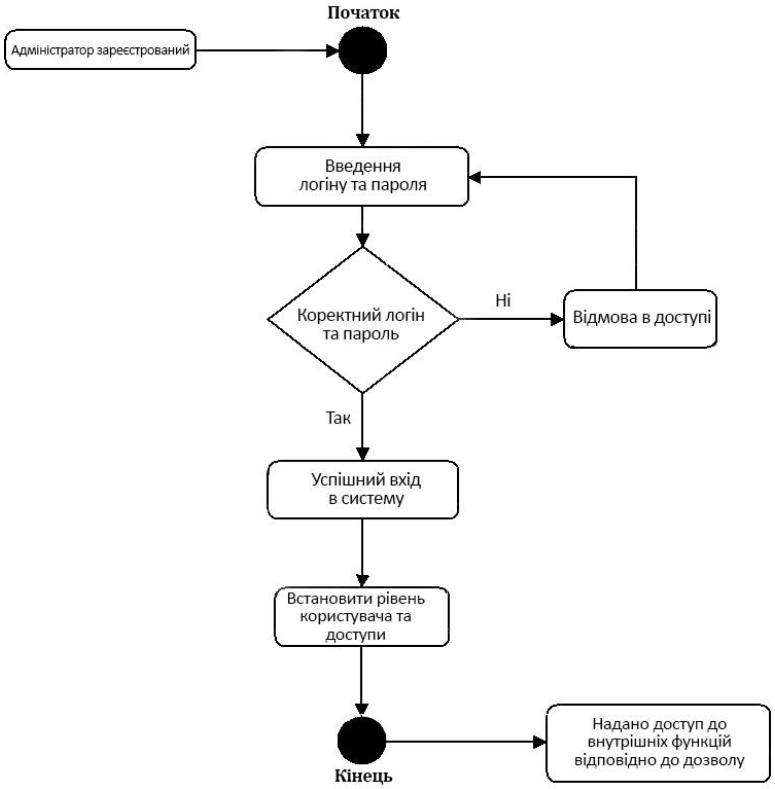

Рисунок 3.11 – Діаграма діяльності надання доступів

Class Diagram (діаграма класів) – візуалізує структуру системи та класи, їх атрибути та методи [12]. Діаграма класів системи керування проєктами описує структуру класів системи керування проєктами, їхні атрибути, операції (або методи) та зв’язки між об’єктами. Основними класами системи управління проєктами є Project, Employee, Employee Salary, Task, Bug, Ticket (табл. 3.6).

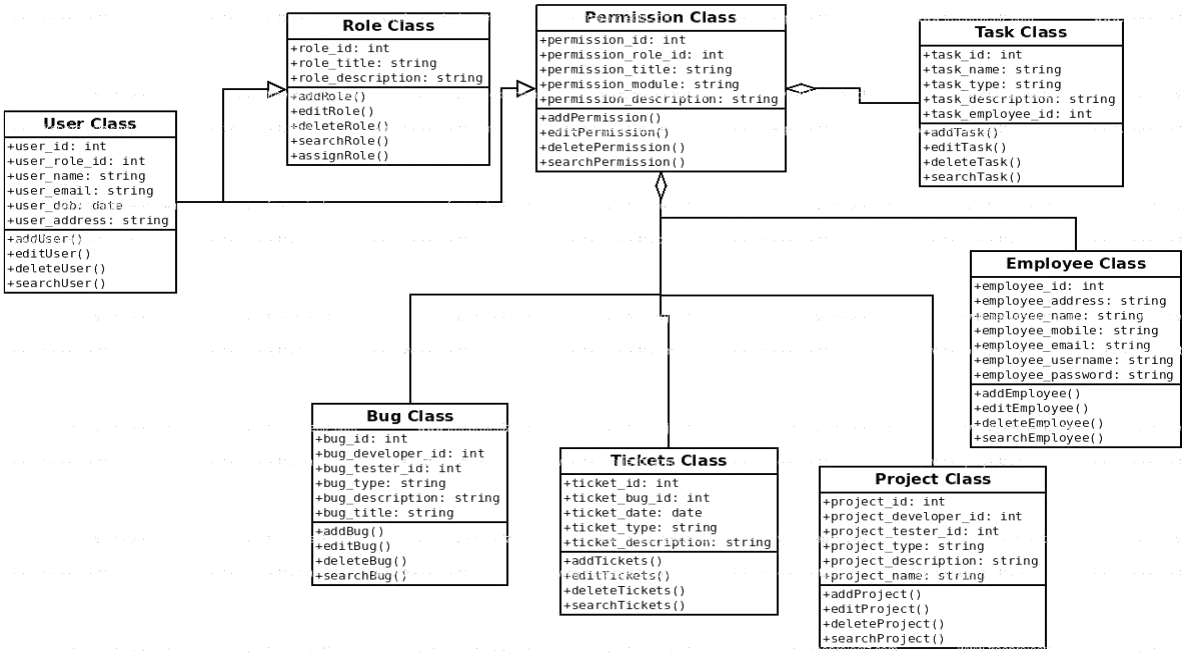

Рисунок 3.12 – Діаграма класів системи управління проєктами

Діаграма класів класів системи управління проєктами складається з:

– Project Class: керування всіма операціями Project (проєкт);

– Employee Class: керування всіма операціями Employee (працівник);

– Employee Salary Class: керування всіма операціями Employee Salary (зарплатня працівника);

– Task Class: керування всіма операціями Task (завдання);

– Bug Class: керування всіма операціями Bug (помилка);

– Ticket Class: керування всіма операціями Ticket (задача).

Таблиця 3.6 – Класи та їх атрибути й методи

Клас	Атрибути	Методи
Project	project_id, project_developer_id, project_tester_id, project_name, project_assign, project_last_date, project_type, project_description	addProject(), editProject(), deleteProject(), updateProject(), saveProject(), searchProject()
Employee	Employee_id, Employee_name, Employee_mobile, Employee_email, Employee_username, Employee_password,	addEmployee(), editEmployee(), deleteEmployee(), updateEmployee(), saveEmployee(), searchEmployee()

Кінець таблиці 3.6

Employee Salary	salary_id, salary_employee_id, salary_amount, salary_total, salary_type, salary_description	addEmployeeSalary(), editEmployeeSalary(), deleteEmployeeSalary(), updateEmployeeSalary(), saveEmployeeSalary(), searchEmployeeSalary()
Task	task_id, task_employee_id, task_name, task_type, task_description	addTask(), editTask(), deleteTask(), updateTask(), saveTask(), searchTask()
Bug	bug_id, bug_developer_id, bug_tester_id, bug_title, bug_type, bug_description	addBug(), editBug(), deleteBug(), updateBug(), saveBug(), searchBug()
Ticket	ticket_id, ticket_bug_id, ticket_type, ticket_date, ticket_description	addTicket(), editTicket(), deleteTicket(), updateTicket(), saveTicket(), searchTicket()

Squence Diagram (діаграма послідовностей) – моделює взаємодію об’єктів у системі в послідовному порядку [13]. Це діаграма послідовності UML системи управління проєктами, яка показує взаємодію між об’єктами Employee, Timesheet, Bug, Salary, Ticket.

Екземпляри об’єктів класу, які беруть участь у цій діаграмі послідовності UML системи керування проєктами:

- об’єкт працівника (Employee);
- об’єкт табеля робочого часу (Timesheet);
- об’єкт помилок (Bug);
- об’єкт зарплати (Salary);
- об’єкт задачі (Ticket);

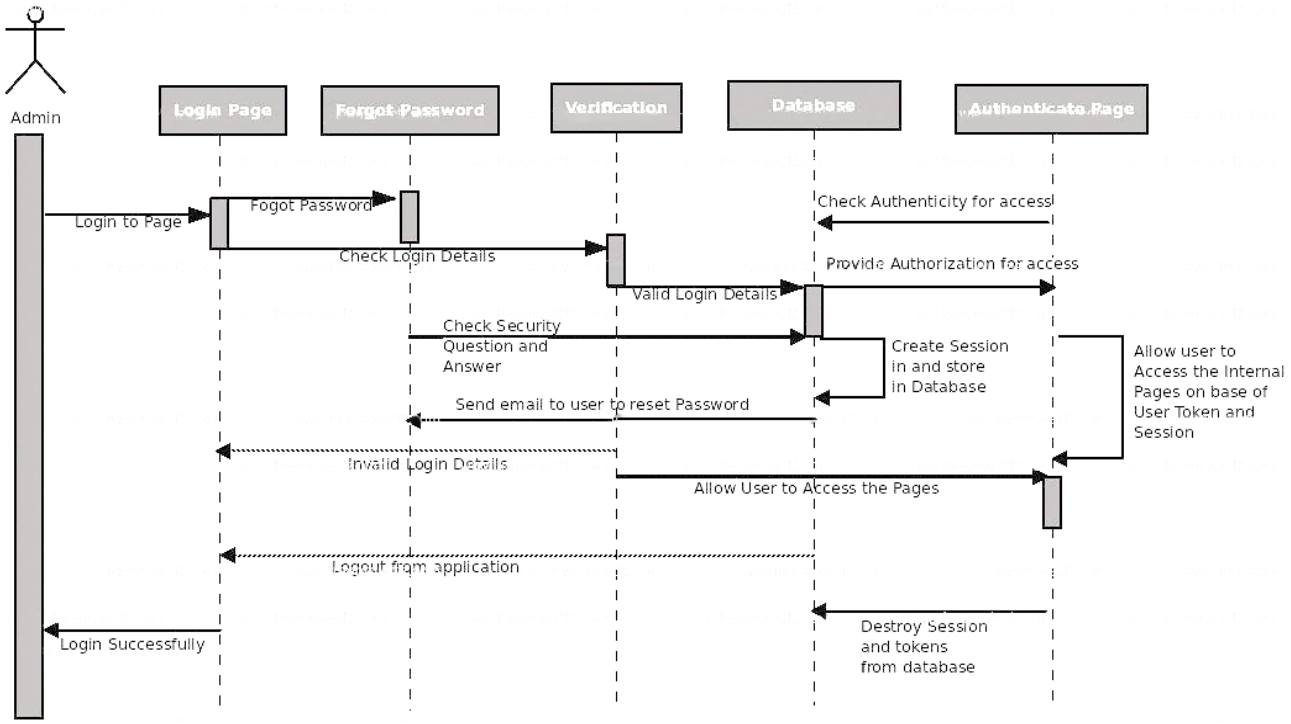

Рисунок 3.13 – Діаграма послідовності входу
в систему управління проєктами

Діаграма послідовності входу в систему керування проєктами (рис. 3.13), де адміністратор зможе ввійти у свій обліковий запис, використовуючи свої облікові дані. Після входу користувач може керувати всіма операціями щодо помилки, працівника, табелю, квитка, зарплати. Усі сторінки, такі як «Табель обліку робочого часу», «Задача», «Зарплата», захищені, і користувач може отримати доступ до них після входу.

