

Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту

Рекомендовано до захисту:

протокол засідання кафедри

№ _____ від _____._____.2025 р.

Завідувач кафедри _____

(підпис) Поклонський Є.В.
(прізвище та ініціали)

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему Розробка засобу управління проєктами з рекомендаціями на основі штучного інтелекту

Захищено на засіданні ЕК № _____

протокол № _____ від _____._____.2025 р.

Оцінка _____ / _____

Голова ЕК

(підпис) Фесенко Г.В.
(прізвище та ініціали)

Виконав:

студент 4 курсу, групи КС-42

Спеціальності:

122 Комп'ютерні науки
(шифр і назва спеціальності підготовки)

Махов Максим Володимирович
(прізвище, ім'я та по батькові, підпис) 

Керівник кандидат фіз.-матем. наук,

доцент, Споров О.Є.

(ступень, звання, прізвище та ініціали, підпис)

Рецензент кандидат технічних наук,

доцент, Олешко Олег Іванович

(ступень, звання, прізвище та ініціали, підпис)

Харків – 2025

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра складається з: 89 с., 30 рис., 7 табл., 43 джерел, 2 додатків.

Метою роботи є створення програмного засобу управління проєктами "CircleFlow", що інтегрує модуль штучного інтелекту в Kanban-підхід для надання рекомендацій щодо вибору виконавців і прогнозування термінів виконання завдань для підвищення ефективності командної роботи.

У ході дослідження проаналізовано сучасні підходи до управління проєктами та можливості застосування штучного інтелекту. Обґрунтовано вибір моделей машинного навчання (RandomForestRegressor та RandomForestClassifier). Спроектовано трикомпонентну архітектуру системи (React SPA, Spring Boot Backend, Python/FastAPI ШІ-мікросервіс) та реляційну базу даних (PostgreSQL). Розроблено програмний прототип "CircleFlow" з ключовим функціоналом та механізмами безпеки.

Проведено функціональне тестування та оцінку ефективності ШІ-модуля на синтетичних даних. Результати підтвердили працездатність системи та задовільну якість ШІ-моделей ($R^2=0.8445$ для прогнозування, ROC-AUC=0.949 для рекомендацій). Новизна полягає в підході до інтеграції моделей Random Forest в Kanban-систему для оптимізації планування та розподілу ресурсів. Розроблений прототип може слугувати основою для створення інструменту підвищення ефективності команд.

КЛЮЧОВІ СЛОВА: УПРАВЛІННЯ ПРОЄКТАМИ, ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ, KANBAN, РЕКОМЕНДАЦІЙНА СИСТЕМА, ПРОГНОЗУВАННЯ ЧАСУ, РОЗПОДІЛ ЗАВДАНЬ, SPRING BOOT, REACT, FASTAPI, RANDOMFOREST.

ABSTRACT

Explanatory note for the bachelor's qualification paper consists of: 89 p., 30 fig., 7 tabl., 43 ref., 2 append.

The goal of this work is to create a project management software tool “CircleFlow” that integrates an artificial intelligence module into the Kanban approach to provide recommendations on the selection of performers and predict task completion times to improve teamwork efficiency.

During the research, modern approaches to project management and the possibilities of artificial intelligence application were analyzed. The choice of machine learning models (RandomForestRegressor and RandomForestClassifier) was substantiated. A three-component system architecture (React SPA, Spring Boot Backend, Python/FastAPI AI-microservice) and a relational database (PostgreSQL) were designed. A software prototype of "CircleFlow" with core functionality and security mechanisms was developed.

Functional testing and an evaluation of the AI module's performance on synthetic data were conducted. The results confirmed the system's operability and the satisfactory quality of the AI models ($R^2=0.8445$ for prediction, ROC-AUC=0.949 for recommendations). The novelty lies in the approach of integrating Random Forest models into a Kanban system to optimize planning and resource allocation. The developed prototype can serve as a basis for creating a tool to enhance team efficiency.

KEYWORDS: PROJECT MANAGEMENT, ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, KANBAN, RECOMMENDATION SYSTEM, TIME FORECASTING, TASK ASSIGNMENT, SPRING BOOT, REACT, FASTAPI, RANDOMFOREST.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ.....	6
ВСТУП.....	7
1. АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО УПРАВЛІННЯ ПРОЄКТАМИ ТА МОЖЛИВОСТЕЙ ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ.....	10
1.1 Огляд існуючих методологій та інструментів.....	10
1.1.1 Традиційні методології: Модель Waterfall.....	10
1.1.2 Гнучкі методології (Agile).....	11
1.2 Проблеми та обмеження традиційних систем.....	14
1.2.1 Обмеження традиційної моделі Waterfall.....	15
1.2.2 Виклики при застосуванні гнучких методологій.....	15
1.2.3 Обмеження існуючих програмних інструментів УП.....	16
1.2.4 Вплив людського фактора.....	16
1.3 Аналіз застосування методів ШІ для оптимізації процесів.....	17
1.3.1 Роль ШІ в трансформації УП.....	17
1.3.2 Класифікація та огляд ключових методів ШІ в УП.....	18
1.3.3 Застосування методів ШІ для оптимізації процесів.....	21
1.3.4 Переваги впровадження методів ШІ в УП.....	22
1.3.5 Обмеження та етичні міркування при застосуванні ШІ.....	22
1.4 Постановка задачі дослідження.....	24
2. ПРОЄКТУВАННЯ ЗАСОБУ "CIRCLEFLOW".....	27
2.1 Формулювання функціональних та нефункціональних вимог.....	27
2.2 Проєктування бази даних.....	30
2.3 Визначення основних компонентів та їх взаємодії.....	33
2.3.1 Опис серверного компонента (Backend).....	34
2.3.2 Опис компонента ШІ-рекомендацій (ШІ-мікросервіс).....	37
2.3.3 Опис клієнтського компонента (Frontend).....	38
2.3.4 Взаємодія компонентів засобу “CircleFlow”.....	39
2.4 Проєктування модуля штучного інтелекту.....	41

2.4.1 Вибір моделей машинного навчання.....	41
2.4.2 Опис процесу підготовки даних та формування ознак.....	44
3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ЗАСОБУ "CIRCLEFLOW".....	49
3.1 Реалізація серверної частини.....	49
3.2 Реалізація модуля штучного інтелекту.....	56
3.2.1 Реалізація моделей машинного навчання.....	56
3.2.2 Розробка ШІ-мікросервісу.....	61
3.3 Реалізація клієнтською частини.....	63
3.4 Результати розробки і тестування засобу.....	66
3.4.1 Результати функціонального тестування.....	67
3.4.2 Результати роботи ШІ-модуля.....	73
ВИСНОВКИ.....	80
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ.....	82
ДОДАТОК А.....	87
ДОДАТОК Б.....	89

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

БД – база даних

ІТ – інформаційні технології

МН – Машинне навчання

ПЗ – програмне забезпечення

УП – управління проєктами

ШІ – штучний інтелект

API – Application Programming Interface

CRUD – Create, Read, Update, Delete

CSS – Cascading Style Sheets

CSV – Comma Separated Values

DND – Drag and Drop

DTO – Data Transfer Object

ESLint – ECMAScript Lint

HTTP – Hypertext Transfer Protocol

JPA – Java Persistence API

JSON – JavaScript Object Notation

JWT – JSON Web Token

MAE – Mean Absolute Error

Kanban – візуальний метод управління проєктами та робочим процесом

ORM – Object-Relational Mapping

PK – Primary Key

QA – Quality Assurance

REST – Representational State Transfer

SPA – Single Page Application

SQL – Structured Query Language

UI – User Interface

URL – Uniform Resource Locator

ВСТУП

Світ інформаційних технологій завжди характеризувався високою динамікою та необхідністю швидкої адаптації до змін. В умовах прискорених темпів розробки та посиленої конкуренції ефективне керування проєктами стає критично важливим фактором успіху для ІТ-компаній та стартапів. Традиційні підходи до планування, визначення дедлайнів, розподілу завдань та оцінки завантаженості членів команди часто стикаються з проблемами. Ці проблеми пов'язані з великим обсягом даних, потребою в незалежній оцінці ситуації та швидкому реагуванні на зміни пріоритетів. Тому для прийняття рішень у таких ситуаціях потрібні більш досконалі інструменти, які здатні аналізувати весь обсяг даних та обирати найоптимальніший варіант.

Зараз існує безліч засобів для управління проєктами, за різноманітними методологіями та підходами, зокрема і Kanban. Як одну із важливих властивостей таких підходів можна відзначити візуалізацію робочого процесу, що дозволяє відстежувати стан завдань (від "To do" до "Done"). Але, через те, що більшість із них зосереджені переважно на візуалізації та ручному керуванні, то за межами своєї функціональності залишається автоматизованість. Найефективніший розподіл завдань між виконавцями, оцінка реалістичності термінів їх виконання та ідентифікація потенційних "вузьких місць" часто залишаються на розсуд менеджера проєкту. Це може призводити до суб'єктивних рішень, неефективного використання наданих ресурсів, перевантаження окремих членів команди та, як наслідок, до прострочення дедлайнів або зниження ККД. Адже в умовах зростаючої складності проєктів та команд менеджери часто стикаються з потребою в об'єктивній оцінці навичок багатьох виконавців для виконання конкретних задач. Але зі збільшенням масштабу діяльності менеджерам доводиться спиратися на інтуїцію або на несистематизовані дані. Це підкреслює потребу в засобах, що могли б забезпечити як візуалізацію, так і інтелектуальну підтримку для продуктивного робочого процесу.

Об'єктом роботи є процеси управління проєктами в командній роботі.

Предметом роботи є методи покращення процесів управління проєктом шляхом засобів інтеграції штучного інтелекту.

Метою кваліфікаційної роботи є проєктування та розробка засобу для управління проєктами під назвою “CircleFlow” для підвищення продуктивності робочого процесу за допомогою вбудованого модуля штучного інтелекту.

Для досягнення мети були визначено наступні завдання:

- Розробити архітектуру клієнт-серверної системи управління проєктами з інтегрованим модулем штучного інтелекту.
- Створити функціонал для управління користувачами, проєктами та завданнями, включаючи реалізацію Kanban-дошки з можливістю Drag&drop.
- Розробити модуль штучного інтелекту, що буде здатний прогнозувати час виконання завдань та рекомендувати найбільш підходящих виконавців для кожного завдання.
- Інтегрувати модуль зі штучним інтелектом до основної системи.
- Забезпечити автентифікацію, авторизацію та логування активностей.

Актуальність роботи зумовлена постійно зростаючою потребою у найефективнішому сценарії командної роботи в динамічних умовах ІТ-індустрії. Існуючі на даний момент інструменти часто не надають достатньої автоматизації для підтримки прийняття рішень щодо розподілу ресурсів та прогнозування. Розробка інтелектуальної системи, що поєднує візуалізацію робочого процесу з автоматизованими рекомендаціями на основі штучного інтелекту, є відповіддю на ці виклики, дозволяючи знизити суб'єктивність рішень та підвищити об'єктивність планування.

Новизна кваліфікаційної роботи полягає в інтеграції моделей машинного навчання для надання рекомендацій щодо вибору виконавців та прогнозування часу виконання завдань безпосередньо в систему управління проєктами, що надає користувачеві не лише дані, а й обґрунтовані

інтелектуальні підказки для оптимізації робочих процесів. Запропоноване рішення дозволяє не тільки візуалізувати процес, але й активно впливати на нього, мінімізуючи ризики, пов'язані з людським фактором.

У даній кваліфікаційній роботі представлено систему управління проєктами "CircleFlow", яка реалізує зазначений підхід шляхом поєднання Kanban-методології та засобів штучного інтелекту в програмне рішення з мікросервісною архітектурою, де модуль штучного інтелекту на базі моделей RandomForest відповідає за генерацію інтелектуальних підказок. Завдяки реалізації вищезгаданих методів та ідей пропонується засіб, що покликаний покращити процеси планування та розподілу ресурсів у командній роботі.

1 АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО УПРАВЛІННЯ ПРОЄКТАМИ ТА МОЖЛИВОСТЕЙ ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ

1.1 Огляд існуючих методологій та інструментів

Головне в управлінні проєктами це застосування знань, навичок, інструментів та технік до проєктної діяльності, що будуть задовольняти вимогам певного проєкту. Під ефективним управлінням проєктами можна розуміти планування, організацію та виконання завдань. Методологія управління проєктами (УП) визначається як набір принципів та цінностей, що підкріплені практикою, які і слугують головними настановами для ефективного управління та завершення проєктів. Для конкретних проєктів вибір методології залежить від багатьох факторів, як сама галузь, рівень складності, кількість членів команди і так далі. Саме правильний вибір методології та відповідних інструментів, що їх використовують буде впливати як на ефективність роботи, так і на успішність результату в цілому, що підкреслює важливість даного огляду [1].

1.1.1 Традиційні методології: Модель Waterfall

Водоспадна модель (Waterfall) це класичний підхід, вона підходить туди, де проєкт проходить строго визначені фази - аналіз, проєктування, реалізація, тестування, впровадження, підтримка. І важливий момент тут – кожен наступний етап починається тільки після завершення попереднього. Цей підхід буде ефективним для проєктів зі стабільними, чітко визначеними на початку вимогами. Наприклад, у галузях із суворим регулюванням чи для державних проєктів. Перевагами такої методології є чітка структура, полегшена координація та детальне документування. Однак, головний недолік це низька гнучкість та складність внесення змін на пізніх етапах. Саме це і робить її менш привабливою для динамічних ІТ-проєктів, в яких адаптація до змін є критичною [2].

1.1.2 Гнучкі методології (Agile)

На противагу традиційним підходам, гнучкі методології (Agile) пропонують ітеративний та інкрементальний підхід до розробки. З принципів Agile можна побачити, що це противага обмеженням водоспадної моделі. Agile ґрунтується на документі "Маніфест гнучкої розробки програмного забезпечення", що проголошує чотири ключові цінності:

- Люди та взаємодія важливіші за процеси та інструменти.
- Працюючий продукт важливіший за вичерпну документацію.
- Співпраця із замовником важливіша за обговорення умов контракту.
- Готовність до змін важливіша за дотримання початкового плану.

Також ці цінності доповнюються дванадцятьма принципами, які деталізують вже практичне застосування Agile. Основні це – задоволення потреб замовника через регулярне постачання програмного забезпечення, схвальне ставлення до змін вимог, часті релізи працюючого продукту, щоденна співпраця між розробниками та представниками бізнесу, підтримка вмотивованих працюючих спеціалістів, а також безпосередня увага до технічної досконалості та рівня якості дизайну. Можна побачити, що Agile-підходи охоплюють більшу площу вимог сучасних проєктів. Так як сприяють швидшим циклам зворотного зв'язку, ранньому виявленню проблем, вищій задоволеності стейкхолдерів та значному скороченню часу виходу продукту на ринок [3].