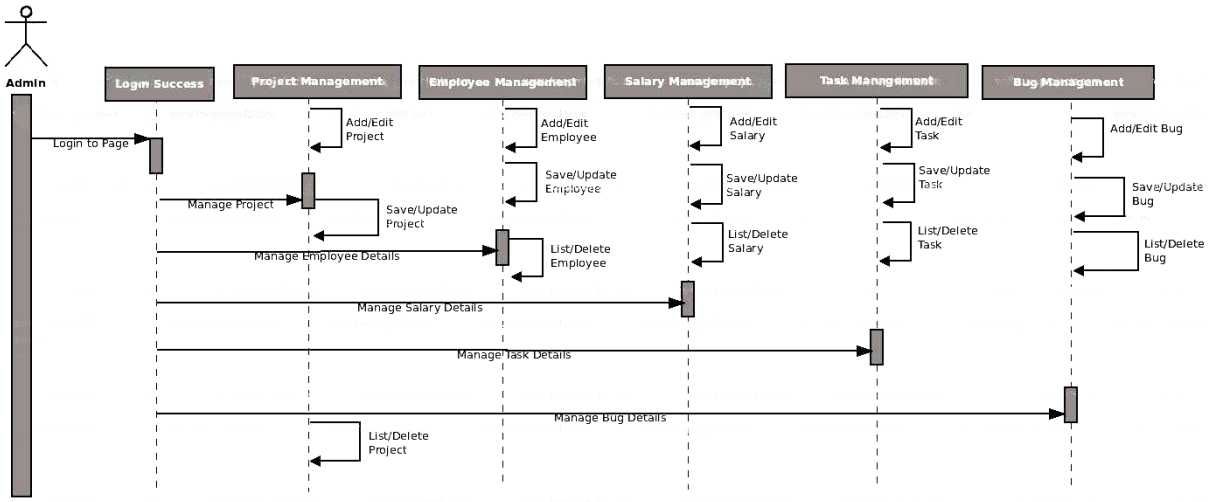

Рисунок 3.14 – Діаграма послідовності роботи з системою

Це діаграма послідовності системи управління проєктами, яка показує взаємодію між об’єктами Employee, Timesheet, Bug, Salary, Ticket. Наведена схема (рис. 3.14) допомагає продемонструвати, як працює сторінка входу в систему керування проєктами. Різні об’єкти на сторінці «Задача», «Помилка», «Працівник», «Табель» і «Зарплата» взаємодіють протягом послідовності, і користувач не зможе отримати доступ до цієї сторінки, не підтвердивши свою особу.

3.3 Огляд технологій

При розробці сучасних вебзастосунків ключовою ролью відводиться правильному вибору технологій, які забезпечать ефективну та надійну роботу продукту. Особливо важливими складовими є front-end, back-end та система управління базою даних, кожна з яких виконує свої унікальні функції, сприяючи зручності використання та функціональності.

Рисунок 3.15 – Стек технологій частини back-end

Back-end вебзастосунку реалізований за допомогою Node.js та Express. Node.js – це середовище виконання JavaScript на сервері, яке дозволяє створювати швидкі та масштабовані додатки. Express – це легковаговий фреймворк для створення вебзастосунків на Node.js, який дозволяє швидко створювати API. Для управління CORS (Cross-Origin Resource Sharing)

використано спеціальний пакет, що дозволяє налаштовувати доступ до ресурсів на сервері з різних джерел.

Додатково, використано OpenAI API для інтеграції із штучним інтелектом, що дозволяє використовувати потужні моделі машинного навчання для розв'язання різноманітних завдань, таких як генерація тексту, аналіз мови та багато інших. OpenAI API – це інтерфейс програмування застосунків, який надає доступ до потужних моделей штучного інтелекту, розроблених OpenAI [14]. Ці моделі можуть бути використані для різноманітних завдань, включаючи аналіз тексту, генерацію контенту, мовний переклад, створення візуального контенту та багато іншого.

Рисунок 3.16 – Логотип OpenAI API

У контексті системи керування проєктами, OpenAI API може бути використаний для аналізу даних та візуалізації шляхом використання його моделей для вирішення різних завдань:

– Аналіз тексту: OpenAI API може бути використаний для аналізу текстових даних, таких як описи завдань, коментарі користувачів або документація проєкту. За допомогою моделей машинного навчання, доступних через OpenAI API, можна виявляти ключові слова, теми, сутності, відносини між даними тощо.

– Генерація звітів та рекомендацій: на основі аналізу тексту, моделі OpenAI можуть генерувати звіти, які містять важливу інформацію про стан проєкту, рекомендації щодо подальших кроків або описи можливих ризиків.

– Прогнозування та візуалізація даних: OpenAI API може бути використаний для прогнозування розвитку проєкту на основі аналізу історичних даних. Результати прогнозу можна візуалізувати у вигляді графіків, діаграм або інших візуальних елементів, що дозволяє керівництву та учасникам проєкту краще розуміти ситуацію та приймати обґрунтовані рішення.

– Розпізнавання та класифікація даних: OpenAI API може допомогти в розпізнаванні та класифікації різних типів даних, що може бути корисним для автоматизації обробки і аналізу інформації у системі керування проєктом.

Загалом, інтеграція OpenAI API у систему керування проєктами може допомогти покращити аналітику, прийняття рішень та ефективність роботи команди, надаючи доступ до потужних інструментів штучного інтелекту.

Рисунок 3.17 – Стек технологій частини front-end

Front-end вебзастосунку реалізований за допомогою React, який є одним з найпопулярніших фреймворків для створення користувацьких інтерфейсів. Ми також використовуємо бібліотеки React Query, React Redux та React Redux Toolkit для кращого управління станом додатку та оптимізації запитів до сервера. Для маршрутизації ми використовуємо React Router, що дозволяє нам легко переходити між різними сторінками нашого вебзастосунку. Для стилізації використовується Tailwind CSS, який дозволяє швидко та зручно створювати стильовий дизайн.

Рисунок 3.18 – СКБД PostgreSQL

Оскільки існує декілька популярних реляційних баз даних – PostgreSQL та MySQL, прийнято рішення порівняти їх за рядом ключових характеристик (табл. 3.7). Відповідно до результатів порівняння, для зберігання даних використовується PostgreSQL – потужна та надійна система керування реляційними базами даних. PostgreSQL надає нам можливість ефективно зберігати, оновлювати та отримувати дані з нашого вебзастосунку.

Таблиця 3.7 – Порівняння СКБД PostgreSQL та MySQL

Характеристика	**PostgreSQL**	**MySQL**
Тип бази даних	Реляційна	Реляційна
Відкритість	Вільний та відкритий код	Вільний та відкритий код
Підтримка ACID	Повна	Повна
Спрощення реплікації	Є	Є
Підтримка географічних функцій	Так	Обмежена
Масштабованість	Добра масштабованість на великі обсяги даних	Добра масштабованість, але PostgreSQL краще підходить для великих обсягів даних
Підтримка JSON	Повна	Є, але обмежена
Підтримка розширень	Широкий вибір розширень	Обмежена
Продуктивність	Висока	Висока
Управління транзакціями	Потужне	Потужне
Оптимізація запитів	Є, оптимізація та підтримка засобів для аналізу запитів	Є, але менш обширна
Комплексність	Є різноманітність функцій та можливостей	Є, але PostgreSQL має більш широкі можливості
Підтримка відкритих стандартів	Широко підтримується	Широко підтримується

Кінець таблиці 3.7

Відмовостійкість	Висока	Висока
Підтримка партицій	Повна	обмежена
Підтримка крос-платформеності	Є	Є

Вона відображає переваги PostgreSQL над MySQL у багатьох аспектах, таких як повнота підтримки JSON, географічних функцій, оптимізація запитів та широкий набір можливостей для великих обсягів даних. Висновок полягає в тому, що PostgreSQL може бути більш коректним вибором для складних та вимогливих проєктів, де потрібна висока продуктивність, розширені функціональність та надійність системи керування базами даних.

Комунікація між компонентами в контексті розробки програмного забезпечення означає взаємодію та обмін даними між різними частинами системи (рис. 3.19). Це може включати взаємодію між клієнтським та серверним компонентами, різними мікросервісами або модулями програми.

Рисунок 3.19 – Принцип роботи моделі RESTful API

Для цього процесу обрано RESTful API (Representational State Transfer), який є стандартним підходом для створення вебсервісів, які дозволяють взаємодіяти з сервером через Інтернет. Він базується на принципах архітектури REST, яка використовує HTTP протокол для передачі даних між клієнтом та сервером. RESTful API визначає набір операцій (або «ресурсів») та HTTP методів (GET, POST, PUT, DELETE тощо), які можуть бути використані для створення, зчитування, оновлення та видалення даних на сервері. Це дозволяє клієнтським додаткам легко спілкуватися з сервером та отримувати необхідні дані для відображення користувачу або виконання певних операцій.

У вебзастосунку, що розробляється, використовується JSON Web Tokens (JWT) для безпечної автентифікації та авторизації користувачів. JWT дозволяє створювати токени, які містять інформацію про користувача та його права, що дозволяє безпечно передавати цю інформацію між клієнтом та сервером. Під час обрання технології проведенно порівняння з іншим протоколом авторизації – OAuth 2.0 (табл. 3.8). OAuth 2.0 – це протокол авторизації, який використовується для надання доступу до ресурсів користувача стороннім додаткам без необхідності передавати їм логін та пароль. Основна ідея полягає в тому, щоб користувач міг надати дозвіл на доступ до своїх ресурсів (наприклад, дані акаунту) іншим додаткам без потреби надавати їм свої облікові дані.

Таблиця 3.8 – Порівняння JSON Web Tokens (JWT) та OAuth 2.0

Характеристика	**JSON Web Tokens**	**OAuth 2.0**
Тип	Токен авторизації та ідентифікації, який містить закодовану інформацію	Протокол авторизації, що використовує різні методи (наприклад, авторизацію через код, токени доступу тощо)
Загальна структура	Компактний формат, що містить заголовок, корисне навантаження та підпис	Протокол, який описує різні типи токенів та методи їх отримання
Захист від зміни	Підписаний та може бути зашифрований для захисту від зміни	Використовує HTTPS для захисту від перехоплення та зміни
Валідність	Може бути перевірений без необхідності звертатися до бази даних	Вимагається звернення до сервера аутентифікації для перевірки та отримання токенів доступу
Розширені можливості	Підтримує розширені можливості, такі як вказівка часу дії, обмін токенами тощо	Надає стандартний протокол авторизації, але потребує реалізації додаткових застосунків для обміну токенами

Кінець таблиці 3.8

Характеристика	JSON Web Tokens	OAuth 2.0
Переваги	Простота реалізації, легка перевірка, можливість передачі корисної інформації	Широкий спектр можливостей для різних типів авторизації та механізмів забезпечення безпеки

JSON Web Tokens (JWT) мають кілька переваг над OAuth 2.0, зокрема, щодо простоти реалізації, можливості передачі корисної інформації, а також можливості перевірки токенів без необхідності звертатися до сервера аутентифікації. Таким чином, в контексті багатьох застосувань JWT можуть бути більш практичним та зручним вибором для забезпечення безпеки та автентифікації. Алгоритм роботи JSON Web Tokens (JWT) можна розділити на наступні кроки:

1. Створення токена, який складається з трьох частин: заголовка (header), корисного навантаження (payload) та підпису (signature). В заголовку зазвичай вказується тип токена і алгоритм шифрування. У корисному навантаженні може міститися будь-яка корисна інформація, яку потрібно передати. Підпис генерується на основі заголовка і корисного навантаження з використанням секретного ключа.

2. Після створення токена він передається клієнту, який зазвичай зберігає його в локальному сховищі, наприклад, у кукісах або в localStorage.

3. Коли клієнт намагається отримати доступ до захищеного ресурсу (наприклад, за допомогою HTTP запиту), він включає токен у заголовку запиту, наприклад, в заголовку Authorization зі значенням «Bearer [токен]».

4. Сервер, який отримує запит, перевіряє токен. Перевірка включає перевірку підпису для визначення його валідності, а також перевірку терміну дії токена та інших параметрів, які можуть бути включені в корисне навантаження. Якщо токен валідний, сервер надає доступ до запитаного ресурсу, в іншому випадку він повертає помилку або вимагає авторизації.

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи магістра проведено детальний аналіз процесу проєктування вебзастосунку та огляд технологічного стеку, необхідного для реалізації цього проєкту. В рамках розділу була розглянута ієрархічна модель прогнозування станів системи, а також метод аналізу ієрархій, що дозволяє ефективно вирішувати проблеми прийняття рішень у складних системах. Крім того, в розділі проведено розробку UML-діаграм, зокрема Activity Diagram, Class Diagram та Sequence Diagram, які відображають структуру та взаємодію компонентів системи.

У контексті огляду технологій розглянуто використання різноманітних інструментів для реалізації back-end, front-end та бази даних. До back-end технологій включено Node.JS, Express, CORS та OpenAI API, що дозволяють створювати потужні та ефективні вебзастосунки. Для реалізації front-end використовуються такі технології, як React, React Query, React Redux, React Redux Toolkit, React Router та Tailwind CSS, що забезпечує гнучкість та зручність у розробці інтерфейсу користувача. Базу даних реалізується з використанням PostgreSQL. Для забезпечення безпеки вебзастосунку використовуються JSON Web Tokens (JWT), що забезпечує безпечну автентифікацію та авторизацію користувачів. Загальною метою цього розділу є обґрунтування вибору конкретних технологій та методів проєктування для успішної реалізації вебзастосунку, а також підготовка підґрунтя для подальшої реалізації та розвитку проєкту.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ТА ПРОВЕДЕННЯ ТЕСТУВАННЯ

4.1 UI/UX дизайн

UI/UX дизайн є ключовим елементом успіху будь-якого вебзастосунку, включаючи систему керування проєктами на основі засобів штучного інтелекту. Врахування цих принципів дозволяє створити продукт, який буде зручним та приємним у використанні для користувачів, забезпечуючи високий рівень задоволення від взаємодії з ним. Відповідно, це сприяє підвищенню конкурентоспроможності продукту на ринку та задоволенню потреб його цільової аудиторії.

UI (або інтерфейс користувача) – це частина програмного забезпечення або пристрою, яка дозволяє користувачам взаємодіяти з системою. Це може бути графічний інтерфейс, текстовий інтерфейс або будь-яка інша форма взаємодії, яка дозволяє користувачам виконувати дії та сприймати інформацію. UI включає в себе такі елементи, як кнопки, меню, поля введення, списки, вікна та інші компоненти, які дозволяють користувачам взаємодіяти з програмою або пристроєм. Головною метою UI є забезпечення зручності, ефективності та задоволення від використання продукту.

UX (або взаємодія з користувачем) – це концепція, яка описує враження, які користувач отримує під час взаємодії з продуктом, системою або послугою. Це включає в себе всі аспекти сприйняття та взаємодії користувача з продуктом, від першого знайомства до завершення використання. У UX дизайні враховується не лише інтерфейс користувача, але й всі інші аспекти, які впливають на враження та задоволення від користування продуктом. Це можуть бути такі фактори, як дизайн взаємодії, швидкість завантаження, зручність використання, доступність, якість виконання завдань та багато іншого. Метою UX дизайну є створення продуктів, які задовольняють потреби та очікування користувачів, забезпечуючи їм позитивний досвід взаємодії з продуктом. Це

сприяє підвищенню задоволення користувачів, збільшенню їх лояльності та покращенню репутації продукту або бренду.

Принципи UI/UX дизайну (рис. 4.1):

1. простота – інтерфейс повинен бути легким для розуміння і використання;
2. ясність – інформація повинна бути представлена зрозуміло і доступно;
3. консистентність – усі елементи дизайну повинні бути узгоджені і однакові для забезпечення єдності в інтерфейсі;
4. наявність зворотного зв’язку – користувач повинен отримувати повідомлення про свої дії та взаємодію з системою;
5. адаптивність – дизайн повинен адаптуватися до потреб і контексту користувача.

Рисунок 4.1 – Принципи UI/UX дизайну

Для спрощення розробки, уніфікації дизайну та документації дизайну системи керування проєктами на основі засобів штучного інтелекту використано UI kit (рис. 4.2–4.5). User interface kit (UI kit) – це набір готових компонентів, шаблонів і стилів, призначених для створення інтерфейсів користувача в вебзастосунках або мобільних додатках. UI kit спрощує процес

розробки, надаючи консистентний набір елементів, які можуть бути використані для створення кожної сторінки чи компоненту. Причини використання UI kit:

– дозволяє розробникам швидко створювати нові сторінки та компоненти, використовуючи готові елементи і шаблони;

– усі сторінки та компоненти мають спільний вигляд і стиль, що забезпечує єдність в інтерфейсі застосунку, особливо на це впливають обрані шрифти та колірна гама (рис. 4.2–4.3);

Рисунок 4.2 – Компоненти «Colors»

Рисунок 4.3 – Компоненти «Fonts»

– слугує головним дизайнерським документом проєкту, що полегшує спілкування між дизайнерами, верстальниками та розробниками. Всі учасники проєкту можуть використовувати UI kit як візуальний референс для створення та оцінки нових елементів і сторінок, як наприклад, при використанні групи значків з моностилем (рис. 4.4) та елементів (кнопки, блоки тощо), які створені з однаковим форматуванням (рис. 4.5).

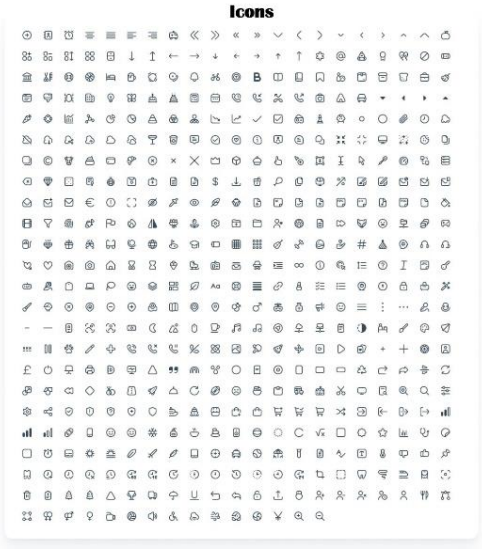

Рисунок 4.4 – Компоненти «Icons»

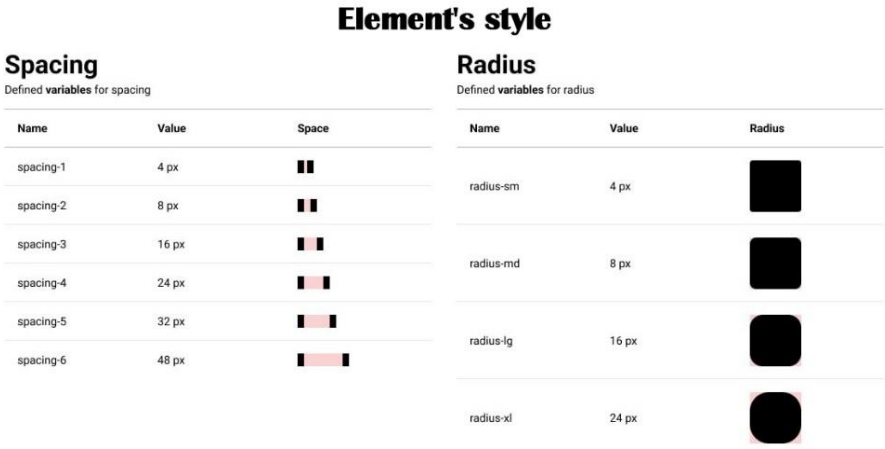

Рисунок 4.5 – Компоненти «Element’s style»

Для розробки системи в частині UI було використано інструмент Figma (рис. 4.6). Figma – це вебсервіс та програмне забезпечення для дизайну і прототипування, яке дозволяє командам спільно працювати над проєктами, створювати високоякісні інтерфейси користувача та здійснювати спільний аналіз та редагування дизайну в реальному часі [15].

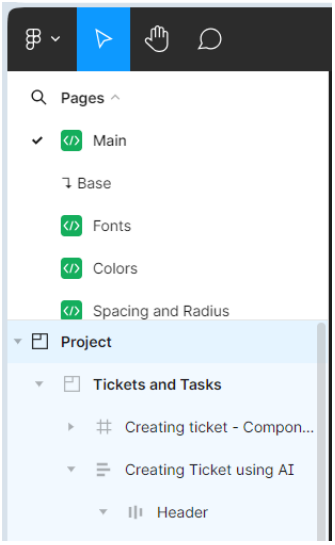

Рисунок 4.6 – Структура проєкту в Figma

Деякі з переваг використання Figma включають:

1. Дозволяє дизайнерам та розробникам спільно працювати над проєктами, незалежно від їх географічного розташування. Вони можуть одночасно редагувати документи, спілкуватися через коментарі та переглядати зміни в реальному часі.

2. Можна швидко створювати прототипи інтерфейсу користувача для тестування та оцінки концепцій (рис. 4.7). Це дозволяє командам швидше виявляти та виправляти проблеми ще на етапі розробки.

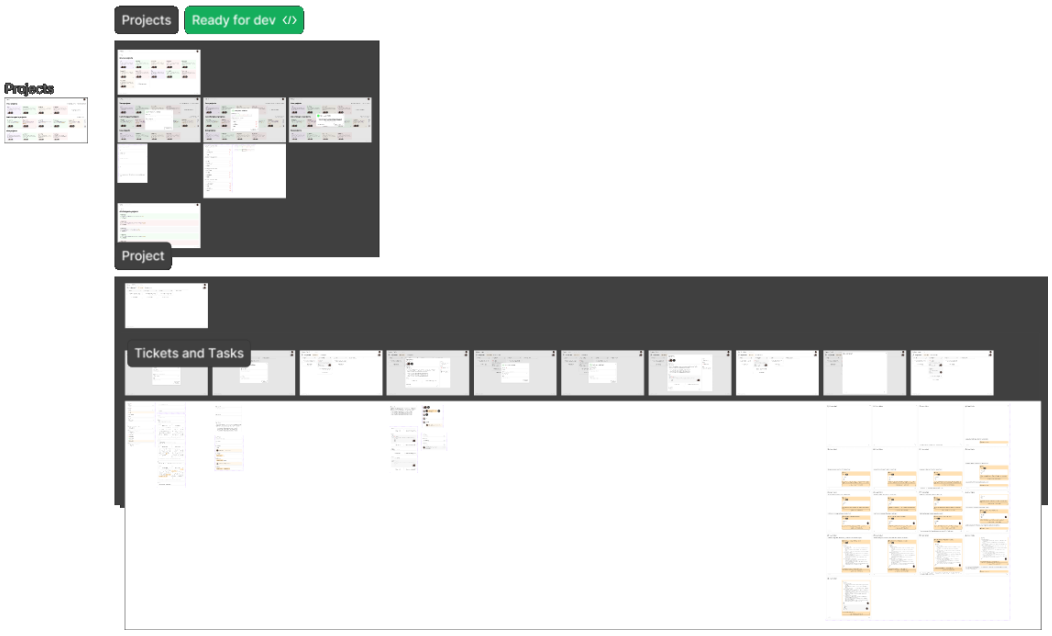

Рисунок 4.7 – Загальний вигляд проєкту

3. Автоматично генерує специфікації для розробки, що полегшує співпрацю між дизайнерами та розробниками. Це допомагає уникнути непорозумінь і помилок під час переходу від дизайну до розробки.