Scrum: Scrum є одним з найпопулярніших фреймворків Agile. Він базується на емпіризмі та заощадливому мисленні. Тут використовується ітеративний підхід, тобто виконується в коротких циклах, що називаються Спринтами (Sprint). Scrum визначає три ключові ролі :

- **Власник Продукту:** Представляє інтереси замовника, відповідає за максимізацію цінності продукту та управління Беклогом Продукту.
- **Скрам-майстер:** Відповідає за ефективність команди, допомагає команді дотримуватися принципів Scrum.

- Команда розробки: спеціалісти, діями яких створюється Інкремент продукту протягом Спринта.

До цієї методології входять 5 подій : Спринт (сюди входять всі потрібні для завершення проєкту завдання), Планування Спринта (зустріч для детального планування спринтів), Щоденний Скрам (щоденні зустрічі для контролю прогресу), Огляд Спринта (огляд виконаної роботи зі стейкхолдерами) та Ретроспектива Спринта (зустріч команди після спринта). Основними артефактами Scrum є Беклог Продукту, Беклог Спринта та Інкремент. Можна зазначити, що Scrum особливо корисний для проєктів з високим ступенем невизначеності, де вимоги можуть змінюватися [4].

Kanban: Це метод, що приділяє особливу увагу візуалізації робочого потоку, обмеженню незавершеної роботи (Work in Progress - WIP), що і є основними принципами методу. Основний атрибут Kanban - це Канбан-дошка, яка містить колонки, що представляють різні етапи робочого процесу (наприклад, To Do, In Progress, QA/Review, Done). При використанні цього метода всі завдання представляються у вигляді карток, що переміщуються по дошці відповідно до їхнього статусу [5].

В контексті проєкту "CircleFlow", завдяки аналізу вище приведених принципів, Kanban виглядає привабливо в порівнянні з іншими методами. Візуалізація за допомогою Канбан-дошки забезпечує прозорість, дозволяючи всім членам команди бачити поточний стан завдань, хто над чим працює, та де можуть виникати затримки. Це сприяє кращій координації та командному розумінню прогресу проєкту. WIP, про який йшла мова вище, є фундаментальним аспектом Kanban, це допомагає запобігти перевантаженню команди. Що призводить до скорочення циклу виконання завдань та підвищення загальної продуктивності. Всі ці механізми дозволяють ефективно виявляти "вузькі місця" (bottlenecks) у робочому процесі. Також Kanban вирізняється своєю гнучкістю та можливістю безперервної доставки цінності. На противагу Scrum з його фіксованими ітераціями, використовуючи Kanban можна вносити зміни та додавати нові завдання в

потік роботи у будь-який час. Без жорстко визначених ролей (як у Scrum) та обов'язкових подій Kanban є простішим у використанні та пристосованні до існуючих процесів команди. Візуалізація, гнучкість та просте виявлення проблемних зон роблять Kanban ефективним та зручним інструментом, особливо при інтеграції з інтелектуальними компонентами, як це планується в CircleFlow [5-6].

Для підтримки реалізації обраних методологій існує велика кількість програмних інструментів. Вони вже допомагають у плануванні, відстеженні прогресу, розподілі завдань, комунікації та управлінні ресурсами. Розглянемо найпопулярніші з них:

- **Jira:** Популярний інструмент для управління проєктами особливо з розробки програмного забезпечення. Підтримує гнучкі методології, такі як Scrum та Kanban, дозволяючи створювати та налаштовувати дошки, керувати беклогом, відстежувати задачі та помилки (баги) [7].
- **Trello:** Простий інструмент, що базується на концепції Канбан-дошки. Можна створювати дошки для проєктів, списки для етапів та картки для завдань, що легко переміщувати між списками. Trello добре підходить для управління завданнями та невеликими проєктами в основному завдяки своїй простоті [7].
- **Asana:** Інструмент для управління проєктами, дозволяє виконувати стандартні операції, як створення, встановлення термінів, призначення пріоритетів та виконавців до кожного завдання. Також пропонується гнучке налаштування, зручна система сповіщень. Asana має більше функцій для більш загального управління проєктами, тоді як, наприклад, Jira краща в контексті саме Agile-розробки [7].

Більшість з вище перелічених інструментів не мають інтегрованих можливостей пов'язаних з прогнозуванням часу та рекомендаціями виконавців. Хоча деякі сучасні платформи починають впроваджувати елементи ІІІ (наприклад, для автоматизації рутинних завдань або базового прогнозування), комплексна інтеграція ІІІ, як це планується в CircleFlow

залишається обмеженою. Традиційні інструменти зазвичай покладаються на ручне введення даних та суб'єктивні оцінки менеджерів. Через такого роду проблеми під час робочого процесу і виникають питання, щодо неефективного розподілу ресурсів та нереалістичного планування. Вище описані труднощі і створюють попит для таких інноваційних засобів, як CircleFlow. Розроблюваний засіб прагне поєднати перевірені методології та практики УП з широкими можливостями штучного інтелекту для оптимізації робочого потоку.

1.2 Проблеми та обмеження традиційних систем

Попередній огляд у підрозділі 1.1 продемонстрував різноманітність існуючих методологій та інструментів управління проєктами. Незважаючи на їх поширеність та функціональність, ці програмні засоби мають низку суттєвих обмежень та створюють певні виклики для команд та менеджерів. Даний пункт присвячений детальному аналізу цих проблем, що обґрунтовує необхідність розробки більш досконалих, інтелектуальних систем управління проєктами, таких як CircleFlow.

1.2.1 Обмеження традиційної моделі Waterfall

Водоспадна модель, незважаючи на свою структурованість, демонструє значні недоліки в динамічному середовищі сучасної ІТ-індустрії:

- **Низький рівень пристосування:** Основна проблема - це низька гнучкість та виникнення ускладнень при внесенні змін після завершення конкретної фази. Як було згадано раніше, в ІТ-галузі вимоги до проєкту рідко залишаються нединамічними. А структура, яка пропонується Waterfall, що вимагає чіткої послідовності, погано пристосована до будь-яких змін. Це може призводити до затримок, невідповідності бюджету та навіть створення продукту, що не є релевантним. А будь-яке застосування змін на пізніх етапах виконання вимагають повернення до попередніх, що у свою чергу є не вигідним як по часу та і по ресурсам.

- **Запізнілий зворотний зв'язок:** Тестування та валідація продукту, все це відбувається аж на завершальних етапах. Це означає, що якщо у продукті і є помилки, то вони будуть виявлені пізно, коли їх виправлення буде складно реалізувати та можливо дорогим. Також факт, що проміжних ітерацій тут немає, обмежує можливість отримати своєчасний зворотний зв'язок.
- **Ризик невідповідності кінцевого продукту реальним потребам:** Всі ці тривалі цикли та обмежена залученість клієнта підвищують ризик невідповідності кінцевого продукту відносно побажань замовника, які могли не один раз змінитися за час розробки [8].

1.2.2 Виклики та обмеження при застосуванні гнучких методологій

Гнучкі ж методології мають більш привабливі властивості, серед них і швидка адаптивність. Але і у цих методологій можуть бути такі недоліки:

- **Проблеми масштабування:** Ефективне впровадження Agile у великих організаціях або для комплексних проєктів може бути складним і часто вимагає додаткових фреймворків.
- **Складність точного довгострокового планування:** Точне довгострокове визначення кінцевих термінів та бюджету для всього проєкту може бути проблемним через властиву гнучкість.
- **Залежність від досвіду та самоорганізації команди:** Також такі підходи часто покладаються на професійний рівень членів команди та їхню мотивацію. Що через людський фактор гарантуватися не може.
- **Специфічні виклики Scrum:** У таких методологіях є характерними виклики, що викликані тиском коротких спринтів. Це може призводити до вигорання членів команди, та впливати на визначення складності завдань на спринт.
- **Специфічні виклики Kanban:** У Kanban же можливі труднощі з термінами без додаткових практик, оскільки сам метод не нав'язує жорстких часових обмежень. Також існує ризик перетворення Kanban-

дошки на "звалище" завдань без належного управління потоком та дотримання WIP-лімітів [9].

1.2.3 Обмеження існуючих програмних інструментів управління проєктами

Сучасні програмні інструменти управління проєктами (Jira, Trello, Asana та інші) надають функціонал для організації роботи, однак мають суттєві обмеження в контексті інтелектуальної підтримки.

- Недостатня інтелектуальна підтримка прийняття рішень: Більшість інструментів є системами для фіксації та візуалізації даних, а не для їх глибокого аналізу та надання нагальних рекомендацій.
- Складність об'єктивної оцінки завантаженості та вибору виконавців: Існуючі інструменти рідко допомагають визначити найкращого виконавця шляхом аналізу навичок, реального навантаження та попередньої роботи.
- Труднощі з точним прогнозуванням термінів виконання завдань: Оцінки тривалості завдань часто базуються на суб'єктивних оцінках і можуть бути неточними. Інструменти також рідко пропонують предиктивні моделі на основі даних про завдання для більш точного прогнозування.
- Інформаційне перевантаження: Велика кількість завдань та даних в інструментах ускладнює сприйняття та виокремлення дійсно важливої інформації для прийняття ефективних рішень менеджером [10-11].

1.2.4 Вплив людського фактора

Навіть за наявності досконалих методологій та інструментів, людський фактор часто суттєво впливає на процеси управління проєктами. Менеджери та члени команди часто покладаються на власний досвід та інтуїцію при прийнятті рішень. Системи же на основі ШІ можуть запропонувати більш об'єктивні, дані-орієнтовані прогнози. Людина, при аналізі великого об'єму даних, може призводити до неточних оцінок та упереджених рішень. Наслідком цього є неефективне використання ресурсів. Також схильність

доручати завдання "перевіреним" виконавцям може призвести до їх перевантаження та вигорання. У той час як інші члени команди можуть бути недостатньо навантажені, що обмежує розвиток та загальну продуктивність команди [12].

Весь зроблений аналіз проблем та обмежень традиційних і гнучких методологій, а також існуючих програмних інструментів управління проєктами, чітко вказує на наявність значного простору для вдосконалення. Низька адаптивність Waterfall, виклики масштабування та прогнозування в Agile, і головне – недостатня інтелектуальна підтримка, суб'єктивність оцінок, труднощі з прогнозуванням та оптимальним розподілом ресурсів. Всі перелічені проблеми в сучасних інструментах УП створюють труднощі на шляху до максимально продуктивної командної роботи. Вплив людського фактора лише посилює ці проблеми. Все це підкреслює гостру необхідність у розробці та впровадженні засобів управління проєктами нового покоління. Якби могли інтегрувати можливості штучного інтелекту для надання об'єктивних, даних-орієнтованих рекомендацій, автоматизації складних аналітичних завдань та управління ризиками. Саме такий підхід закладено в основу розроблюваного засобу "CircleFlow", що буде детальніше розглянуто в наступних розділах.

1.3 Аналіз застосування методів ШІ для оптимізації процесів

1.3.1 Роль ШІ в трансформації управління проєктами

Управління проєктами більшість часу стикається із зростаючою складністю, великими обсягами даних та потребою у швидких рішеннях. ШІ пропонує трансформаційний потенціал для автоматизації та покращення точності прогнозування, оптимізації ресурсів та підвищення ефективності діяльності. Більшість спеціалістів УП вже відзначають вплив ШІ, особливо в ІТ-галузі. На відміну від людини ШІ виявляє приховані закономірності, що сприяє оптимізації процесів та дозволяє менеджерам зосередитись на

стратегічних питаннях. Прогнозується, що до 2030 року до 80% завдань УП виконуватимуться за допомогою ІІІ [13-14].

Незважаючи на усвідомлення потенціалу ІІІ керівництвом компаній, його активне використання менеджерами залишається обмеженим. Це можна пояснити певними “бар'єрами”, як от недостатність знань, або нерелевантність інструментів. Інтеграція ІІІ автоматизує рутину та дозволяє зосередитись на стратегії та управлінні відносинами, тим самим трансформуючи роль проєктного менеджера [15].

1.3.2 Класифікація та огляд ключових методів ІІІ для управління проєктами

Для ефективного застосування ІІІ в УП треба добре розумітися у доступній площині методів. Машинне навчання (МН) є галуззю ІІІ, що надає системам здатність навчатися на основі даних без явного програмування кожного правила. В рамках ML виділяють контрольоване навчання (Supervised Learning), яке використовує мічені дані для вирішення регресійних задач (як-от тривалість чи вартість, за допомогою алгоритмів типу лінійної регресії або Random Forest Regressor) та класифікаційних задач (віднесення об'єктів до класів, наприклад, ризиків чи пріоритетів, з використанням Support Vector Machines (SVM), дерев рішень або ж Random Forest Classifier). Іншим типом є неконтрольоване навчання (Unsupervised Learning), яке виявляє приховані структури в немічених даних, наприклад, через *кластеризацію* для групування схожих об'єктів (стейкхолдерів, проєктів) за допомогою алгоритмів як K-Means. Тут можна виділити ансамблеві методи (Ensemble Learning), де завдяки комбінування декількох моделей покращується точність. До них належать беггінг (Bagging), прикладом якого є Random Forest, що будує множину дерев рішень на випадкових підвбірках даних та ознак; бустинг (Boosting), де моделі будуються послідовно, виправляючи помилки попередніх та стекінг (Stacking), що комбінує прогнози різних базових моделей за допомогою мета-моделі [16].

Глибоке навчання (Deep Learning, DL), також підрозділ машинного навчання, що використовує багатoshарові штучні нейронні мережі для навчання на великих обсягах даних та автоматичного вивчення складних ієрархічних ознак. В управлінні проєктами DL може застосовуватися для прогнозування складних нелінійних залежностей та аналізу неструктурованих даних [13, 16].

Обробка природної мови (Natural Language Processing, NLP) наділяє програми здатністю розуміти та генерувати людську мову. В УП NLP часто застосовується для аналізу документації, звітів, комунікацій, автоматизації звітності та аналізу настроїв [13, 17].

Рекомендаційні системи надають персоналізовані пропозиції так як прогнозують вподобання користувачів. Колаборативна фільтрація базується на аналізі поведінки схожих користувачів (наприклад, з використанням K-Nearest Neighbors, KNN), тоді як контентно-орієнтовані підходи аналізують характеристики самих об'єктів та профіль інтересів конкретного користувача. В УП такі системи можуть рекомендувати оптимальних виконавців для завдань, підбирати релевантні інструменти, а також допомагати у формуванні команд [18].

До інших релевантних методів ШІ належать системи, що накопичують знання фахівців, вони звуться експертними. Та оптимізаційні алгоритми призначені для пошуку найкращих рішень [14].