4. Може легко інтегруватися з іншими інструментами розробки, такими як Zeplin, Avocode, а також з системами керування версіями, що полегшує спільну роботу з іншими інструментами розробки.

5. Має дружній інтерфейс, який легко засвоюється навіть новачками в області дизайну. Він має широкий набір інструментів і можливостей, що дозволяє реалізувати різноманітні дизайн-концепції.

4.1 Реалізація програмних компонентів бізнес-логіки

Підрозділ включає в себе кілька ключових етапів, які є важливими для успішної розробки системи керування проєктами на основі засобів штучного інтелекту. Першим етапом є створення програмних компонентів, які відповідають за різноманітні функціональні аспекти системи, такі як створення, редагування та видалення проєктів, завдань, аналітика даних та інші. Ці компоненти є основою функціональності системи та визначають способи взаємодії користувача з нею.

Далі важливим етапом є інтеграція з штучним інтелектом. Це включає в себе впровадження засобів штучного інтелекту, які допомагають автоматизувати деякі процеси та покращити аналітику даних у системі. Наприклад, застосування моделей машинного навчання для прогнозування та оптимізації робочих процесів.

Останнім етапом є тестування та оптимізація. Після завершення розробки програмних компонентів важливо перевірити їхню працездатність та ефективність за допомогою тестування. Під час цього етапу виявляються та виправляються будь-які помилки або недоліки в роботі системи. Після цього можуть проводитися оптимізації для підвищення продуктивності та надійності

системи. Проте всі етапи є важливими для успішної реалізації програмних компонентів бізнес-логіки в системі та забезпечення її ефективної роботи.

4.2.1 Створення проєкту, додавання завдань та адаптивний інтерфейс

Вебзастосунок надає широкий спектр можливостей для управління проєктами, що дозволяє користувачам ефективно організовувати свою роботу та спілкуватися з командою. Основними функціями цього інструмента є створення проєктів з можливістю детального опису, призначення відповідальних осіб та встановлення термінів виконання. Крім того, вебзастосунок надає зручний інтерфейс для додавання нових завдань, визначення їх пріоритетів та категоризації за різними параметрами. Особливу увагу вебзастосунок приділяє керуванню статусами завдань, що дозволяє користувачам чітко визначати, на якому етапі знаходиться кожне завдання: від планування до завершення. Крім того, користувачі можуть легко вносити зміни у статус завдань та відстежувати їхній прогрес.

Важливим аспектом вебзастосунка є можливість спільної роботи над проєктами, що дозволяє командам спільно працювати над завданнями, обмінюватися інформацією та взаємодіяти в реальному часі. Такий підхід забезпечує збільшення ефективності та зменшення часу, необхідного для досягнення поставлених цілей [16]. Загалом, вебзастосунок є потужним інструментом для керування проєктами, що допомагає організувати робочий процес, встановлювати пріоритети та ефективно виконувати завдання в спільній команді.

Для створення проєкту виконується блок коду на JS (рис. 4.8), який забезпечує створення та збереження нового проєкту в БД, попередньо під’єднавшись до неї. Також він містить в собі частину, що використовується для обробки помилок, які можуть виникнути під час створення проєкту. Коли відбувається помилка, вона зловлюється оператором catch, і відповідний об’єкт помилки доступний у змінній error.

У цьому випадку, помилка виводиться до консолі за допомогою console.error, щоб розробник міг легко відстежити її. Після цього відправляється відповідь клієнту з статусом 500 (внутрішня помилка сервера) та об’єктом JSON, що містить повідомлення про помилку. Це дозволяє коректно обробляти помилки та повідомляти клієнта про те, що сталося під час обробки їхнього запиту на створення проєкту.

```
// Маршрут для створення нового проєкту
app.post('/projects', async (req, res) => {
  try {
    // Отримання даних з тіла запиту
    const { name, description, color, columns } = req.body;

    // Додавання проєкту до бази даних
    const result = await client.query('INSERT INTO projects (name, description, color, columns) VALUES (

    // Відправка відповіді з новим створеним проєктом та повідомленням про успішне створення
    res.status(201).json(result.rows[0]);
    console.log('Your Project successfully created!');
  } catch (error) {
    console.error('Error creating project:', error);
    res.status(500).json({ error: 'Failed to create project' });
  }
});
```

Рисунок 4.8 – Блок коду для створення проєктів

Таким чином код містить маршрут /projects, який обробляє POST-запити для створення нового проєкту. Дані проєкту (назва, опис, колір, колонки) передаються у вигляді JSON-об’єкта у тілі запиту. Після успішного створення проєкту відповідь містить створений проєкт у форматі JSON, а користувачу виводиться повідомлення «Your Project successfully created!», який свідчить про успішне створення нового проєкту. Результатом є форма створення нового проєкту (рис. 4.9).

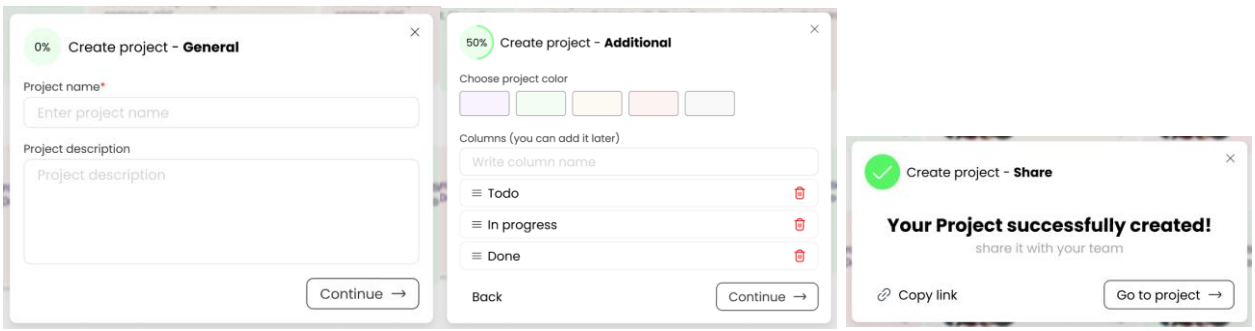

Рисунок 4.9 – Послідовність створення нового проєкту

Після створення проєкту необхідно додавати завдання учасникам, для чого використовується блок коду (рис. 4.10), який додає завдання до певної з існуючих колонок в проєкті, також очікує POST-запити на шлях /projects/:projectId/tasks, де :projectId – це параметр, який вказує на ідентифікатор проєкту, до якого потрібно додати завдання. Дані завдання (колонка, назва, пріорітет, дата, опис, учасники) передаються у вигляді JSON-об’єкта у тілі запиту.

Код реагує на успішне додавання завдання, відправляючи клієнту статус 201 (Створено) та об’єкт JSON з даними нового завдання. У разі помилки відправляється статус 500 (Внутрішня помилка сервера) та об’єкт JSON з повідомленням про помилку. Тепер можна використовувати цей маршрут для додавання нових завдань до певного проєкту відповідно до вказаних у завданні параметрів.

```
app.post('/projects/:projectId/tasks', async (req, res) => {
  try {
    // Отримання ідентифікатора проєкту та даних завдання з тіла запиту
    const projectId = req.params.projectId;
    const { column, title, priority, date, description, participants } = req.body;

    // Додавання завдання до бази даних
    const result = await client.query('INSERT INTO tasks (project_id, column, title, priority, date,

    // Відправка відповіді з новим створеним завданням
    res.status(201).json(result.rows[0]);
  } catch (error) {
    console.error('Error creating task:', error);
    res.status(500).json({ error: 'Failed to create task' });
  }
});
```

Рисунок 4.10 – Блок коду для створення завдання

Виконання коду забезпечує коректну роботу функції «Створити завдання» (рис. 4.11), після заповнення всіх необхідних даних завдання додається до обраного блоку.

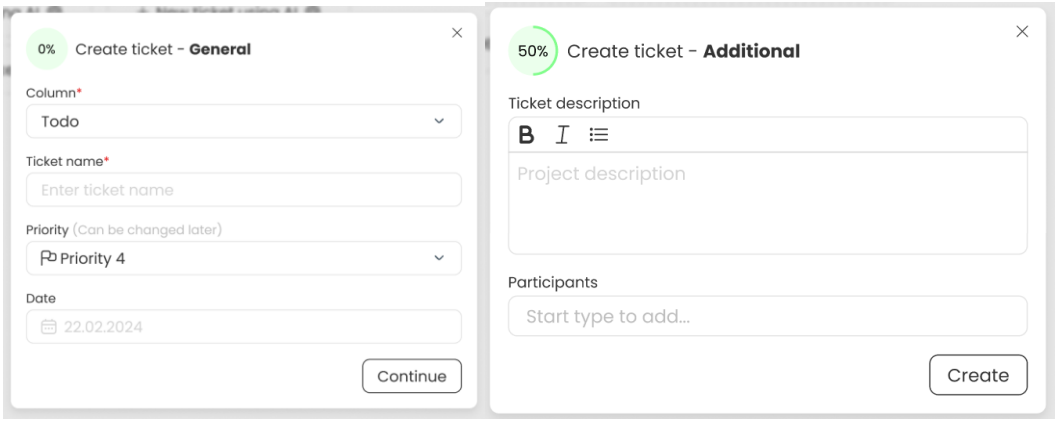

Рисунок 4.11 – Створення завдання

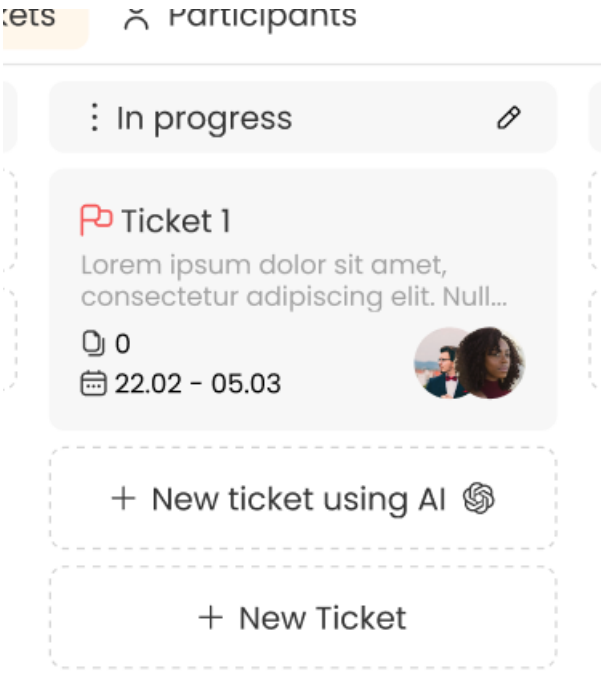

Рисунок 4.12 – Створене завдання в блоці «In progress»

Найкращим варіантом для забезпечення адаптивності інтерфейсу в даному контексті є використання Responsive Design з Tailwind CSS. Tailwind CSS надає простий і потужний спосіб створення адаптивного дизайну за допомогою готових класів, що відповідають розмірам екрану (рис. 4.13). У Tailwind CSS існує набір зростаючих розмірів екранів, які використовуються для визначення адаптивних стилів:

- sm: Для екранів шириною більше або рівно 640 пікселів;
- md: Для екранів шириною більше або рівно 768 пікселів;

- lg: Для екранів шириною більше або рівно 1024 пікселів;
- xl: Для екранів шириною більше або рівно 1280 пікселів.

```
import React from 'react';

const Button = () => {
  return (
    <button className="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded-lg
                       md:bg-green-500 md:hover:bg-green-700 lg:bg-red-500 lg:hover:bg-red-700">
      Click me
    </button>
  );
}

export default Button;
```

Рисунок 4.13 – Використання Tailwind CSS для адаптивного оформлення кнопки

У цьому прикладі, за допомогою класів Tailwind CSS, задано різні стилі кнопки в залежності від розміру екрану:

- «bg-blue-500 hover:bg-blue-700»: стилі для екранів менш ніж md;
- «md:bg-green-500 md:hover:bg-green-700»: стилі для екранів md і більше;
- «lg:bg-red-500 lg:hover:bg-red-700»: стилі для екранів lg і більше.

Це дозволяє автоматично адаптувати оформлення кнопки для різних розмірів екрану без необхідності вручній настройки CSS медіа-запитів.

4.2.2 Інтеграція штучного інтелекту у вебзастосунок

Для реалізації аналізу (рис. 4.14) з використанням штучного інтелекту за допомогою OpenAI у вебзастосунку «Система керування проєктами на основі засобів штучного інтелекту», можна використати OpenAI API для виконання запитів на аналіз даних. Наприклад, можна використати засоби OpenAI для аналізу текстової інформації про статуси проєктів, прийняття рішень щодо стратегій управління проєктами або прогнозування можливих ризиків.

```
app.get('/analyze', async (req, res) => {
    try {
        const analysisResult = await openai.analyzeData(req.query.data);
        res.json(analysisResult);
    } catch (error) {
        console.error("Помилка під час аналізу даних:", error);
        res.status(500).json({ error: "Помилка під час аналізу даних" });
    }
});
```

Рисунок 4.14 – Блок коду на JS для використання ШІ з метою аналізу даних

Для візуалізації (рис. 4.15) даних можна використати бібліотеку Plotly.js, яка дозволить побудувати різноманітні графіки та діаграми для відображення аналізованих даних. Наприклад, можна побудувати діаграму Ганта для відображення термінів виконання різних етапів проєкту або кругову діаграму для відображення розподілу ресурсів між різними завданнями.

```
app.get('/visualize', (req, res) => {
    try {
        const figure = {
            data: [{ x: [1, 2, 3], y: [2, 4, 6], type: 'scatter' }],
            layout: { title: 'Графік з Plotly.js' }
        };

        plotly.getImage(figure, { format: 'png', width: 800, height: 600 }, (err, imageStream) => {
            if (err) {
                console.error("Помилка під час створення графіку:", err);
                res.status(500).send("Помилка під час створення графіку");
            } else {
                res.writeHead(200, { 'Content-Type': 'image/png' });
                imageStream.pipe(res);
            }
        });
    } catch (error) {
        console.error("Помилка під час візуалізації даних:", error);
        res.status(500).json({ error: "Помилка під час візуалізації даних" });
    }
});
```

Рисунок 4.15 – Блок коду на JS для використання ШІ з метою візуалізації даних

Для реалізації чату з OpenAI для створення завдань (тасків у проєкті) в системі керування проєктами на основі засобів штучного інтелекту, можна використати API OpenAI для генерації тексту на основі введеного користувачем

контексту. Припустимо, що користувач хоче створити нове завдання в проєкті через чат. Він може ввести опис завдання, а система, використовуючи OpenAI, може розпізнати текст та згенерувати відповідний текстовий варіант завдання.

Наприклад, якщо користувач вводить: «Створити нову сторінку для додавання завдань в проєкт», система може використати OpenAI для генерації тексту, який буде представляти собою опис цього завдання. Потім цей згенерований текст може бути відображений користувачеві як нове завдання, а користувач може підтвердити створення цього завдання. Щоб реалізувати це в коді, можна використати JavaScript разом з бібліотеками React та Axios для взаємодії з API OpenAI (рис. 4.16).

```
const TaskChat = () => {
  const [userInput, setUserInput] = useState('');
  const [generatedTask, setGeneratedTask] = useState('');

  const generateTask = async () => {
    try {
      const response = await axios.post(
        'https://api.openai.com/v1/completions',
        {
          model: 'text-davinci-003',
          prompt: userInput,
          max_tokens: 50 // Максимальна кількість токенів для генерації
        },
        {
          headers: {
            'Content-Type': 'application/json',
            'Authorization': 'Bearer YOUR_OPENAI_API_KEY'
          }
        }
      );
      setGeneratedTask(response.data.choices[0].text.trim());
    } catch (error) {
      console.error('Error generating task:', error);
    }
  };
```

Рисунок 4.16 – Патерн створення завдання
з використання ШІ (без ключів)

У коді не вказано власний ключ (токен) для безпеки та конфіденційності даних. Це зроблено для того, щоб не розголошувати особисту інформацію в публічному середовищі. Зазвичай власний ключ OpenAI API зберігається в змінній середовища або файлі конфігурації, який не включений до репозиторію

коду. Це дозволяє уникнути можливого витоку ключа та незаконного використання API.

У коді використовується компонент React, який має текстове поле для введення користувачем тексту та кнопку для генерації завдання. Після введення користувачем тексту і натискання кнопки, викликається функція generateTask, яка відправляє запит на API OpenAI з введеним текстом. Потім згенерований текст завдання відображається користувачеві.

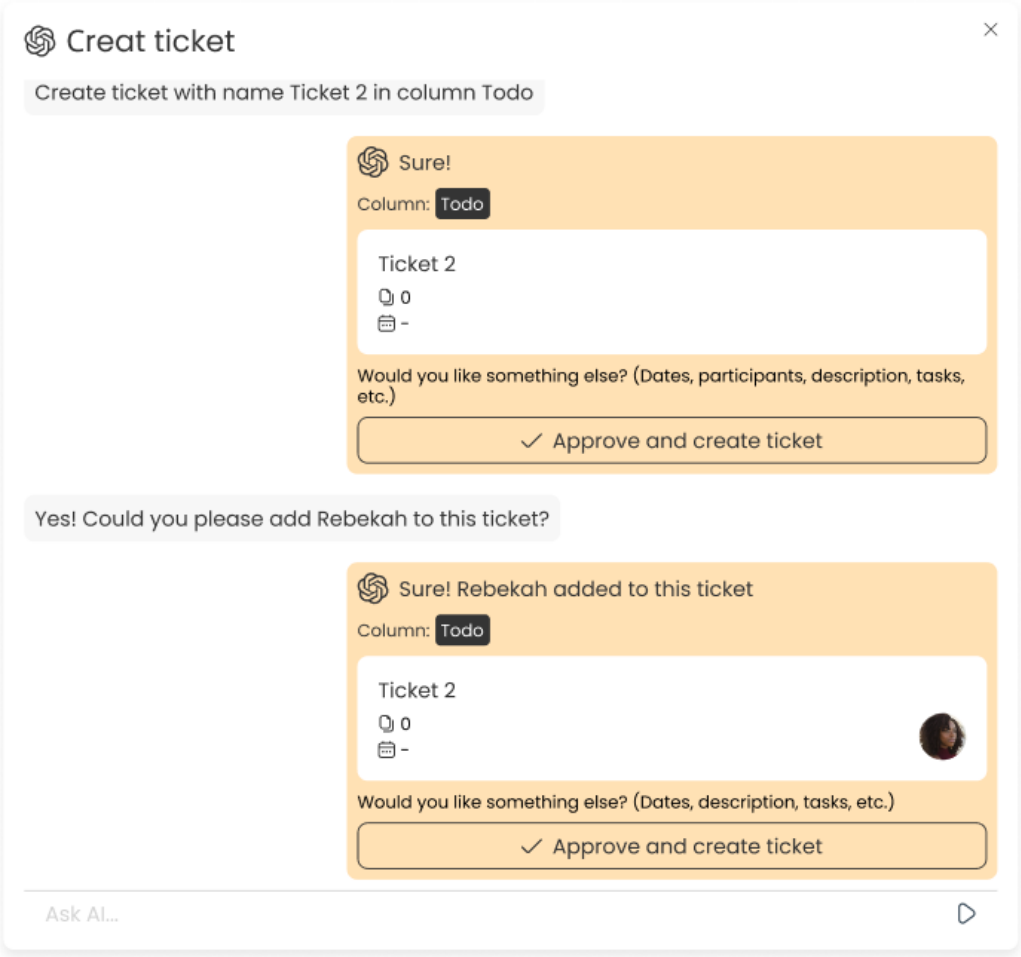

Рисунок 4.17 – Вигляд чату у вебзастосунку

Цей чат може бути зручним для користувача, який в команді виконує роль проєктного менеджера, з декількох причин. По-перше, він дозволяє швидко та зручно створювати нові завдання в проєкті, просто вводячи опис у чаті. Це полегшує процес створення та надання відомостей щодо завдань.

По-друге, використання OpenAI для генерації тексту може допомогти швидко сформулювати опис завдання на основі введеного користувачем

контексту, що збільшує ефективність комунікації. Крім того, цей чат може бути зручним для проєктного менеджера, оскільки він інтегрований безпосередньо в систему керування проєктами [17]. Це означає, що створені завдання автоматично додаються до системи, що спрощує процес ведення проєкту та забезпечує актуальність інформації для всіх учасників команди.

4.2.3 Тестування вебзастосунку

У сучасному світі програмне забезпечення стало неодмінною складовою практично кожної галузі, від інформаційних технологій до медицини та фінансів [18]. За відсутності належного тестування програмного продукту можуть виникнути серйозні проблеми, що можуть негативно вплинути на користувачів та підірвати довіру до компанії. Тестування застосунку виявляється надзвичайно важливим етапом у процесі розробки програмного забезпечення, оскільки забезпечує відповідність продукту вимогам, його надійність та ефективність, а також сприяє виявленню та усуненню помилок на ранніх етапах розробки.

Тестування застосунку є надзвичайно важливим етапом в процесі розробки програмного забезпечення з ряду ключових причин:

1. забезпечення якості продукту;
2. виявлення помилок та усунення дефектів;
3. забезпечення надійності та стабільності;
4. виявлення та усунення проблем на ранніх стадіях розробки дозволяє уникнути витрат часу та грошей на виправлення дефектів у подальшому;
5. забезпечення сумісності та взаємодії з іншими системами.

В цілому, тестування є ключовим елементом у забезпеченні успішної та ефективної розробки програмного забезпечення. Крім того, допомагає забезпечити якість, надійність та стабільність продукту, що є критичними факторами для задоволення потреб користувачів та досягнення успіху на ринку.