Зараз частіше використовується гібридизація методів ШІ для вирішення складних завдань, зростання ролі ансамблевих методів (Random Forest) для підвищення надійності прогнозів та перехід від простої автоматизації до інтелектуальної підтримки прийняття рішень [13].

Таблиця 1.3.1 – Ключові методи ІІІ та їх застосування в оптимізації процесів управління проєктами [13, 16].

Метод ІІІ/Категорія	Короткий опис	Приклади процесів УП для оптимізації	Приклади конкретних алгоритмів/моделей
Машинне навчання (ML)	Навчання на даних для виявлення закономірностей.	Прогнозування тривалості, оцінка ризиків, пріоритезація завдань, розподіл ресурсів.	Random Forest, SVM, Decision Trees, Logistic Regression, K-Means.
Глибоке навчання (DL)	Використання багатоварових нейронних мереж для складних завдань.	Аналіз неструктурованих даних, розпізнавання образів у проєктних даних.	Нейронні мережі (CNN, RNN).
Обробка природної мови (NLP)	Розуміння та генерація людської мови.	Аналіз документації, звітів, комунікацій; автоматизація звітності; аналіз настроїв.	Техніки класифікації тексту, генерація тексту.
Рекомендаційні системи	Прогнозування вподобань та надання рекомендацій.	Призначення завдань виконавцям, підбір членів команди.	Колаборативна фільтрація.
Оптимізаційні алгоритми	Пошук найкращих рішень за критеріями.	Розподіл ресурсів, планування графіків, оптимізація портфеля проєктів.	Генетичні алгоритми, лінійне програмування.

1.3.3 Застосування методів ШІ для оптимізації конкретних процесів управління проєктами

Методи ШІ можуть оптимізувати ключові процеси УП на всіх етапах життєвого циклу проєкту, як планування, прогнозування та оцінка. Для прогнозування тривалості завдань використовуються моделі МН (Random Forest, SVM, нейронні мережі), що аналізують попередні дані, такі як складність та кваліфікацію команди. Ключовими ознаками можуть бути "тривалість робочого часу", "ідентифікатор виконавця", "опис завдання". До прикладу успішного застосування входять розробки Accenture та Fingent. Крім того, ШІ сприяє *оптимізації графіків*, аналізуючи залежності та ресурси для створення оптимальних розкладів [19].

В управлінні ресурсами та завданнями ШІ допомагає в автоматизованому розподілі ресурсів та завдань шляхом аналізу навичок, доступності та завантаженості виконавців. Для цього використовуються нейронні мережі, дерева рішень, оптимізаційні алгоритми та рекомендаційні системи. Інструменти, як ClickUp AI, вже тестують такий функціонал. Також МН сприяє балансуванню робочого навантаження та інтелектуальній оцінці завдань, прикладом чого є Taskade [20].

Для запобігання проблем ШІ забезпечує виявлення та оцінку ризиків. Моделі ML (SVM, Random Forest, нейронні мережі) аналізують історичні та поточні дані для ідентифікації потенційних загроз. А NLP аналізує текстову інформацію для виявлення прихованих ризиків [13].

У сфері моніторингу, контролю та комунікації NLP часто використовується для аналізу проєктної документації та комунікацій, вилучаючи ключову інформацію та узагальнюючи тексти. ШІ також автоматизує *звітність та відстеження прогресу*, прикладами чого є Fireflies.ai та ClickUp Brain. Також аналіз настроїв у команді (Sentiment Analysis) за допомогою NLP дозволяє оцінювати моральний дух колективу [20].

Ефективність ІІІ залежить від специфіки даних, а інтеграція ІІІ в інструменти УП стає стандартом (як в Asana, Jira та ін.). ІІІ сприяє цілісному підходу до УП, створюючи зв'язки між різними аспектами (як ризики, ресурси, комунікації) [19].

1.3.4 Переваги впровадження методів ІІІ в управління проєктами

Впровадження ІІІ в УП дає переваги, як підвищення точності прогнозів та оцінок. Це в свою чергу призводить до зростання ефективності та продуктивності команд через автоматизацію рутинних завдань, причому McKinsey оцінює потенціал зростання продуктивності до 40%. Важливою є оптимізація використання ресурсів та скорочення витрат, що є основною мотивацією для 63% компаній. З попереднього аналізу можна сказати, що ІІІ сприяє покращенню якості рішень на основі даних, зменшенню впливу людського фактору, покращеній співпраці та проактивному управлінню ризиками [14-15].

1.3.5 Обмеження та етичні міркування при застосуванні ІІІ

Попри потенціал, впровадження ІІІ в УП має певні труднощі. У таблиці 1.3.2 узагальнено переваги та певні виклики.

Таблиця 1.3.2 – Порівняльний аналіз переваг та викликів застосування ІІІ в управлінні проєктами [21].

Категорія	Переваги	Виклики/Обмеження/Етичні проблеми
Точність та Прогнозування	Підвищення точності прогнозів (терміни, ресурси). Проактивне виявлення ризиків.	Залежність від якості та обсягу даних. Можливість неточних результатів.
Ефективність та Продуктивність	Автоматизація рутинних завдань. Зростання продуктивності команд. Оптимізація використання ресурсів.	Потреба в кваліфікованих фахівцях. Висока вартість впровадження. Опір змінам.

Закінчення таблиці 1.3.2

Категорія	Переваги	Виклики/Обмеження/Етичні проблеми
Прийняття Рішень	Покращення якості рішень на основі даних. Зменшення впливу людського фактору та помилок.	Проблема "чорної скриньки". Складність інтерпретації без ХАІ.
Етичні та Соціальні Аспекти	Потенціал для більш справедливого розподілу завдань.	Упередженість в алгоритмах. Конфіденційність даних. Відповідальність за рішення ІІІ.
Комунікація та Співпраця	Покращений інформаційний потік, автоматичні сповіщення. Аналіз настроїв.	Ризик надмірної залежності від автоматизованих систем.

Бачимо, що ключовими проблемами є залежність від якості та обсягу даних та проблема "чорної скриньки", що підкреслює необхідність Пояснюваного ІІІ (ХАІ) для забезпечення прозорості рішень. Визначення відповідальності за рішення, прийняті штучним інтелектом, захист конфіденційності даних відповідно до законодавства, оцінка впливу автоматизації на зайнятість та упередженість алгоритмів, що виникає внаслідок упередженості історичних даних, — все це етичні міркування щодо використання ІІІ. І також важливими є потреба в кваліфікованих фахівцях та зміні культури, технічні обмеження та вартість впровадження, а також безпека систем ІІІ від кібератак [22].

Етичні проблеми та феномен «чорного ящика» взаємопов'язані, і ХАІ служить засобом для вирішення цих питань. Успішне впровадження ІІІ – це також і організаційний виклик, і він вимагає інвестицій у дані, людський капітал та управління змінами. Існує такий собі потенційний конфлікт між прагненням до автоматизації та необхідністю людського контролю, що вимагає збалансованих гібридних систем "людина-ІІІ" [22].

Отже, після аналізу застосування ШІ для оптимізації УП можна сказати багато про його значний трансформаційний потенціал. Такі технології, як машинне навчання, глибоке навчання, обробка природної мови, а також системи рекомендацій та оптимізації можуть покращити планування, розподіл ресурсів та нагляд.

Їх переваги включають підвищення точності прогнозів, ефективності та покращення якості рішень.

Водночас існує доволі немало викликів, як залежність від даних, проблема "чорної скриньки", етичні дилеми (упередженість, конфіденційність), потреба у фахівцях, вартість та безпека. Після огляду досліджень, згаданих вище, можна зробити висновок, що майбутнє ШІ пов'язане з його повної інтеграції. Проте успішне впровадження буде залежати від всебічної стратегії, мінімізації ризиків та дотримання етичних принципів. У цьому контексті ШІ треба розглядати радше як потенційного "партнера" менеджера, що вимагає постійного дослідження та адаптації у цій сфері.

1.4 Формулювання завдань дослідження на основі проведеного аналізу

При проведенні у попередніх підрозділах аналізу сучасних методологій та інструментів УП було виявлено, що, попри їхню різноманітність та функціональність, більшість існуючих рішень не надають достатньої інтелектуальної підтримки для ефективного вирішення низки ключових проблем. Це в першу чергу стосується неоптимального розподілу завдань та ресурсів, у точному прогнозуванні термінів виконання, недостатньої проактивності у виявленні та управлінні ризиками, а також браку персоналізованої підтримки прийняття рішень. Ці обмеження часто призводять до зниження ефективності роботи команди, відкладення проектів та неефективного використання наявних ресурсів [21].

Та технології ШІ та машинного навчання демонструють значний потенціал для подолання цих викликів. Завдяки ним можна зробити інструменти для автоматизації складних аналітичних завдань, покращення точності прогнозів та надання обґрунтованих рекомендацій. Актуальність даного дослідження зумовлена необхідністю розробки таких інтелектуальних систем УП, які б дозволили підвищити ефективність командної роботи та якість управлінських рішень. Одночасно з цим надзвичайно важливо формувати довіру користувачів до цих систем, підкреслюючи важливість принципів пояснюваної штучної інтелектуальної системи [13, 21].

Метою даної роботи є розробка концепції та програмного прототипу засобу управління проєктами "CircleFlow", яка інтегрує Kanban-методологію з модулем штучного інтелекту для надання інтелектуальних рекомендацій щодо вибору виконавців та прогнозування часу виконання завдань, з метою підвищення ефективності командної роботи та оптимізації процесів управління проєктами.

Для досягнення поставленої мети потрібно буде вирішити такі основні завдання:

1. Розробити концептуальну модель та архітектуру інтелектуального засобу управління проєктами "CircleFlow", що забезпечує ефективну взаємодію між користувацьким інтерфейсом, серверною частиною та інтегрованим модулем штучного інтелекту.
2. Створити та розробити прототип програмного забезпечення для інструменту CircleFlow, що включає ключові функціональні компоненти для управління проєктами.
3. Провести експериментальну оцінку ефективності створеного модуля штучного інтелекту на основі встановлених критеріїв та надати пропозиції щодо практичного використання та майбутнього вдосконалення системи. В результаті постійних зусиль очікуються такі результати:

- Добре обґрунтована концепція інтелектуальної системи управління проектами, відомої як CircleFlow.
- Розроблену архітектуру системи, що включає опис взаємодії її основних компонентів.
- Програмний прототип засобу "CircleFlow" з реалізованим базовим функціоналом УП та модулем ШІ для прогнозування та рекомендацій.
- Результати експериментальної оцінки ефективності роботи ШІ-моделей, що використовуються в системі.
- Рекомендації щодо подальшого розвитку та практичного впровадження розробленої системи.

Дане дослідження та розробка програмного прототипу матимуть певні обмеження. Наприклад, для навчання та тестування моделей машинного навчання будуть використовуватися синтетичні або обмежені набори даних, що може вплинути на узагальнюючу здатність моделей у реальних умовах. Прототип програмного забезпечення буде зосереджений на основних елементах інтелектуальних рекомендацій і може не охоплювати всі потенційні функції комплексної системи управління проектами.

2 ПРОЄКТУВАННЯ ЗАСОБУ УПРАВЛІННЯ ПРОЄКТАМИ "CIRCLEFLOW"

2.1 Формулювання функціональних та нефункціональних вимог

У цьому підрозділі визначимо ключові функціональні та нефункціональні вимоги до системи управління проєктами "CircleFlow". Визначені тут вимоги будуть слугувати основою для подальшого проєктування та розробки системи, забезпечуючи її відповідність поставленим завданням та очікуванням.

Визначення вимог для CircleFlow базується на стандартних практиках інженерії програмного забезпечення та аналізі предметної області, представленому в Розділі 1. Вимоги поділяються на функціональні (FR), що описують, *що саме* система повинна робити, та нефункціональні (NFR), що визначають, *як* система це має виконувати [23].

Функціональні вимоги визначають основні функції та дії, які CircleFlow повинен підтримувати, і вони можуть відрізнятися залежно від ролі користувача (адміністратор, учасник).

CircleFlow повинен пропонувати основні функції управління проєктами, які включають можливість створювати та керувати проєктами і завданнями. Також обов'язково повинна бути візуалізація робочого процесу на Kanban-дошці з можливістю drag-and-drop, відстеження часу та інструменти для командної комунікації та співпраці.

Особливістю CircleFlow є інтелектуальна підтримка користувача на основі ШІ. Тобто він повинен надавати рекомендації щодо оптимальних виконавців завдань та прогнозувати тривалість їх виконання. Ці функції базуватимуться на аналізі даних за допомогою моделей машинного навчання, таких як Random Forest [18-19].

Для забезпечення безпеки та контролю доступу, CircleFlow повинен мати реєстрацію та безпечну автентифікацію користувачів (із застосуванням JWT) та управління профілями (обов'язково навички для ШІ-модуля). Також

повинна бути реалізована рольова модель доступу для розмежування прав та управління членством у проєктах [24].

Система також повинна мати функції для створення звітів та аналітики. Тобто, необхідно налагодити реєстрацію даних. Ці дані будуть використовуватися для формування звітів про статус проєкту, виконання завдань, використання ресурсів. Функціональні вимоги зображені на рисунку 2.1.1 у вигляді Use Case діаграми.