Unit tests – це вид тестів у програмуванні, який використовується для перевірки окремих одиниць програмного коду, таких як функції, методи або

класи. Основна ідея полягає в тому, щоб перевірити, чи працює кожна одиниця програми коректно та відповідає вимогам специфікації, незалежно від інших частин системи. Unit tests допомагають виявляти помилки в програмному коді на ранніх етапах розробки і забезпечують більшу стабільність та надійність програмного забезпечення [19].

У системі керування проєктами на основі засобів штучного інтелекту використання unit tests є критично важливим з точки зору забезпечення якості програмного забезпечення та надійності системи. Unit tests дозволяють перевірити коректність окремих компонентів системи, таких як функції для реєстрації користувачів, створення проєктів чи завдань, безпосередньо на ранніх етапах розробки. Вони допомагають виявляти помилки та дефекти в програмному коді, що можуть призвести до неправильної роботи системи або виникнення проблем для користувачів.

Крім того, unit tests допомагають підтримувати стабільність системи під час внесення змін або рефакторингу коду, переконуючись, що існуючі функціональності не порушуються під час вдосконалення або розширення системи. Таким чином, використання unit tests є необхідним елементом для забезпечення якості, стабільності та надійності системи керування проєктами.

Jest – це популярна бібліотека для тестування функцій JavaScript, яка часто використовується для написання unit tests [20]. А також, надає ряд корисних функцій і можливостей для створення та виконання тестів, включаючи підтримку звичайних тестових умов, визначення функцій перевірки очікуваного результату, а також розширення можливостей за допомогою плагінів та розширень. Тестування окремих (базових) функцій застосунку наведено в табл. 4.1.

Таблиця 4.1 – Опис функцій та тестових випадків для операцій у системі

Функція	**Опис**	**Тестовий випадок**	**Очікуваний результат**
User Registration	Ця функція перевіряє можливість реєстрації нового користувача у системі.	Перевіряється, чи успішно можливо зареєструвати нового користувача з введеними даними (ім'я користувача та пароль).	Успішна реєстрація користувача у системі.
User Authentication (рис. 4.18)	Ця функція перевіряє можливість авторизації користувача з введеними даними.	Перевіряється, чи можливо авторизувати користувача з введеними правильними даними для входу (ім'я користувача та пароль).	Успішна авторизація користувача та отримання доступу до системи.
Project CRUD Operations (рис. 4.19)	Операції CRUD для проєктів		Успішне створення, оновлення або видалення проєкту у системі.
	createProject: Створює новий проєкт у системі.	Перевіряється, чи успішно можливо створити новий проєкт у системі.	
	updateProject: Оновлює інформацію про існуючий проєкт у системі.	Перевіряється, чи можливо оновити інформацію про існуючий проєкт у системі.	
	deleteProject: Видаляє проєкт з системи.	Перевіряється, чи можливо видалити проєкт з системи.	
Task CRUD Operations (рис. 4.20)	Операції CRUD для завдань		Успішне створення, оновлення або видалення завдання у системі.
	createTask: Створює нове завдання в рамках проєкту.	Перевіряється, чи можливо створити нове завдання у системі.	
	updateTask: Оновлює інформацію про існуюче завдання.	Перевіряється, чи можливо оновити інформацію про існуюче завдання у системі.	
	deleteTask: Видаляє завдання з системи.	Перевіряється, чи можливо видалити завдання з системи.	

```
// Тестування авторизації користувача
describe('User Authentication', () => {
  test('Should authenticate a user with valid credentials', () => {
    // Описуємо тестовий випадок
    expect(authenticateUser('username', 'password')).toEqual('success');
  });
});
```

Рисунок 4.18 – Шаблон коду для тестування функції авторизації

```
// Тестування операцій CRUD для проєктів
describe('Project CRUD Operations', () => {
  test('Should create a new project', () => {
    // Описуємо тестовий випадок
    expect(createProject('projectData')).toEqual('success');
  });

  test('Should update an existing project', () => {
    // Описуємо тестовий випадок
    expect(updateProject('projectId', 'updatedProjectData')).toEqual('success');
  });

  test('Should delete an existing project', () => {
    // Описуємо тестовий випадок
    expect(deleteProject('projectId')).toEqual('success');
  });
});
```

Рисунок 4.19 – Шаблон коду для тестування функцій CRUD-операцій над проєктами

```
// Тестування операцій CRUD для завдань
describe('Task CRUD Operations', () => {
  test('Should create a new task', () => {
    // Описуємо тестовий випадок
    expect(createTask('projectId', 'taskData')).toEqual('success');
  });

  test('Should update an existing task', () => {
    // Описуємо тестовий випадок
    expect(updateTask('taskId', 'updatedTaskData')).toEqual('success');
  });

  test('Should delete an existing task', () => {
    // Описуємо тестовий випадок
    expect(deleteTask('taskId')).toEqual('success');
  });
});
```

Рисунок 4.20 – Шаблон коду для тестування функцій CRUD-операцій над завданнями

Після проведення всіх тестів в системі керування проєктами на основі засобів штучного інтелекту, ми підтвердили коректність роботи всіх функцій програмного забезпечення. Усі тести були успішно завершені, що підтверджує правильність реалізації кожної окремої функціональної можливості [21]. Крім того, під час проведення тестування були виявлені деякі помилки, які були виправлені згідно з вимогами. Це дозволило забезпечити стабільність та надійність програмного забезпечення, а також гарантує безперебійну роботу системи для користувачів. Завдяки повному охопленню тестуванням усіх

аспектів програми, ми можемо з упевненістю стверджувати, що система готова до продуктивного використання та може надійно функціонувати в реальних умовах експлуатації.

4.3 Керівництво користувача

Керівництво користувача – це документ, який надає користувачам інформацію про те, як використовувати програмне забезпечення або систему. Воно включає в себе опис інтерфейсу, інструкції з використання, пояснення функціональності та можливостей програми, а також поради щодо вирішення проблем. У системі керування проєктами на основі засобів штучного інтелекту, керівництво користувача включає інформацію про те, як реєструватися та авторизуватися в системі, як створювати нові проєкти, додавати та видаляти завдання тощо.

Система передбачає роботу тільки з авторизованими користувачами, тому перший етап використання системи полягає в реєстрації або авторизації (рис. 4.21). Без відповідного входу до системи користувачам не надається доступ до її функціональності [22]. Після успішної авторизації користувач отримує доступ до всіх можливостей системи та може приступити до роботи з нею. Такий підхід до роботи забезпечує безпеку і конфіденційність даних, а також забезпечує інтегровану та персоналізовану роботу для кожного користувача.

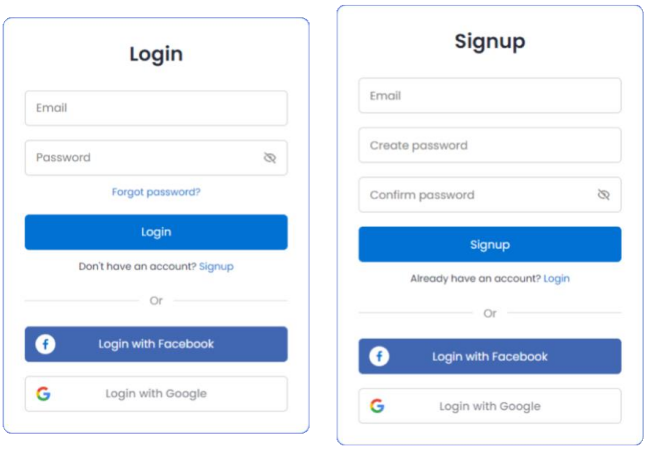

Рисунок 4.21 – Вигляд форм авторизації та реєстрації користувачів

Після успішної аутентифікації користувача системою, сторінка за замовчуванням, яка відображається, – це сторінка «Проєкти» (рис. 4.22). На цій сторінці користувач може переглянути список вже створених проєктів, до яких він має доступ, або створити новий проєкт, якщо потрібно. Це забезпечує зручний доступ до основного функціоналу системи та дозволяє користувачам швидко розпочати роботу з проєктами або створити нові, використовуючи інтуїтивно зрозумілий інтерфейс.

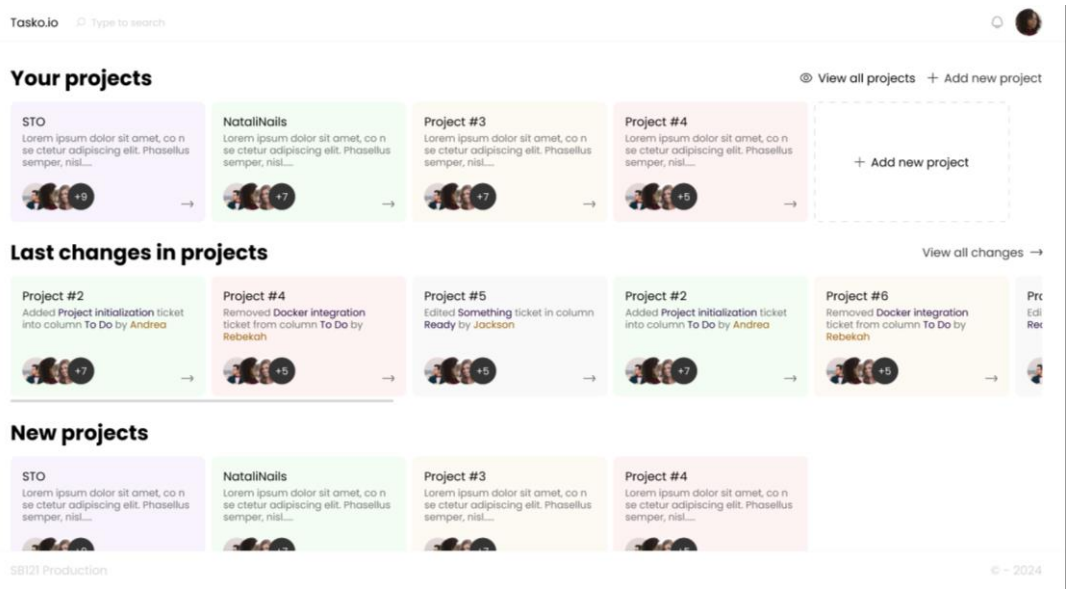

Рисунок 4.22 – Сторінка «Проєкти»

Процес створення проєкту та завдання описаний в підрозділі 4.2.1. Для створення проєкту (рис. 4.9) необхідно виконати наступні етапи та надати наступні дані:

- введення назви проєкту;
- вибір збереження та доступності проєкту;
- додавання опису проєкту.

Для створення завдання (рис. 4.11) потрібно виконати наступні кроки:

- введення назви завдання;
- вибір пріоритету та терміну виконання;
- додавання опису завдання;
- призначення відповідального користувача.

Після створення завдання можна переглянути його вигляд (рис. 4.12), де будуть відображені всі введені дані, включаючи назву, опис, пріоритет, термін виконання та інші характеристики. Це дозволяє користувачам з легкістю створювати та відстежувати завдання у межах своїх проєктів.