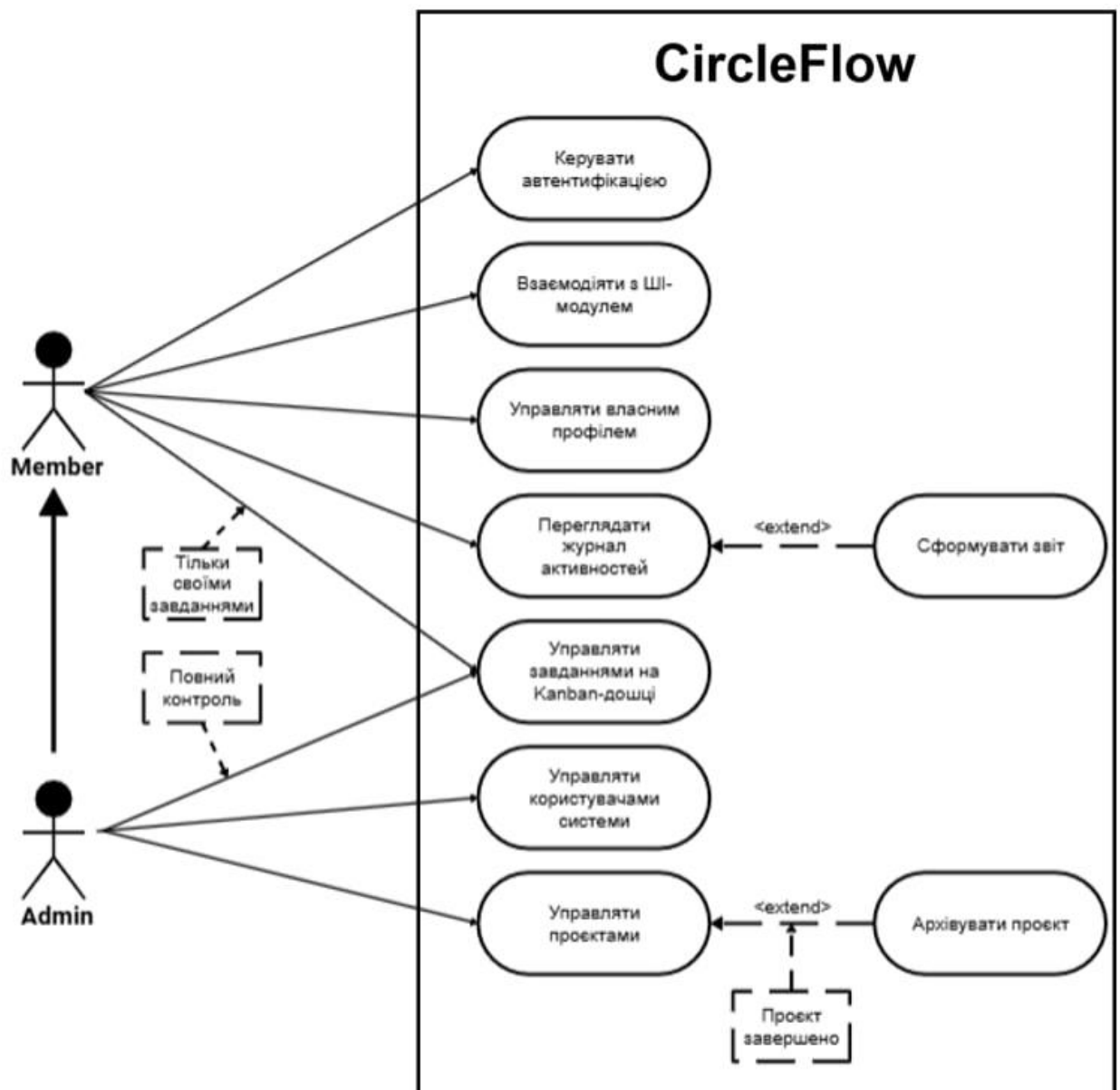


Рисунок 2.1 – Use Case діаграма для засобу "CircleFlow"

Нефункціональні вимоги встановлюють показники якості для CircleFlow відповідно до стандарту ISO/IEC 25010 щодо якості програмних продуктів:

- Функціональна придатність: система повинна повністю і точно виконувати визначені функціональні вимоги, забезпечуючи їх відповідність потребам користувача.
- Продуктивність: додаток повинен швидко реагувати на інтерактивні завдання (протягом 2 секунд) і генерувати рекомендації на основі штучного інтелекту (протягом 5 секунд). Ефективне використання ресурсів сервера та архітектурні заходи для забезпечення масштабованості мають вирішальне значення.
- Сумісність: Веб-інтерфейс має правильно відображатися в сучасних браузерах. Найкращим буде використання RESTful API для інтеграції з іншими засобами.
- Зручність використання: Інтерфейс користувача повинен бути простим та інтуїтивно зрозумілим. Таким, щоб якомога більше мінімізувати помилки користувача.
- Безпека: для інструменту такого типу оптимальним підходом до забезпечення конфіденційності є аутентифікація (JWT) та авторизація. Повинна бути забезпечена захист від значних веб-вразливостей (OWASP Top 10). Паролі користувачів також повинні зберігатися в хешованому форматі з використанням надійних алгоритмів.
- Легкість обслуговування: код та архітектура інструменту повинні бути модульними, добре задокументованими та легко тестованими. Дотримання цих стандартів полегшить постійну підтримку.

Сформульовані вище функціональні та нефункціональні вимоги базуються на аналізі предметної області, сучасних тенденцій застосування ШІ в УП, стандартів якості ПЗ та цілей даної кваліфікаційної роботи. Пріоритезація вимог надасть можливість сфокусуватися на основному функціоналі розроблюваного засобу [25-27].

2.2 Проєктування бази даних

Проєктування БД є важливим етапом розробки всіх інформаційних систем, тому що від структури БД залежить її надійність, швидкодія та масштабованість. Засіб "CircleFlow" призначений для управління проєктами, завданнями, користувачами та пов'язаними з ними даними, і, отже, оперує значними обсягами структурованої інформації.

На основі раніше встановлених функціональних та нефункціональних вимог ми визначили наступні критерії вибору СУБД:

- Модель даних: реляційна модель є найбільш підходящим варіантом, враховуючи структурований характер даних у інструменті управління проєктами.
- Надійність: Для надійності СУБД має підтримувати ACID. Тобто цілісність даних має бути гарантована.
- Масштабованість: для забезпечення надійності СУБД повинна дотримуватися принципів ACID, що гарантують цілісність даних.
- Розширюваність: Також переважною буде СУБД, де є можливість створення кастомних типів, функцій і так далі.

Для вибору СУБД було проведено порівняння декількох широко використовуваних систем, результати якої наведено в таблиці 2.2.1. Ця оцінка дозволить ретельно проаналізувати всі варіанти на основі критичних критеріїв.

Таблиця 2.2.1 – Порівняльний аналіз СУБД [28].

Критерій/Вимога	PostgreSQL	MySQL	MongoDB (NoSQL)
Модель даних	Реляційна, об'єктно-реляційна	Реляційна	Документо-орієнтована
ACID-транзакції	Повна підтримка	Підтримка (з InnoDB)	Обмежена (на рівні документа)
Масштабованість	Висока, горизонтальна та вертикальна	Добра, особливо для читання	Висока, горизонтальна

Закінчення таблиці 2.2.1

Критерій/Вимога	PostgreSQL	MySQL	MongoDB (NoSQL)
Розширюваність	Висока (користувацькі типи, функції, розширення)	Помірна	Гнучка схема даних
Підтримка JSON	Нативна, потужна (json/jsonb)	Нативна JSON	Основний формат

Після аналізу властивостей кожної системи та враховуючи специфіку засобу "CircleFlow", було обрано СУБД PostgreSQL. Саме ця система задовільнила раніше визначені потреби до системи управління базами даних. Також, саме ця система надає гнучку платформу для майбутнього розвитку, наприклад, для інтеграції більш складних аналітичних функцій або механізмів звітності.

Виходячи з функціонального призначення засобу "CircleFlow" як інструменту для управління проєктами, були ідентифіковані наступні ключові сутності:

- Користувачі (Users): Представляють осіб, взаємодіючих з засобом. Поля: UserID, Username, Email, PasswordHash (хеш пароля для безпечного зберігання), Role (наприклад, 'Admin', 'M').
- Проєкти (Projects): Кожен рядок представляє окремий проєкт. Поля: ProjectID, Name, Description, OwnerID (посилання на користувача-творця).
- Завдання (Tasks): Одиниця роботи в рамках проєкту. Поля: TaskID, ProjectID (посилання на проєкт), Title, Description, Priority ('High', 'Low'), Status (наприклад, 'To do', 'In progress'), Deadline, Created_At.

Зв'язки між сутностями:

- Один користувач має можливість створювати кілька проєктів (один до багатьох).

- Один користувач може бути творцем численних завдань.
- Один проект охоплює кілька завдань (один до багатьох).
- Кожне завдання може мати одну особу, призначену виконавцем (один до багатьох).
- Завдання може мати багато Тегів, і один Тег може бути застосований до багатьох Завдань (багато-до-багатьох).

Всі зв'язки багато-до-багатьох були реалізовані через окремі таблиці (наприклад, task_tags, user_skills). А зв'язки між усіма таблицями бази даних зображено на ER-діаграмі (рис. 2.2.1).

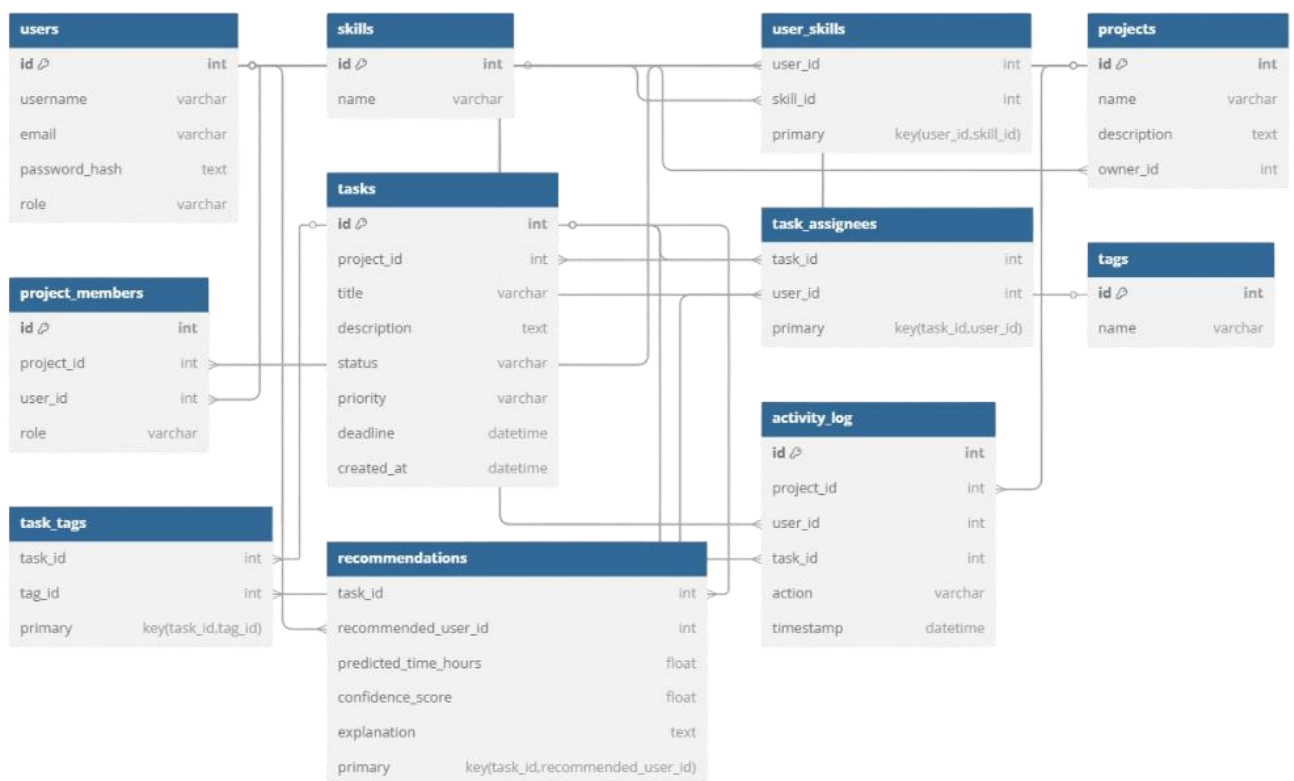


Рисунок 2.2.1 – ER-діаграма бази даних "CircleFlow"

Процес перетворення концептуальної моделі в реляційну схему відповідає певним правилам:

- Кожна сутність, зображена на ER-діаграмі, перетворюється на окрему таблицю в базі даних.
- Поля сутностей стають стовпцями відповідних таблиць.

- Для відносин, класифікованих як “один до багатьох”, до таблиці, що представляє сторону «багато», додаються зовнішні ключі (FK), які пов'язуються з первинним ключем (PK) таблиці, що представляє сторону “один”.
- Для обробки відносин «багато-до-багатьох» створюються проміжні таблиці. Ці таблиці містять зовнішні ключі, які посилаються на первинні ключі обох пов'язаних таблиць, і разом ці два зовнішні ключі зазвичай утворюють складений первинний ключ для проміжної таблиці [29].

Також обов'язково в кожній таблиці визначаються первинні ключі, що ідентифікують кожен унікальний запис таблиці.

Не менш важливим аспектом проєктування є нормалізація даних. Її метою є усунення надлишковості даних. Для засобу "CircleFlow" цільовим рівнем нормалізації буде третя нормальна форма (3НФ). Досягнення 3НФ для таблиць системи "CircleFlow" забезпечить логічно обґрунтовану структуру, також мінімізує надлишковість даних та пов'язані з нею проблеми [30].

Відповідно до принципів, викладених раніше, було створено схему бази даних за допомогою СУБД PostgreSQL. Скрипти DDL (мова визначення даних) для генерації всіх таблиць бази даних можна знайти в Додатку А.

2.3 Визначення основних компонентів та їх взаємодії

Для CircleFlow було обрано клієнт-серверну архітектуру, де інтелектуальний сервіс винесений в окремий мікросервіс. А взаємодія з користувачем відбувається через так званий односторінковий застосунок (Single Page Application – SPA). Цей підхід дозволить доволі гнучко управляти функціональними блоками. До таких блоків відносяться основний бекенд, що відповідає за бізнес-логіку УП та мікросервіс з ШІ-модулем. Те, що ШІ-модуль виокремлений у окремий сервіс надасть змогу незалежно масштабувати цей компонент у разі такої потреби. SPA на фронтенді забезпечить швидкий користувацький досвід. А база даних на PostgreSQL

забезпечить ефективне керування даними [31]. Загальна архітектура засобу “CircleFlow” представлена на рисунку 2.3.1.