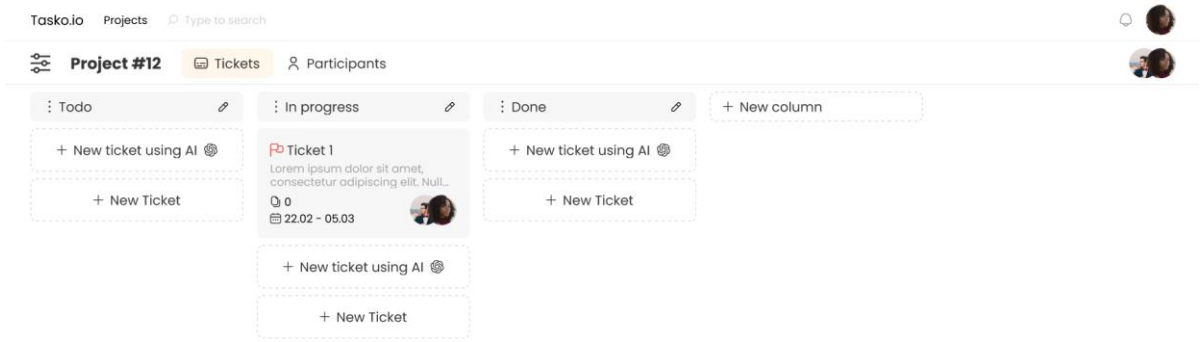

Рисунок 4.23 – Вигляд сторінки певного проєкту

За потребою, система також надає сторінку «Сповіщення» (рис. 4.24), яка містить інформацію про зміни в проєктах. На цій сторінці користувач може переглядати сповіщення про оновлення, коментарі, зміни статусів завдань та інші важливі події, що відбуваються у проєктах, до яких він має доступ. Для того щоб перейти на цей розділ, користувач може скористатись символом дзвіночка, який зазвичай розміщується у верхньому меню або панелі навігації. Використання сторінки «Сповіщення» дозволяє користувачам оперативно відстежувати всі важливі події та оновлення у їхніх проєктах.

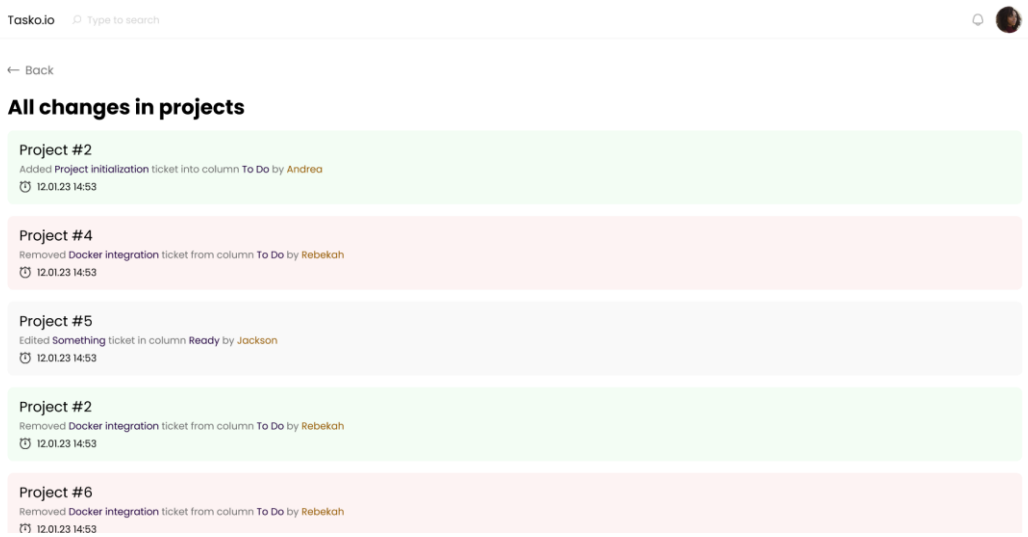

Рисунок 4.24 – Сторінка «Сповіщення»

Система надає можливість детального перегляду кожного завдання окремо (рис. 2.25). Користувач може отримати додаткову інформацію про кожне завдання, таку як назва, опис, призначений користувач, статус завдання, термін виконання та інші характеристики. Ця функція дозволяє користувачам отримати більш детальне уявлення про кожне завдання, щоб ефективно керувати ними та забезпечувати їх вчасне виконання. Для перегляду деталей кожного завдання користувач може клікнути на назву або ідентифікатор завдання, що відображається у списку завдань на сторінці проєкту.

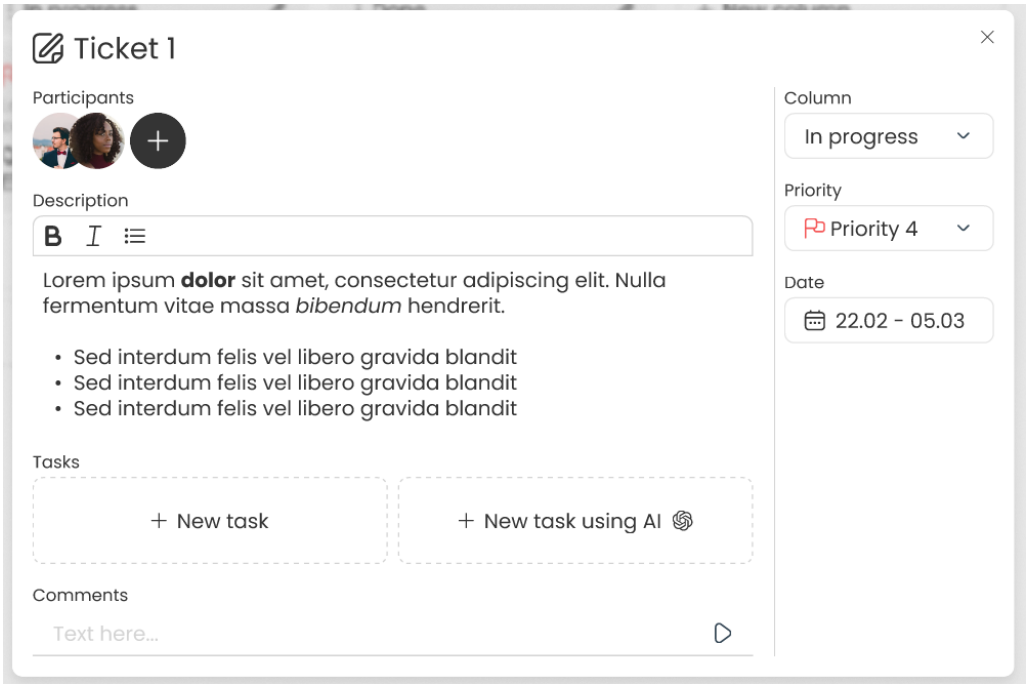

Рисунок 2.25 – Перегляд створенного завдання

Система також надає можливість використання чату з штучним інтелектом (рис. 2.26) для створення нових завдань. Ця функція дозволяє користувачам надсилати текстові повідомлення з описом завдання через чатовий інтерфейс, а ШІ в системі автоматично розпізнає текст та створює нове завдання на основі наданої інформації [23].

Наприклад, користувач може написати у чаті: «Створити завдання: Підготувати презентацію для наступного зустрічі з клієнтом». ШІ системи автоматично розпізнає це як інструкцію для створення нового завдання та створює його з необхідними деталями, такими як назва завдання та опис. Ця

функція полегшує процес створення нових завдань, забезпечуючи швидкий та зручний спосіб додавання їх у систему.

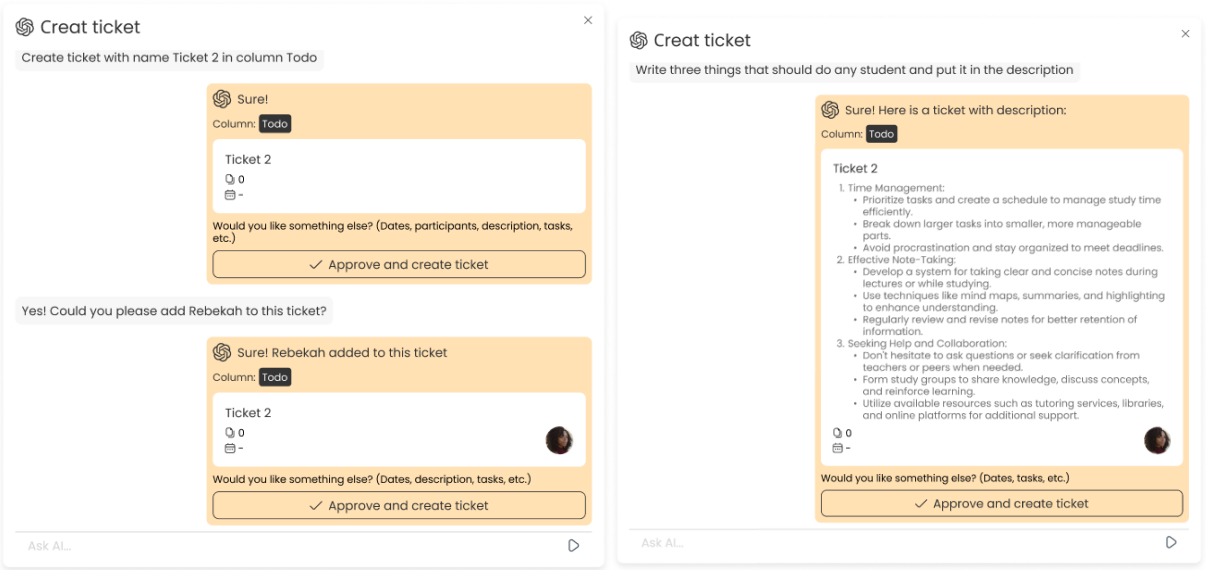

Рисунок 2.26 – Створення завдань з використанням ШІ

Завершуючи аналіз керівництва користувача для системи керування проєктами, можна зробити висновок, що вона повністю відповідає вимогам, що ставляться перед такими системами. Вона надає користувачам широкий функціонал для організації та управління проєктами, включаючи створення, оновлення та видалення проєктів та завдань, обмін даними між користувачами, створення сповіщень та багато іншого.

Оптимізація системи за рахунок використання штучного інтелекту також грає важливу роль у покращенні її ефективності. Використання чату з штучним інтелектом дозволяє користувачам швидко та зручно створювати нові завдання та спілкуватися з системою, а аналіз та візуалізація даних з штучного інтелекту допомагає зрозуміти та використовувати інформацію з системи для прийняття кращих управлінських рішень. Отже, система керування проєктами, оптимізована за рахунок використання штучного інтелекту, відповідає всім сучасним вимогам та забезпечує користувачам ефективний та зручний інструмент для управління проєктами.

Висновки до розділу 4

У четвертому розділі магістерської кваліфікаційної роботи зосереджено увагу на реалізації концепції, яка була розроблена та описана в попередніх розділах. Проведено програмну реалізацію вебзастосунку, що передбачала перетворення концептуальних ідей у функціональний програмний продукт. Цей етап включав в себе написання коду для реалізації різних частин системи, таких як користувацький інтерфейс, бізнес-логіка та зв’язок з базою даних.

У першому підрозділі розділу 4 було здійснено дизайн інтерфейсу користувача (UI/UX дизайн), що включало в себе розробку зручного та привабливого вигляду вебзастосунку з метою покращення користувацького досвіду. У другому підрозділі реалізована програмна логіка вебзастосунку, включаючи створення проєкту, додавання завдань та розробку адаптивного інтерфейсу для забезпечення його коректної роботи на різних пристроях та розмірах екранів. Також у цьому підрозділі проведена інтеграція штучного інтелекту у вебзастосунок з використанням відповідних інструментів та API.