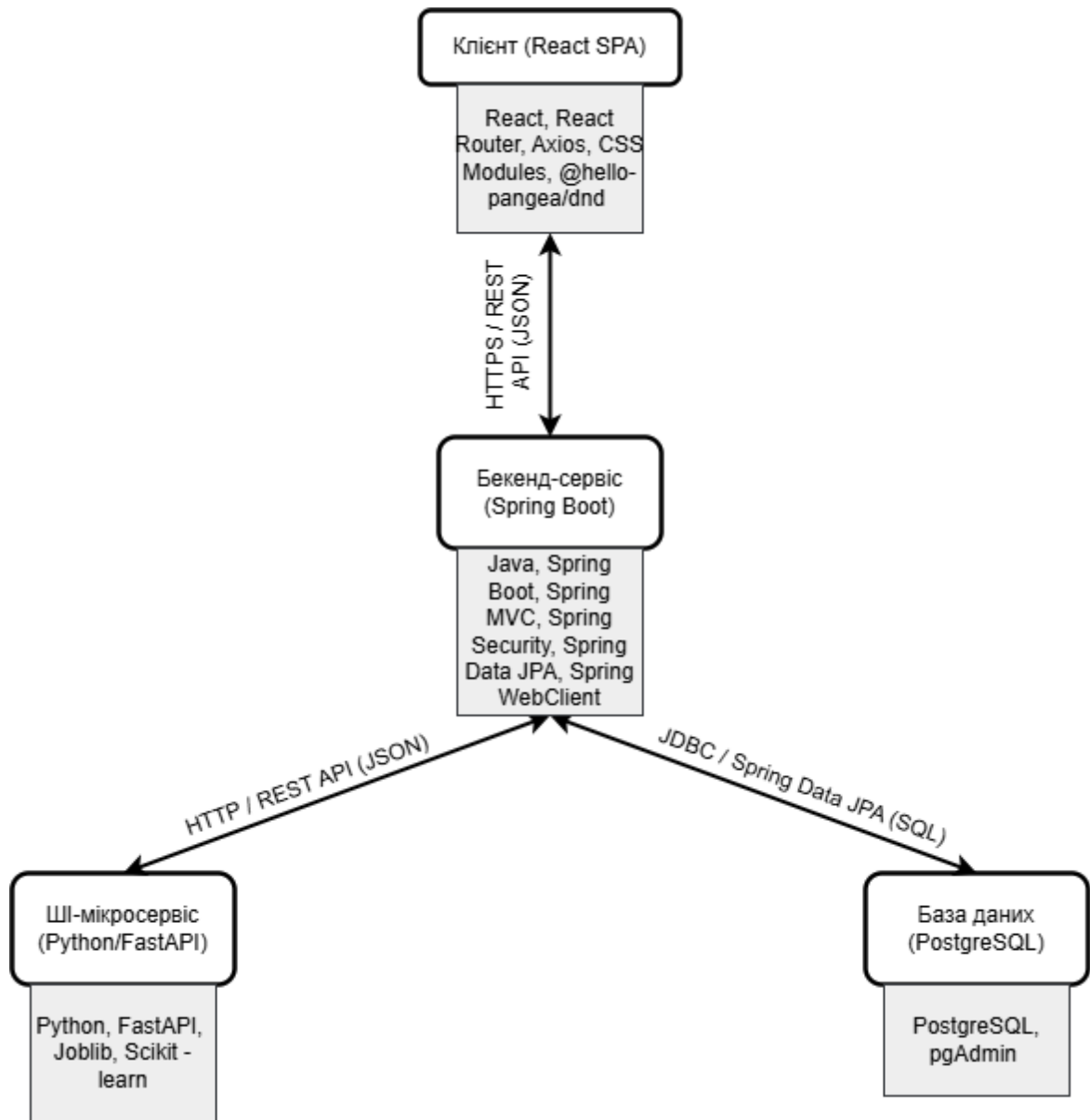


Рисунок 2.3.1 –Архітектурні рівні системи «CircleFlow»

2.3.1 Опис серверного компонента (Backend)

Основний бекенд-сервіс є, так би мовити, ядром системи. Він обробляє бізнес-логіку УП, управління завданнями, користувачами, забезпечення безпеки та, безпосередньо, надання REST API. І реалізований на Java з використанням фреймворку Spring Boot.

Внутрішня архітектура сервісу будується за багат шаровим патерном (Layered Architecture), що включає:

- Рівень представлення (API Layer): Реалізується за допомогою Spring MVC. REST-контролери (анотація у коді - `@RestController`) обробляють HTTP-запити, валідують вхідні дані (використовується DTO з Bean Validation) та надають HTTP-відповіді у JSON. Основні контролери API CircleFlow: `AuthController`, `ProjectController`, `TaskController`, `UserController` [32].
- Сервісний рівень (Service Layer): цей рівень окреслює основну бізнес-логіку, яка реалізується в сервісах (позначених у коді як `@Service`). Ці класи призначені для інкапсуляції логіки маніпулювання даними, оскільки вони взаємодіють з рівнем доступу до даних і контролюють транзакції. Основні сервіси в CircleFlow включають: `UserService`, `ProjectService`, `TaskService`, `RecommendationService`.
- Цей розділ відповідає за комунікацію з базою даних PostgreSQL. Spring Data JPA та Hibernate використовуються тут як ORM для визначення моделей (позначених у коді як `@Entity`) та полегшення операцій CRUD. `FetchType.LAZY` [33] реалізовано для підвищення продуктивності та запобігання перевантаженню сервісу.
- Об'єкти передачі даних (DTO): Ці об'єкти використовуються для передачі тільки необхідної інформації на надіслані запити.
- Проектування API: Надання RESTful API передбачає певні стандарти, наприклад, використання HTTP-методів GET, POST, PUT, DELETE, також JSON для тіл запитів і відповідей і так далі) [34].

Для розуміння того, як виглядають основні ендпоінти API CircleFlow надано таблицю 2.3.1. У таблиці зазначені та описані ендпоінти, за допомогою яких можна маніпулювати даними, що пов'язані з проєктами та завданнями.

Таблиця 2.3.1 – Основні ендпоінти сервісу управління проєктами та завданнями

Операція	HTTP Метод	URL-шаблон	Опис
Створити новий проєкт	POST	/projects	Створює новий проєкт з наданими атрибутами.
Отримати список всіх проєктів	GET	/projects	Повертає список всіх проєктів.
Отримати деталі проєкту за ID	GET	/projects/{projectId}	Повертає інформацію про конкретний проєкт.
Оновити інформацію про проєкт	PUT	/projects/{projectId}	Оновлює атрибути існуючого проєкту.
Видалити проєкт	DELETE	/projects/{projectId}	Видаляє проєкт та пов'язані з ним завдання.
Створити нове завдання в проєкті	POST	/projects/{projectId}/tasks	Створює нове завдання для певного проєкту.
Отримати список завдань проєкту	GET	/projects/{projectId}/tasks	Повертає список завдань для конкретного проєкту.
Отримати деталі завдання за ID	GET	/tasks/{taskId}	Повертає інформацію про конкретне завдання.

Закінчення таблиці 2.3.1

Операція	HTTP Метод	URL-шаблон	Опис
Оновити інформацію про завдання	PUT	/tasks/{taskId}	Оновлює атрибути існуючого завдання (статус, виконавець, термін).
Видалити завдання	DELETE	/tasks/{taskId}	Видаляє завдання.
Призначити виконавця на завдання	POST	/tasks/{taskId}/assignees	Призначає користувача виконавцем завдання.

- **Безпека:** Реалізується за допомогою Spring Security. Автентифікація на з використанням JWT (JSON Web Tokens), які генеруються після успішного входу. Авторизація на основі ролей (позначається у коді як @PreAuthorize). Паролі користувачів зберігаються у базі даних у хешованому вигляді [24].
- **Взаємодія з III-сервісом:** RecommendationService використовує Spring WebClient для асинхронних HTTP-запитів до III-сервісу для отримання прогнозів та рекомендацій.

2.3.2 Опис компонента III-рекомендацій (III-мікросервіс)

III-сервіс відповідає за надання раніше визначених інтелектуальних функцій: прогнозування тривалості завдань та рекомендацій, щодо оптимальних виконавців. Він реалізований як окремий локальний сервіс з використанням фреймворку FastAPI .

FastAPI надає RESTful API з ендпоінтами POST /predict_time/, що приймає характеристики завдання у JSON та повертає прогнозовану тривалість також у JSON. І POST /recommend_user/, який приймає характеристики завдання у JSON, і також повертає ID рекомендованого користувача у JSON. FastAPI

забезпечує автоматичну валідацію даних за допомогою Pydantic та генерацію документації з OpenAPI.

У внутрішній структурі цього сервісу вхідні дані (ознаки завдання) готуються для моделей. Ці моделі також попередньо навчені за допомогою бібліотеки Scikit-learn. Моделі навчаються за методом RandomForestRegressor для прогнозування тривалості завдань. І методом RandomForestClassifier для рекомендації виконавців. Моделі зберігаються за допомогою joblib і завантажуються при запуску сервісу. Запити обробляються, моделі застосовуються, і відповіді генеруються. Крім того, для майбутніх удосконалень проекту буде можливо включити пояснювальний ШІ (XAI), який запропонує більш детальні рекомендації.

2.3.3 Опис клієнтського компонента (Frontend)

Front-end частина - це по суті односторінковий додаток (SPA), створений за допомогою React, який пропонує користувацький інтерфейс для використання інструменту “CircleFlow”. Користувальницький інтерфейс побудований з використанням React-компонентів, сприяючи модульності та багаторазовості. Основні модулі інтерфейсу включають: LoginPage, ProjectsPage, ProjectBoardPage (де реалізована дошка Kanban), TaskModal та RecommendationPanel. Крім того, управління станом обробляється React Context API разом із використанням вбудованих React-хуків, таких як useState, useEffect та useReducer [35].

За допомогою React Router v6 реалізована маршрутизація між сторінками SPA. А з API основного бекенд-сервісу фронтенд взаємодіє за допомогою бібліотеки Axios для HTTP-запитів. Також тут обробляється JSON, JWT надсилається в заголовках Authorization. Були також використані сервісні обгортки (такі як authService.js, projectService.js, тощо). Функціонал Drag&drop для Kanban-дошки тут зроблений з використанням бібліотеки @hello-pangea/dnd, а для стилізації та інкапсуляції стилів використовується CSS Modules.

2.3.4 Взаємодія компонентів засобу “CircleFlow”

Отже, елементи інструменту “CircleFlow” в основному взаємодіють через RESTful API за допомогою протоколу HTTP/HTTPS, з усіма даними, переданими в форматі JSON. Малюнок 2.3.2 ілюструє блок-схему високого рівня, що демонструє взаємодію між основними компонентами розробленої системи.

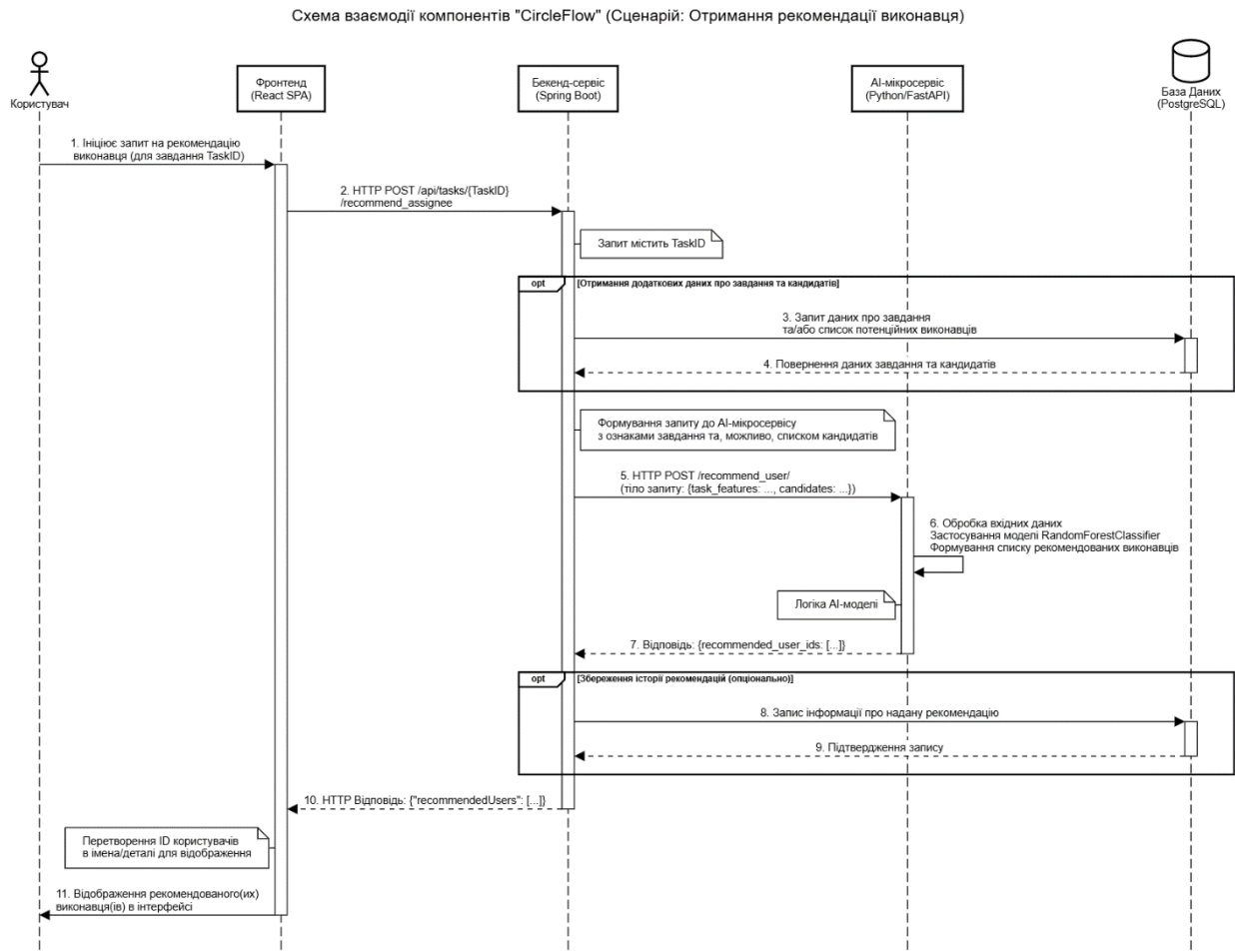


Рисунок 2.3.2 – Схема взаємодії компонентів системи «CircleFlow»

Послідовність основних кроків на схемі, описана нижче:

1. Користувач на Frontend просить рекомендацію для виконавця.
2. Інтерфейс перенаправляє цей запит до основної служби сервера.
3. Серверна служба (RecommendationService) отримує запит, отримує дані з бази даних і відправляє запит HTTP службі AI на кінцевій точці /recommend_user/.

4. III-сервіс обробляє отриманий запит, використовує модель RandomForestClassifier і повертає рекомендованого користувача.
5. Серверна служба отримує відповідь від III-сервісу, також може зберігати рекомендацію в базі даних, і відправляє на фронтенд.
6. Інтерфейс показує користувачеві рекомендованого виконавця.

Діаграма компонентів, що представляє загальну архітектуру програми, відображається на малюнку 2.3.3. Видно, що кожен компонент має окрему роль, і їх взаємодія через стандартизовані API забезпечить гнучкість і потенціал для майбутніх удосконалень інструменту “CircleFlow”.

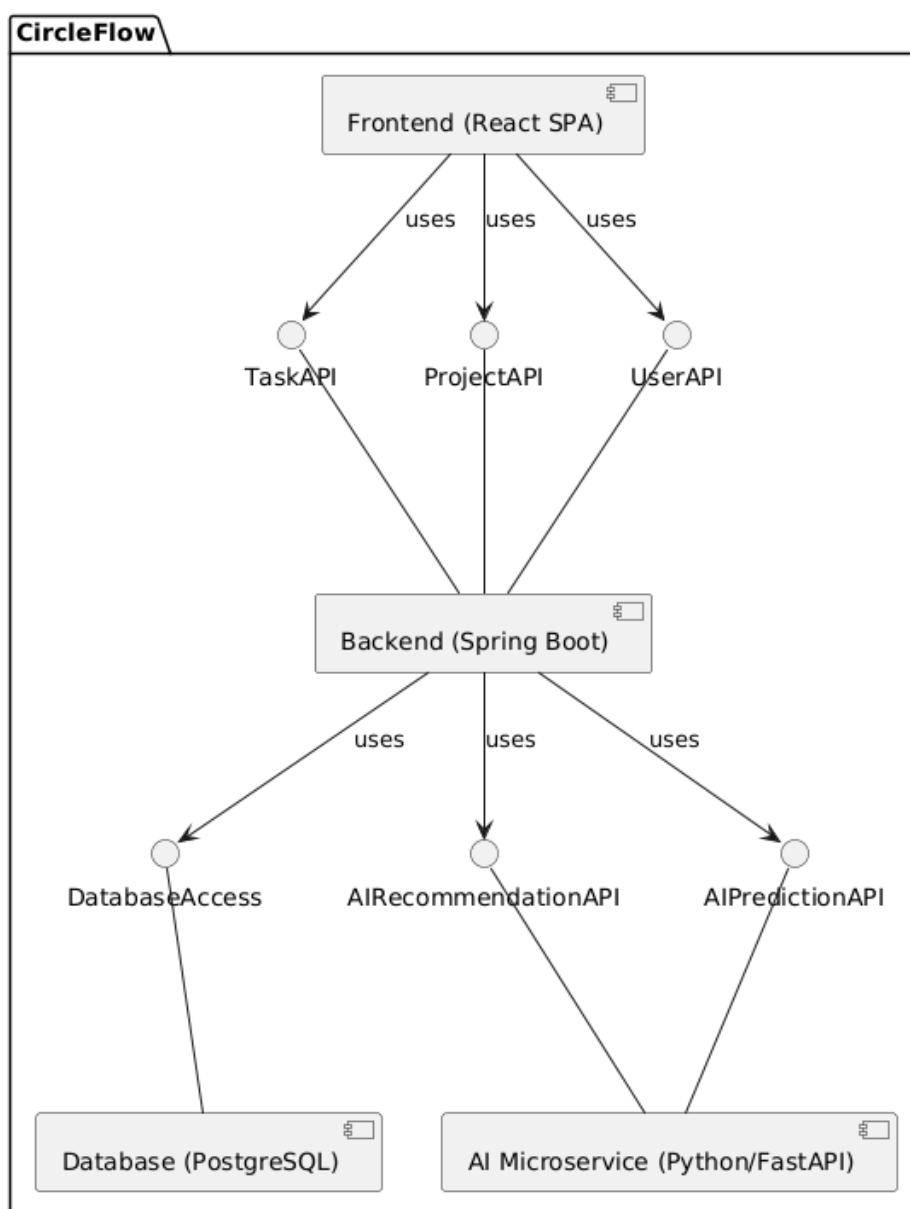


Рисунок 2.3.3 – Component – діаграма CircleFlow

2.4 Проєктування модуля штучного інтелекту

У засобі “CircleFlow” ІІІ-модуль виконує дві ключові функції: прогнозування часу виконання конкретного завдання та рекомендацію найкращих виконавців для його реалізації. Раніше окреслена мета - підвищення ефективності планування та розподілу ресурсів всередині системи. Цей модуль включений в якості незалежного мікросервісу в рамках CircleFlow і розроблений з використанням бібліотек машинного навчання. Бібліотека Scikit-Learn використовується для створення самих моделей, оскільки вона забезпечує різноманітний набір алгоритмів для завдань класифікації та регресії. Використання вже існуючих інструментів scikit-learn дозволить оптимізувати розробку та забезпечити оптимальну роботу модуля ІІІ.

2.4.1 Вибір моделей машинного навчання

Для оцінки тривалості завдань поставлено завдання регресії. Це необхідно, оскільки вхідні функції повинні давати неперервне значення (зокрема, час виконання), класифікуючи його як проблему регресії. Раніше були оцінені різні методи регресії, включаючи лінійну регресію та окремі дерева рішень, але було зроблено висновок, що використання RandomForestRegressor з бібліотеки scikit-learn буде найбільш ефективним. Це рішення було засноване на кількох вагомих причинах: по-перше, Random Forest служить потужним ансамблевим методом. Він перевершує шляхом агрегації прогнозів з декількох дерев рішень і обчислення середнього, що дає більш збалансований вихід. По-друге, він ефективно пом'якшує перенавчання шляхом усереднення результатів з різних дерев і використання природного процесу перевірки завантажувальної карти. Найголовніше, що RandomForestRegressor автоматично оцінює важливість функції, дозволяючи аналізувати, як кожен фактор впливає на остаточне передбачення, тим самим підвищуючи прозорість процесу [36].

Для вхідних ознак моделі регресії використовуються параметри, що описують завдання, такі що показують їхні вимоги та умови виконання.

Наприклад, його можна класифікувати за типом завдання, терміновістю, необхідними навичками, робочим навантаженням тощо. Ці атрибути можуть бути числовими або перетворені в числові значення (закодовані від категорійних до числових).

Згенерований вихід - прогнозована тривалість завдання. Це означає, що модель виробляє єдиний цифровий вихід (в годинах), який згодом відображається в деталях завдання.

Щоб запропонувати відповідних виконавців для завдання, це перетворюється на класифікацію (або ранжирування) проблеми. У своїй найпростішій формі це являє собою багатокласове завдання класифікації, де модель визначає категорію “найкращого виконавця” з пулу всіх співробітників для кожного завдання. Раніше були вивчені різні методи, такі як рекомендації на основі вмісту, машинне навчання та спільна фільтрація. Однак для цього конкретного проекту (враховуючи обмежені дані про виконавців) був обраний класифікаційний підхід, що використовує навчання користувачів, зокрема `RandomForestClassifier`. Основною причиною вибору `RandomForestClassifier` є його володіння табличними даними, різноманітними типами функцій та багатовимірними вихідними змінними. Подібно до свого регресивного аналога, `Random Forest` складається з групи дерев, які визначають найбільш ймовірний клас, який є виконавцем, через процес голосування. Крім того, `Random Forest` спрощує попередню обробку даних, оскільки це не вимагає масштабування ознак. [37].

Вхідні дані для класифікаційної моделі подаються як сукупність характеристик завдання та характеристик виконавця (таких як наявні навички, залучення до інших завдань тощо). Всі ці характеристики дозволять моделі “розуміти”, який виконавець є найбільш підходящим.

Як вихідний результат модель повертає ймовірності для кожного потенційного виконавця. Після навчання модель може видавати рейтинг виконавців за ймовірністю успішного виконання даного завдання. Клас

(виконавець) з найбільшою ймовірністю, тобто вершина рейтингового списку стає рекомендацією.

Вибір моделей Random Forest можна обґрунтувати також і загальними міркуваннями. Можна сказати, що ці алгоритми добре взаємодіють з бібліотекою scikit-learn та мовою Python, саме спрощує їхнє розгортання у вигляді мікросервісу. Крім того, характер даних у проєкті (переважно категоріальні або числові ознаки обмеженого обсягу) добре підходить для методів Random Forest, що відомі своєю стійкістю до невеликих наборів даних, також вони не надто чутливі до їх попередньої обробки. Використання саме scikit-learn обумовлене її популярністю та широким набором реалізованих у ній алгоритмів (регресія, класифікація, кластеризація і так далі), це як раз відповідає вимогам проєкту і дає можливість легко перемикатися між різними моделями.

При цьому, для подальшої оцінки ефективної роботи розроблених моделей планується використовувати стандартні для таких типів задач метрики. У регресійній моделі для прогнозування тривалості використовується середня абсолютна похибка (MAE), яка визначається шляхом усереднення абсолютних відмінностей між передбаченими та фактичними значеннями (Ця метрика відображає середню похибку передбачення в годинах без урахування напрямку похибки). А для класифікаційної моделі рекомендації виконавців використовуються: Точність (Accuracy) - визначається як відношення кількості правильно класифікованих прикладів (як істинно позитивних так і істинно негативних) до загальної кількості прикладів, Повнота (Recall) - для класу "найкращий виконавець" ця метрика показує, яку частку з усіх реально найкращих виконавців модель змогла ідентифікувати як найкращих, та Прецизійність (Precision) - для класу "найкращий виконавець", ця метрика показує, яка частка виконавців, рекомендованих моделлю як найкращі, дійсно є такими, відносно всіх, кого модель позначила як найкращих [38-40].

2.4.2 Підготовка даних та формування ознак

Як джерело даних для навчання моделей використовуються синтетичні дані, тобто штучно згенеровані. Вони імітують реалістичну інформацію про виконані завдання та їхні характеристики. Цей підхід дозволить отримати достатню кількість записів та варіативність ознак, при відсутності потрібних даних з реальних випадків, необхідних для тренування моделей. Набір буде містити дані про тисячі завдань та відповідно інформацію про виконавців. Кожен запис включає ряд ознак, таких як тип завдання, пріоритет, дедлайн, тощо.

Для генерації синтетичних даних використовувалися стандартні бібліотеки Python - NumPy для числових операцій та генерації випадкових даних, і Pandas для структурування даних у табличному вигляді. Процес генерації даних включав такі кроки:

1. Визначення необхідної кількості зразків (5000 записів).
2. Для кожної функції, будь то числовий або категоріальний, був обраний відповідний статистичний розподіл. Наприклад, для категоріальних змінних, таких як `task_type` або `experience_level`, функція `numpy.random.choice` була використана з заздалегідь визначеними ймовірностями для кожної категорії. У випадку числових змінних, таких як `task_complexity`, різні розподіли від NumPy (наприклад, гамма-розподіл з `numpy.random.gamma` або рівномірний розподіл з `numpy.random.uniform`) використовувалися для відтворення реалістичних шаблонів даних, а значення були обмежені за допомогою `numpy.clip` для усунення екстремальних викидів.
3. Для цільових змінних було визначено набір логічних правил, що базуються на згенерованих вхідних ознаках. Цей підхід дозволив нам встановити значущі зв'язки між вхідними даними та результатами. Для посилення реалізму, невелика кількість випадкового шуму також була введена в цільову змінну за допомогою `numpy.random.normal`.

4. Для забезпечення відтворюваності результатів генерації даних на початку процесу встановлювалося фіксоване значення для генератора випадкових чисел (`numpy.random.seed(42)`).
5. Згенеровані таким чином ознаки та цільова змінна об'єднувалися у `Pandas DataFrame`, який надалі використовувався для попередньої обробки та навчання моделей МН.

Для належної підготовки даних для навчання моделей, вони пройдуть такі етапи:

1. Очищення: Це видалення дублікатів, заповнення або вилучення пропущених значень, усунення аномалій. Цей етап необхідний для уникнення некоректних результатів навчання.
2. Кодування категорійних ознак: алгоритми машинного навчання зазвичай вимагають ознак у числовому форматі, тому категорійні стовпці (такі як терміновість питання) перетворюються на числові представлення за допомогою `one-hot-encoding`. Так, наприклад, для однієї з категоріальних ознак створюється декілька бінарних ознак (кожна відповідає конкретній категорії), це і дозволить врахувати їх незалежну інформацію.
3. Обробка текстових характеристик: коли текстові поля присутні (наприклад, опис завдання), вони можуть бути підготовлені для аналізу за допомогою вилучення ключових слів. У даному синтетичному датасеті передбачається, що таким полем буде Опис завдання, де і буде використовуватися метод вилучення ключових слів.
4. Масштабування ознак: Як було згадано вище, `Random Forest` не чутливий до шкали числових ознак, це означає що масштабування (наприклад, стандартне перетворення, нормалізація) тут не є обов'язковим етапом. Але цей етап корисний у разі використання інших алгоритмів чи моделей. Воно дозволить уникнути домінування ознак з великими величинами.

5. Створення функцій: Цей етап передбачає створення набору функцій, які будуть безпосередньо надані моделям. Вибір ознак здійснюється з урахуванням їхньої інформативності для даної задачі [41].

Для регресійної моделі ознаками можуть бути категорія завдання (тобто область вмінь, які потрібні для його виконання), пріоритет та терміновість (різниця поточного часу та часу дедлайну). Також можуть додаватися обчислені ознаки. Обрані ознаки базуються на припущенні, що вони корелюють з реальною тривалістю виконання.

Для моделі класифікації атрибути виконавців (наприклад, їх досвід, область знань, кількість виконаних завдань і середня швидкість виконання завдань) включені в атрибути завдання. Ці особливості допоможуть моделі відрізнити потенційно ефективних виконавців від тих, хто менш актуальний. Вибір особливостей проводиться з урахуванням їх потенційної важливості для завдання. Весь процес вибору та генерації ознак, включно з гіперпараметричним налаштуванням моделей машинного навчання, як правило, ітераційний. Це вказує на те, що пропонується в даний час набір ознак є попереднім і може бути уточнений і розширений за допомогою подальших експериментів і аналізу результатів моделі для досягнення оптимального прогнозування і якості рекомендацій.

Таблиця 2.4.1 ілюструє приклад особливостей, заснованих на характеристиках завдань, які можуть бути використані в регресійних моделях. Додатково ознаки також можуть комбінуватися і перетворюватися виходячи з цілей моделі та аналізу кореляцій між даними.

Таблиця 2.4.1 – Ознаки для моделей регресії

Назва ознаки	Опис ознаки	Тип даних	Приклад значення
TaskType	Категорія або тип завдання	Категоріальна	"Frontend"
Priority	Пріоритет завдання	Категоріальна	"Високий"

Закінчення таблиці 2.4.1

Назва ознаки	Опис ознаки	Тип даних	Приклад значення
RequiredSkills	Назви навичок, необхідних для виконання завдання	Категоріальна	"Design"
TaskDescriptionLength	Довжина текстового опису завдання (у словах)	Числова (ціла)	150
TimeToDeadline	Різниця між поточним часом та часом до дедлайну (у годинах)	Числова (ціла)	5

Також бажано розділити набір даних на навчальні та тестові зразки. Як правило, звичайною практикою є виділення близько 80% даних для навчання та 20% для тестування моделі. Для цього можна використовувати функцію `train_test_split` з бібліотеки `scikit-learn`. Ця функція перемішує дані, залишаючи частину для оцінки якості. Крім того, k -кратна перехресна перевірка використовується для оцінки стійкості моделей. У цьому методі навчальний набір даних ділиться на k розділів, що дозволяє використовувати одну секцію для перевірки в свою чергу, що полегшує більш об'єктивну оцінку можливостей узагальнення моделі. Ця стратегія допомагає мінімізувати випадкові помилки, які можуть виникнути внаслідок невдалого розподілу даних [42].