У третьому підрозділі проведено тестування вебзастосунку з метою виявлення помилок, а також перевірки відповідності вимогам та очікуванням користувачів. Для цього використовувалися різноманітні методи тестування, такі як модульне тестування, інтеграційне тестування та функціональне тестування. Завершальним етапом в розділі є керівництво користувача, де надаються відповіді на питання щодо використання вебзастосунку, його функціональності та особливостей. У цілому, четвертий розділ є ключовим у процесі розробки та впровадження вебзастосунку, оскільки саме тут відбувається його фактична створення, тестування та визначення способу взаємодії з користувачем.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи магістра здійснено обширний аналіз предметної області, який включав огляд методологій керування проєктами, аналіз існуючих систем та застосунків-аналогів, а також вивчення методів штучного інтелекту в контексті управління проєктами. Цей аналіз став основою для подальшого проектування та розробки системи керування проєктами з використанням засобів штучного інтелекту.

На етапі моделювання системи використано різні типи моделей: ERD, DFD, IDEF0, Use case, а також діаграми розгортання, що сприяло кращому розумінню структури та функціоналу системи керування проєктами. Основні завдання вирішено через систематичний підхід та використання передових методів розробки програмного забезпечення. Розроблено специфікацію вимог до системи, описано алгоритм роботи програмного забезпечення, побудовано архітектуру програмного комплексу та створено моделі системи за допомогою UML-діаграм.

У четвертому розділі проведена програмна реалізація вебзастосунку та його тестування для перевірки функціональності та коректності роботи системи. Це був ключовий етап, що вимагав відповідності розробленої системи вимогам та очікуванням користувачів.

Для досягнення визначеної мети вирішено поставлені завдання:

– оглянуто існуючі методології керування проєктами;

– проведено аналіз сучасних систем керування проєктами, що охоплює вивчення існуючих підходів систем до керування проєктами, їх переваги і недоліки;

– проаналізовано систему, що розробляється, описано структури та функціоналу;

– оглянуто методи ШІ та їх застосування в керуванні проєктами;

– сформувано специфікацію вимог до програмного забезпечення;

– описано етапи реалізації проєкту;

– змоделювано систему з графічним представленням моделей даних;
– описано та візуалізовано алгоритми роботи програмного забезпечення;
– побудовано архітектуру програмного комплексу;
– створено моделі системи, які деталізують структуру та функціонал системи керування проєктами, зокрема, використанням UML-діаграм;
– оглянуто дизайн вебзастосунку, включаючи UI/UX;
– реалізовано програмні компоненти бізнес-логіки;
– проведено тестування та впровадження вебзастосунку.

Загалом, виконана робота відображає глибоке розуміння предметної області, систематичний підхід до розробки програмного забезпечення та вміння використовувати передові технології. Результатом є створення функціонального вебзастосунку для керування проєктами з використанням методів штучного інтелекту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Jira | Issue & Project Tracking Software | Atlassian. Collaboration software for software, IT and business teams. URL: https://www.atlassian.com/software /jira (дата звернення: 12.11.2023).

2. Manage Your Team’s Projects From Anywhere | Trello. URL: https://trello.com (дата звернення: 13.11.2023).

3. Manage your team’s work, projects, & tasks online. Asana. URL: https://asana.com (дата звернення: 14.11.2023).

4. Гаршіна К. А. Застосунок для керування проєктами : bachelor’s thesis. 2020. 65 с. URL: https://ela.kpi.ua/handle/123456789/40902 (дата звернення: 18.10.2023).

5. Кизименко І., Гусєва Н. Штучний інтелект: філософія розумних машин. Theoretical and practical aspects of modern scientific research. 2023. URL: https://doi.org/10.36074/logos-28.04.2023.55 (дата звернення: 18.10.2023).

6. Овецька О. В., Кукудяк Н. В. Управління проєктами: стан та перспективи розвитку підприємства. Економіка і організація управління. 2022. № 2. С. 139–147. URL: https://doi.org/10.31558/2307-2318.2022.2.14 (дата звернення: 18.10.2023).

7. Karkeraa S. Use Cases. Unlocking Blockchain on Azure. Berkeley, CA, 2020. С. 229–266. URL: https://doi.org/10.1007/978-1-4842-5043-3_9 (дата звернення: 18.10.2023).

8. Пчелянський Д. П., Воінова С. А. Штучний інтелект: перспективи та тенденції розвитку. Automation of technological and business processes. 2019. Т. 11, № 3. С. 59–64. URL: https://doi.org/10.15673/atbp.v11i3.1500 (дата звернення: 18.10.2023).

9. Таламанова І. С. Дослідження авторегресійних й нейромережевих моделей для короткострокового прогнозування забруднення повітря. ElAr : Головна. URL: https://openarchive.nure.ua/server/api/core/bitstreams/9d6692b2-fb6f-4092-80cc-88b8b670debb/content (дата звернення: 18.10.2023).

10. Метод аналізу ієрархій - Система підтримки прийняття рішень IT-компанії "Тріумф IT". Система підтримки прийняття рішень IT-компанії "Тріумф IT". URL: https://dss.tg.ck.ua/ahp-help (дата звернення: 11.10.2023).

11. What is Unified Modeling Language. Lucidchart. URL: https://www.lucidchart.com/pages/what-is-UML-unified-modeling-language (дата звернення: 21.02.2024).

12. Class Diagram. Monographs in Computer Science. New York, NY. С. 43–61. URL: https://doi.org/10.1007/0-387-23803-4_3 (дата звернення 17.10.2023).

13. Maxym Z. Діаграма послідовності (Sequence Diagrams). Maxym Zosym. URL: https://www.maxzosim.com/sequence-diagrams/ (дата звернення: 21.10.2023).

14. OpenAI API. OpenAI. URL: https://openai.com/blog/openai-api (дата звернення: 21.02.2024).

15. Figma: The Collaborative Interface Design Tool. Figma. URL: https://www.figma.com/ (дата звернення: 14.10.2023).

16. Рижков Е., Синиціна Ю., Станіна О. Штучний інтелект: що змінилося за 50 років. Theoretical foundations of engineering. Tasks and problems. 2021. С. 341–348. URL: https://doi.org/10.46299/isg.2021.mono.tech.iii.6.2 (дата звернення: 18.10.2023).

17. Системи управління проєктами : thesis / Ю. В. Костинець та ін. 2020. URL: https://er.knutd.edu.ua/handle/123456789/16924 (дата звернення: 18.10.2023).

18. Томачинський С., Ковальова Н., Адлер О. Штучний інтелект, як винахідник: стан, проблеми, перспективи розвитку. Expert: paradigm of law and public administration. 2021. Т. 17, № 5. С. 103–110. URL: https://doi.org/10.32689/2617-9660-2021-5(17)-103-110 (дата звернення: 18.10.2023).

19. Testim. What Is the Best Unit Testing Framework for JavaScript?. AI-driven E2E automation with code-like flexibility for your most resilient tests.

URL: https://www.testim.io/blog/best-unit-testing-framework-for-javascript/ (дата звернення: 01.12.2023).

20. Jest. Delightful JavaScript Testing. URL: https://jestjs.io (дата звернення: 06.12.2023).

21. Testing JavaScript | IntelliJ IDEA. IntelliJ IDEA Help. URL: https://www.jetbrains.com/help/idea/unit-testing-javascript.html#running-and-debugging-tests (дата звернення: 06.12.2023).

22. Томачинський С., Ковальова Н., Адлер О. Штучний інтелект, як винахідник: стан, проблеми, перспективи розвитку. Expert: paradigm of law and public administration. 2021. Т. 17, № 5. С. 103–110. URL: https://doi.org/10.32689/2617-9660-2021-5(17)-103-110 (дата звернення: 18.10.2023).

23. Artificial intelligence applications for project management / O. V. Polonevych та ін. Connectivity. 2020. Т. 146, № 5. URL: https://doi.org/10.31673/2412-9070.2020.054855 (дата звернення: 18.10.2023).

ДОДАТОК А

Програмна реалізація MAI

```
import traceback
import pandas as pd
from itertools import combinations

def fractional_input(x):
    if '/' not in x:
        return float(x)
    else:
        fraction = x.split('/')
        return int(fraction[0]) / int(fraction[1])

def input_data_items():
    quantity_of_items = int(input(«Кількість об’єктів: «))
    items = list()
    for i in range(0, quantity_of_items):
        items.append(input(f»\033[1mОб’єкт №{i + 1}:\033[0m «))
    return items

def prepare_items_for_weight_calculation(items_to_evaluate):
    unique_pairs = list()
    for comb in combinations(items_to_evaluate, 2):
        unique_pairs.append(comb)

    points = dict()
    for pair in unique_pairs:
        points[pair] = fractional_input(input(f»\033[1mОцініть {pair[0]}
відносно {pair[1]}:\033[0m «))

    data_frame = pd.DataFrame(columns=items_to_evaluate,
index=items_to_evaluate)
    for key in points:
        data_frame.at[key[0], key[1]] = points[key]
        data_frame.at[key[0], key[0]] = 1
        data_frame.at[key[1], key[1]] = 1
        data_frame.at[key[1], key[0]] = 1 / points[key]

    return data_frame

def pairwise_weight_calculation(points_data_frame):
    d = points_data_frame.product(axis=«columns»).pow(1 / 3)
    D = d.sum()
    a = d.div(D).apply(lambda x: round(x, 2))
    return a

def get_ri_value(matrix_size):
    ri_values = [0, 0, 0.58, 0.89, 1.1, 1.2, 1.3, 1.4, 1.4, 1.4]
    return ri_values[matrix_size - 1]

def calculate_cr_value(points_data_frame, w, m, ri):
    lambda_max = points_data_frame.sum().mul(w).sum()
    ci = (lambda_max - m) / (m - 1)
    return round(ci / ri, 3)

def calculate_results(items_weights_list, criteria_weight):
    df = pd.DataFrame(items_weights_list)
    return df.mul(criteria_weight.to_numpy(), axis=«index»).sum().apply(lambda
x: round(x, 2))
```

```
if __name__ == '__main__':
    try:
        print(«\033[1mВведіть критерії:\033[0m»)
        criteria_list = input_data_items()
        data = prepare_items_for_weight_calculation(criteria_list)
        criteria_w = pairwise_weight_calculation(data)
        print(f»\n\033[1mВаги критеріїв (попарний
розрахунок):\033[0m\n{criteria_w}»)

        print(«\n\033[1mВведіть предмети для оцінки:\033[0m»)
        items = input_data_items()
        items_w = list()
        items_cr = dict()

        for criteria in criteria_list:
            print(f»\n\033[1mОцінка предметів за критерієм {criteria}:\033[0m»)
            data = prepare_items_for_weight_calculation(items)
            w = pairwise_weight_calculation(data)
            items_w.append(w)
            size = len(items)
            cr = calculate_cr_value(data, w, size, get_ri_value(size))
            items_cr[criteria] = cr

        result = calculate_results(items_w, criteria_w)
        print(f»\n\033[1mЗначення CR:\033[0m\n{items_cr}»)
        print(f»\n\033[1mРезультат:\033[0m\n{result}»)
    except Exception as e:
        print(traceback.format_exc())
```