Для обробки даних і побудови моделей будуть використовуватися доволі популярні у цій сфері інструменти. Як основна мова програмування тут буде Python. Для завантаження та різної обробки датасету будуть

використовуватися бібліотеки `pandas` та `NumPy`, а для попередньої обробки, етапи якої розглядалися вище і для навчання потрібних моделей `RandomForest` - `scikit-learn`. Для виконання потрібних етапів у бібліотеці `scikit-learn` доступні класи `OneHotEncoder`, `StandardScaler`, `RandomForestRegressor` та `RandomForestClassifier` і функція `train_test_split`. Використання такої комбінації технологій і бібліотек дозволяє швидко і чітко інтегрувати модуль ШІ в прикладне середовище за допомогою `FastAPI`.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ЗАСОБУ “CIRCLEFLOW”

3.1 Реалізація серверної частини

Серверний компонент CircleFlow можна розглядати як ядро всієї програми, обробляючи бізнес-логіку, управління даними, безпеку та взаємодію з іншими елементами системи (такими як клієнтська сторона та мікросервіс AI). Бекенд був побудований з використанням фреймворку Spring Boot з Java 17, що забезпечує швидкий розвиток, надійність і масштабованість. Як було визначено раніше, архітектура сервера слідує багат шаровому дизайну, який підтримує чіткий розподіл обов'язків між компонентами, а також спрощує майбутнє обслуговування інструменту.

До основних технологій, що були використані при розробці бекенду додатка відносяться Java 17, Spring Boot 3.4.5 (для створення самостійних вебдодатків на Spring), Spring Security (автентифікація та авторизація з використанням JWT), Spring Data JPA (взаємодія з реляційною БД PostgreSQL), Spring WebClient (HTTP-клієнт для взаємодії з ШІ-мікросервісом), Lombok (зменшення шаблонного коду) та Hibernate Validator (валідація вхідних даних).

Архітектура бекенду дотримується багат шарового підходу, це означає обов'язкову наявність таких компонентів:

- Шар представлення (Controllers): Обробляє HTTP-запити та валідує вхідні дані, потім передає їх на сервісний шар.
- Сервісний шар (Services): Реалізує бізнес-логіку застосунку, також управляє транзакціями і координує взаємодію між репозиторіями.
- Шар доступу до даних (Repositories): Відповідає за взаємодію з БД, використовуючи Spring Data JPA[43].

Будемо розглядати кожен шар та його компоненти послідовно. Почнемо з контролерів, що є входом HTTP-запитів від клієнтської частини (анотація - @RestController). Всі компоненти використовують методи HTTP (GET, POST, PUT, PATCH, DELETE) для виконання операцій CRUD та

інших дій на системних ресурсах. Об'єкти передачі даних (DTO) використовуються для забезпечення безпечного обміну даними. Список основних контролерів можна знайти в таблиці 3.1.1.

Таблиця 3.1.1 – Список реалізованих контролерів

Контролер	Опис	Приклад
AuthController	Відповідає за автентифікацію та реєстрацію користувачів.	POST /auth/login (Автентифікація користувача через LoginRequestDTO та повернення JWT у JwtResponseDTO)
UserController	Управління користувачами системи.	GET /users (Отримання списку всіх користувачів (повертає List<UserDTO>))
ProjectController	Управління проєктами.	POST /projects (Створення нового проєкту на основі ProjectRequestDTO (повертає ProjectDTO))
TaskController	Управління завданнями в межах проєктів.	GET /tasks/{id} (Отримання деталей конкретного завдання (повертає TaskDTO))
RecommendationController	Управління рекомендаціями від III-сервісу.	GET /tasks/{id}/recommendations (Отримання списку рекомендацій (повертає List<RecommendationDTO>))

Інші контролери, такі як ActivityLogController, TagController, SkillController, реалізовані аналогічним чином для управління відповідними сутностями. На рисунку 3.1.1 можна побачити код методу create з контролера ProjectController.

```

@PostMapping
public ResponseEntity<ProjectDTO> create(
    @RequestBody @Valid ProjectRequestDTO dto
) {
    Project created = projectService.createProject(
        ProjectRequestDTO.toEntity(dto), dto.getOwnerId()
    );
    return ResponseEntity
        .status(HttpStatus.CREATED)
        .body(ProjectDTO.fromEntity(created));
}

```

Рисунок 3.1.1 – код методу create з ProjectController

Метод з рисунку 3.1.1 відповідає за обробку POST-запитів для створення проєктів. Спочатку він отримує дані проєкту (ProjectRequestDTO) від органу запиту та виконує попередню перевірку. Далі цей DTO перетворюється в Project entity, а пов'язаний метод (projectService.createProject) викликається для безпосереднього вставлення даних проєкту в базу даних. Після успішного створення метод генерує HTTP-відповідь зі статусом 201 (Created) і повертає DTO новоствореного проєкту клієнту.

Наступним, після контролерів йде сервісний шар, що забезпечує бізнес-логіку додатку (анотація - @Service). Також у сервісах використовуються методи з анотацією @Transactional для управління транзакціями БД, це забезпечує атомарність операцій. Основні реалізовані сервіси показані в таблиці 3.1.2. А на рисунку 3.1.2 зображено код методу AddSkillToUser з сервісу UserService.

Таблиця 3.1.2 – Основні сервіси додатку

Сервіс	Опис	Приклад
UserService	Відповідає за маніпуляції над всіма користувачами	getUserById(Long userId): Знаходить користувача за його ідентифікатором.
ProjectService	Керує проєктами та пов'язаними з ними даними	deleteProject(Long projectId): Видаляє проєкт та пов'язані з ним дані.
TaskService	Управляє завданнями та призначення до них властивостей	createTask(Task task, Long projectId): Створює нове завдання та пов'язує його з відповідним проєктом.
RecommendationService	Запит та отримання рекомендацій	createRecommendation(Recommendation recommendation): Зберігає отримані або створені вручну рекомендації в базі даних.
ActivityLogService	Формування та отримання логу активностей	getActivityLogsByProjectId(Long projectId, Pageable pageable) (отримання логу активностей до проєкту)

```

@Transactional 1 usage Maxim Makhov *
public void addSkillToUser(Long userId, Long skillId) {
    User user = getUserById(userId);
    Skill skill = skillRepository.findById(skillId)
        .orElseThrow(() -> new ResourceNotFoundException("Skill " +
            "not found with id: " + skillId));

    user.getSkills().add(skill);
    userRepository.saveAndFlush(user);
}

```

Рисунок 3.1.2 – код методу AddSkillToUser з UserService

Метод AddSkillToUser з рисунку 3.1.2 призначений для додавання навички до користувача. Спочатку метод отримує сутність користувача за `userId` та сутність навички за `skillId`. Якщо навичка не знайдена, генерується виняток `ResourceNotFoundException`. Після успішного отримання обох сутностей, навичка додається до списку навичок користувача, і оновлена сутність користувача зберігається в БД.

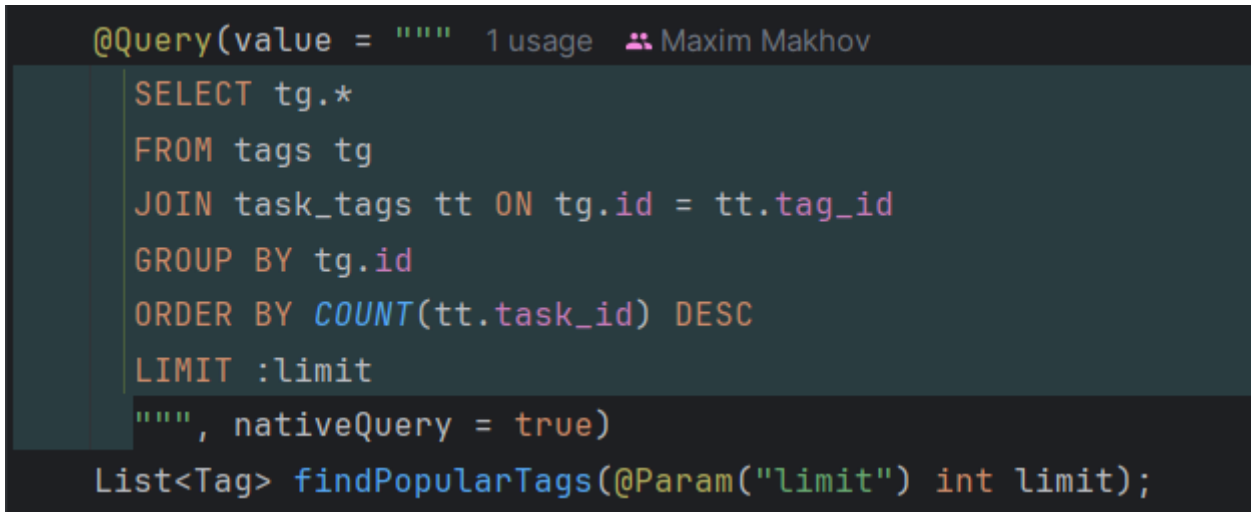
Наступний шар це шар доступу до даних, він реалізований за допомогою Spring Data JPA. Для кожної сутності (анотація - `@Entity`) створено відповідний репозиторій, кожен з яких успадковується від `JpaRepository`. Це успадкування надає набір методів для CRUD-операцій (`save()`, `findById()`, `findAll()`, `deleteById()`) без явної реалізації.

До основних реалізованих репозиторіїв відносяться:

- `ProjectRepository` extends `JpaRepository<Project, Long>`
- `TaskRepository` extends `JpaRepository<Task, Long>`
- `UserRepository` extends `JpaRepository<User, Long>`
- `ActivityLogRepository` extends `JpaRepository<ActivityLog, Long>`
- `RecommendationRepository` extends `JpaRepository<Recommendation, RecommendationId>`

Для складніших запитів, що не можуть бути покритими стандартними методами `JpaRepository`, використовуються кастомні запити за допомогою

анотації `@Query`. Приклад такого методу зображено на рисунку 3.1.3, де зображено код методу `findPopularTags` з репозиторію `TagRepository`.



```
@Query(value = "" 1 usage  Maxim Makhov
SELECT tg.*
FROM tags tg
JOIN task_tags tt ON tg.id = tt.tag_id
GROUP BY tg.id
ORDER BY COUNT(tt.task_id) DESC
LIMIT :limit
"", nativeQuery = true)
List<Tag> findPopularTags(@Param("limit") int limit);
```

Рисунок 3.1.3 – код методу `findPopularTags` з `TagRepository`

Після огляду основних 3-х шарів бекенду додатку можна перейти до не менш важливого компоненту додатку. Це безпека, в `CircleFlow` вона забезпечується за допомогою `Spring Security`, що використовує `JWT`.

Центральний клас конфігурації безпеки це клас `SecurityConfig`. Тут налаштовуються правила доступу до ендпоінтів, вимикається `CSRF`, та реєструються кастомні фільтри.

Також встановлено фільтри автентифікації, такі як `JwtAuthenticationFilter`, що перехоплює кожен запит та перевіряє наявність і валідність `JWT` в заголовку `Authorization`. Коли токен валідний, встановлює автентифікацію користувача в контексті безпеки `Spring`. І другий фільтр це `UsernamePasswordAuthenticationFilter`, що потрібен для обробки запиту на `/auth/login`.

Розмежування доступу у додатку виконано на двох рівнях. По-перше, на рівні `HTTP`-запитів у `SecurityConfig`, приклад цього розмежування зображено на рисунку 3.1.4. І також на рівні методів контролерів, що робиться анотацією `@PreAuthorize`.

```

@Bean
public void partAccess (HttpSecurity http) throws Exception {
    http.authorizeHttpRequests(authorize -> authorize
        .requestMatchers("/auth/**").permitAll()
        .requestMatchers("/projects/**", "/tasks/**").hasAnyRole(
            ...roles: "MEMBER", "ADMIN")
        .anyRequest().authenticated()
    );
}

```

Рисунок 3.1.4 – розмежування доступу на рівні HTTP-запитів

За генерацію JWT після успішної автентифікації та валідацію токенів відповідає сервіс JwtProvider. А за хешування паролів – BcryptPasswordEncoder, після використання якого паролі зберігаються в БД.

І остання частина серверної частини додатку це частина, що взаємодіє з ШІ-мікросервісом, що розроблений на FastAPI (Python). Всі необхідні взаємодії виконуються в рамках RecommendationService за допомогою Spring WebClient. Приклад методу createRecommendation від RecommendationService (рис. 3.1.5) приймає рекомендацію разом з ідентифікаторами завдань і користувачів і перевіряє їх існування в базі даних. Якщо вони вже присутні в базі даних, створюється нова рекомендація, що включає наданий текст рекомендації та ідентифікатори завдання та користувача. Потім знову нова створена рекомендація зберігається в базі даних програми.

```

@Transactional 1 usage Maxim Makhov
public Recommendation createRecommendation(Recommendation recommendation, Long taskId, Long userId) {
    Task task = taskRepository.findById(taskId)
        .orElseThrow(() -> new ResourceNotFoundException("Task not found with id: " + taskId));

    User user = userRepository.findById(userId)
        .orElseThrow(() -> new ResourceNotFoundException("User not found with id: " + userId));

    RecommendationId id = new RecommendationId(taskId, userId);
    recommendation.setId(id);
    recommendation.setTask(task);
    recommendation.setRecommendedUser(user);

    return recommendationRepository.save(recommendation);
}

```

Рисунок 3.1.5 – код методу createRecommendation з RecommendationService

Описана вище реалізація серверної частини CircleFlow забезпечить надійну основу для ефективної роботи всього додатку. Обробляючи дані, керуючи бізнес-процесами та взаємодіючи з іншими компонентами, як клієнтський інтерфейс та ШІ-сервіс серверна частина виконує всі раніше спроектовані функції.

3.2 Реалізація модуля штучного інтелекту

Модуль штучного інтелекту є компонентом засобу "CircleFlow", що відповідає за надання рекомендацій та прогнозів. Його реалізація включає розробку моделей МН та створення окремого мікросервісу для їх обслуговування.

3.2.1 Реалізація моделей машинного навчання

Отже, розглянемо процес розробки та навчання моделей МН для прогнозування тривалості завдань та рекомендації виконавців. Для розробки було використано бібліотеки Python, зокрема Scikit-learn для реалізації алгоритмів та Pandas для маніпуляції даними.

Почнемо з моделі рекомендації виконавців, що призначена для визначення найбільш підходящого співробітника для виконання конкретного завдання на основі аналізу низки факторів.

Як було визначено раніше, для навчання моделі використовувалися синтетичні дані, згенеровані імітуючи реальні характеристики завдань та виконавців. Цільовою змінною для навчання слугувала ознака `is_best_match`, що вказує на оптимальність вибору конкретного виконавця для певного завдання (1 – оптимальний, 0 – ні). Ключові вхідні ознаки включали:

- `task_complexity`: Числова оцінка складності завдання.
- `task_type`: Категоріальна ознака типу завдання (наприклад, 'frontend', 'backend', 'fullstack').
- `task_priority`: Категоріальна ознака пріоритету завдання ('Low', 'Medium', 'High').
- `time_left`: Час, що залишився до дедлайну (в годинах).
- `member_load`: Поточне навантаження потенційного виконавця (кількість активних завдань).
- `avg_task_duration`: Середній час виконання завдань даним виконавцем.
- `skill_match`: Числова оцінка відповідності навичок виконавця вимогам завдання (від 0 до 1).
- `experience_level`: Категоріальна ознака рівня досвіду виконавця ('Junior', 'Mid', 'Senior').

Категоріальні ознаки були перетворені у числовий формат за допомогою `LabelEncoder` з бібліотеки `Scikit-learn`. Для підвищення прогностичної здатності моделі були створені також похідні ознаки, такі як:

- `skill_time_interaction`: Взаємодія між відповідністю навичок та доступним часом ($\text{skill_match} * \log_{1p}(\text{time_left})$).
- `complexity_load_ratio`: Співвідношення складності завдання до навантаження виконавця ($\text{task_complexity} / (\text{member_load} + 1)$).
- `efficiency_score`: Оцінка ефективності виконавця на основі співвідношення його навичок та середнього часу виконання завдань ($\text{skill_match} / (\text{avg_task_duration} + 1)$).

Процес навчання моделі RandomForestClassifier включав ініціалізацію з підібраними гіперпараметрами та навчання на підготовленій навчальній вибірці, як це показано на рисунку 3.2.1.

```
# Ініціалізація та навчання RandomForestClassifier
task_assign_model = RandomForestClassifier(
    n_estimators=100,
    max_depth=10,
    min_samples_split=5,
    min_samples_leaf=2,
    class_weight='balanced',
    random_state=42
)
task_assign_model.fit(X_train_clf, y_train_clf)

# Збереження моделі
import joblib
model_data_to_save = {
    'model': task_assign_model,
    'encoders': label_encoders_dict,
    'feature_names': list(X_train_clf.columns)
}
joblib.dump(model_data_to_save, 'task_assignment_model.pkl')
```

Рисунок 3.2.1 – код для ініціалізації та збереження моделі

Згенерований датасет було розділено на навчальну (80%) та тестову (20%) вибірки. Для задачі класифікації було обрано модель RandomForestClassifier з бібліотеки Scikit-learn. Модель ініціалізувалася з такими гіперпараметрами, що були підібрані на основі попередніх експериментів:

- `n_estimators=100` (кількість дерев у лісі)
- `max_depth=10` (максимальна глибина кожного дерева)
- `min_samples_split=5` (мінімальна кількість зразків, необхідна для розділення внутрішнього вузла)

- `min_samples_leaf=2` (мінімальна кількість зразків, необхідна для знаходження у листовому вузлі)
- `class_weight='balanced'` (для автоматичного коригування ваг класів з огляду на можливий дисбаланс)
- `random_state=42` (для забезпечення відтворюваності результатів)

Для оцінки стабільності та уникнення перенавчання використовувалася 5-фолдова стратифікована крос-валідація (`StratifiedKFold`) на навчальній вибірці. В якості основної метрики якості під час крос-валідації використовувалася ROC-AUC. І після підтвердження стабільності обраних гіперпараметрів, фінальна модель навчалася на повному навчальному наборі даних.

Вже навчена модель `RandomForestClassifier`, разом з усіма використаними об'єктами `LabelEncoder` та списком назв ознак, була серіалізована та збережена у файл з розширенням `.pkl` за допомогою бібліотеки `joblib`. Це дозволить легко завантажувати та використовувати готову модель в ІІІ-мікросервісі без необхідності повторного навчання.

Тепер перейдемо до огляду реалізованої моделі для прогнозування тривалості завдань. Для навчання цієї моделі також використовувався синтетичний датасет. Цільовою змінною тут була `duration_hours` (тривалість завдання в годинах). В якості вхідних ознак використовувалися характеристики завдання, такі як:

- `task_type`: Категоріальна ознака типу завдання.
- `task_priority`: Категоріальна ознака пріоритету завдання.
- Та інші числові параметри, що можуть впливати на тривалість (оцінка складності, кількість залежностей, середній час виконання завдань у призначеного виконавця тощо).

Категоріальні ознаки теж кодувалися за допомогою `OneHotEncoder`. Для ручної обробки даних та застосування перетворень послідовно, був використаний `ColumnTransformer`. Він дозволяє застосувати `OneHotEncoder` лише до вказаних категоріальних стовпців, залишаючи інші стовпці без змін.

Ці дані також були розділені на навчальну (80%) та тестову (20%) вибірки з використанням `random_state=42` для відтворюваності. Було також використано `Pipeline` з `Scikit-learn` для реалізації моделі регресії, код використання якого зображено на рисунку 3.2.2. Цей конвеєр об'єднував два кроки:

1. Попередня обробка даних за допомогою визначеного `ColumnTransformer` (кодування категоріальних ознак).
2. Навчання моделі `RandomForestRegressor`.

```
# Створення та навчання Pipeline для RandomForestRegressor
reg_pipeline = Pipeline([
    ('prep', preprocessor),
    ('reg', RandomForestRegressor(n_estimators=100, random_state=42))
])
reg_pipeline.fit(X_train_reg, y_train_reg)

# Збереження моделі
import joblib
joblib.dump(reg_pipeline, 'task_duration_model.pkl')
```

Рисунок 3.2.2 – Навчання та збереження моделі

Модель `RandomForestRegressor` ініціалізувалася з такими параметрами:

- `n_estimators=100` (кількість дерев)
- `random_state=42` (для відтворюваності) Навчання моделі регресії проводилося на навчальній вибірці.

Якість навченої моделі регресії оцінювалася на тестовій вибірці за допомогою двох стандартних метрик для задач регресії:

- Середньоквадратична помилка (MSE): Отримана оцінка MSE склала приблизно 23.47. Це значення вказує на середній квадрат різниці між прогнозованими та фактичними значеннями тривалості.

- Коефіцієнт детермінації (R^2): Отримане значення R^2 склало приблизно 0.843. Коефіцієнт детермінації, близький до 1, означає що модель добре пояснює варіативність цільової змінної (тривалості завдань).

Для використання в III-мікросервісі, навчений Pipeline, що включає як кроки попередньої обробки, так і саму навчену модель RandomForestRegressor, також зберігається у файл з розштренням .pkl за допомогою joblib.

3.2.2 Розробка III-мікросервісу

Для доступу до функціоналу навчених моделей МН було розроблено окремий III-мікросервіс. Тут реалізується логіка прогнозування та рекомендацій, і надається через REST API.

III-мікросервіс реалізовано на Python з використанням фреймворку FastAPI, що забезпечує високу продуктивність та автоматичну валідацію даних через Pydantic. Для запуску сервісу використовується ASGI-сервер Uvicorn. Робота з моделями здійснюється через Scikit-learn та Joblib.

Основна структура проєкту включає:

- main.py: Міститься логіка FastAPI-серверу, визначення ендпоінтів та безпосереднє завантаження моделей.
- models/: Директорія для збережених файлів моделей (task_assignment_model.pkl, time_prediction_pipeline.pkl).
- schemas.py: Визначення Pydantic-моделей для валідації даних запитів та відповідей.

Також мікросервіс надає два основні ендпоінти:

1. Для прогнозування часу (POST /predict_time/). Тут приймаються характеристики завдання, передаються завантаженій моделі time_prediction_pipeline і повертає прогнозований час виконання. На рисунку 3.2.3 зображено код такого ендпоінту.

```
@app.post("/predict_time/", response_model=TimePredictionOutput)
async def predict_task_time(task_data: TaskFeaturesTimeInput):
    feature_dict = task_data.dict(exclude={'task_id'})
    input_df = pd.DataFrame([feature_dict])

    prediction = time_prediction_pipeline.predict(input_df)
    return TimePredictionOutput(task_id=task_data.task_id,
                               predicted_time_hours=float(prediction[0]))
```

Рисунок 3.2.3 – код ендпоінту, що повертає час виконання завдання

- Для рекомендації виконавця (POST /recommend_user/). Тут приймаються характеристики завдання та виконавців, використовується завантажена модель `task_assignment_model` для підбору оптимального виконавця та повертає його ID. На рисунку 3.2.4 зображено код такого ендпоінту

```
@app.post("/recommend_user/", response_model=UserRecommendationOutput)
async def recommend_task_user(data: TaskRecommendationInput):
    model = task_assignment_model_data['model']
    encoders = task_assignment_model_data['encoders']
    feature_names = task_assignment_model_data['feature_names']

    best_candidate_id = None
    highest_score = -1.0

    for candidate in data.potential_candidates:
        current_features_df = prepare_features_for_model(
            data.task_features, candidate.features, encoders, feature_names)
        score = model.predict_proba(current_features_df)[0][1]

        if score > highest_score:
            highest_score = score
            best_candidate_id = candidate.user_id
    if best_candidate_id is None:
        raise HTTPException(status_code=404, detail="Suitable candidate not found")
    return UserRecommendationOutput(task_id=data.task_id,
                                    recommended_user_id=best_candidate_id)
```

Рисунок 3.2.4 – код ендпоінту що повертає рекомендованого виконавця

Навчені моделі завантажуються один раз при старті FastAPI-застосунку за допомогою обробника події "startup", що забезпечує їх готовність до використання без затримок. Pydantic забезпечує валідацію вхідних даних. А у разі невідповідності FastAPI автоматично повертає помилку 422 Unprocessable Entity. Пайплайн роботи ШІ-мікросервісу наведено на рисунку 3.2.5.



Рисунок 3.2.5 – Пайплайн обробки даних та використання моделей ШІ

3.3 Реалізація клієнтською частини

Клієнтська частина системи "CircleFlow" реалізована як односторінковий застосунок (SPA – Single Page Application), що забезпечує користувачам інтерактивний та швидкий інтерфейс для управління проєктами та завданнями.

Фронтенд розроблено з використанням технологій, що дозволило створити динамічний користувацький інтерфейс. Основний технологічний

стек включає: React 18, React Router v6 (для реалізації маршрутизації), Axios (для взаємодії з REST API бекенд-сервісу), @hello-pangea/dnd (бібліотека для реалізації функціоналу перетягування), CSS Modules (для локалізації CSS-стилів).

Інтерфейс користувача побудований на основі компонентного підходу React, що дозволяє створювати незалежні та перевикористовувані частини UI. До основних сторінок інтерфейсу входять:

- LoginPage.jsx: Сторінка автентифікації користувача. Містить форму для введення email та пароля. При успішному вході викликає authService.login(), зберігає отриманий JWT-токен.
- ProjectsPage.jsx: Відображає список проєктів, доступних користувачеві. Та дає можливість створити новий проєкт.
- ProjectBoardPage.jsx: Реалізує Kanban-дошку для конкретного проєкту.
- TaskModal.jsx: Модальне вікно для створення або редагування завдання.

Ключові UI-компоненти:

- TaskCard.jsx: Компонент, що відображає картку завдання на Kanban-дошці.
- Column.jsx: Компонент, що представляє колонку на Kanban-дошці (наприклад, "To Do").
- RecommendationPanel.jsx: Компонент, що відображається у модальному вікні завдання або на картці завдання. Після натискання кнопки "Generate Recommendations" викликає відповідні методи через бекенд та відображає отриманий прогнозований час та рекомендованого виконавця.

Управління станом в "CircleFlow" реалізовано переважно за допомогою вбудованих механізмів React. Для глобальних даних, таких як інформація про автентифікованого користувача та, можливо, глобальні налаштування теми використовується React Context API.

Для управління локальним станом компонентів використовуються хуки `useState` та `useEffect` (для виконання побічних ефектів, як запити до API).

Маршрутизація в застосунку реалізована за допомогою `React Router v6`. Основні маршрути визначаються в компоненті `AppRoutes.jsx`, який використовує компоненти `<Routes>` та `<Route>` з `react-router-dom`. Приклад маршрутизації фронтенд частини показано на рисунку 3.3.1.

```
import React from 'react';
import { Routes, Route, Navigate } from 'react-router-dom';
import LoginPage from '../pages/LoginPage';
import ProjectsPage from '../pages/ProjectsPage';
import ProjectBoardPage from '../pages/ProjectBoardPage';
import ProtectedRoute from './ProtectedRoute'; // Компонент для захисту маршрутів

function AppRoutes() {
  return (
    <Routes>
      <Route path="/login" element={<LoginPage />} />
      <Route path="/projects" element={<ProtectedRoute><ProjectsPage /></ProtectedRoute>} />
      <Route path="/projects/:projectId/board" element={<ProtectedRoute><ProjectBoardPage /></ProtectedRoute>} />
      <Route path="*" element={<Navigate to="/projects" />} />
    </Routes>
  );
}
```

Рисунок 3.3.1 – маршрутизація фронтенд частини

Всі HTTP-запити до REST API бекенду здійснюються за допомогою бібліотеки `Axios`. Створюється екземпляр `Axios` з базовим URL API та налаштовуються перехоплювачі для автоматичного додавання JWT-токену до заголовка `Authorization` (рис. 3.3.2).

```
import axios from 'axios';

const axiosInstance = axios.create({
  baseURL: 'http://localhost:8080/circleflow',
});

axiosInstance.interceptors.request.use((config) => {
  const token = localStorage.getItem('jwtToken');
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});

export default axiosInstance;
```

Рисунок 3.3.2 – з'єднання з серверної частиною додатка

Для реалізації Kanban дошки, а саме функціоналу Drag&Drop використовувалася бібліотека @hello-pangea/dnd. Це можна зробити наступними кроками:

1. Компонент ProjectBoardPage обгортається в <DragDropContext>, що надає контекст для перетягування.
2. Кожна колонка статусу (Backlog, To Do тощо) є компонентом <Drroppable>, а кожна картка завдання – <Draggable>.
3. Функція handleDragEnd, що передається в DragDropContext, викликається після завершення перетягування. Ця функція оновлює локальний стан дошки та відправляє запит на бекенд через taskService.updateTaskStatus() для збереження змін.

Описана вище реалізація клієнтської частини CircleFlow забезпечує зручний та функціональний інтерфейс для користувачів. Також ефективно взаємодіє з бекендом для управління даними та відображення рекомендацій.

3.4 Результати розробки і тестування засобу

Після завершення розробки основних компонентів засобу "CircleFlow" було проведено тестування всіх його компонентів для перевірки

На рисунку зображено запит методом POST на URL `http://localhost:8080/circleflow/auth/login`. У тілі запиту (у форматі JSON) вказано коректні `username` та `password`. У відповідь отримується статус 200 OK. А в тілі відповіді міститься сам JWT-токен, що означає коректну роботу серверу та контролеру авторизації.

2. Створення нового проєкту: Перевірка ендпоінту `/circleflow/projects` на можливість створення нового проєкту автентифікованим користувачем (рис. 3.4.2).

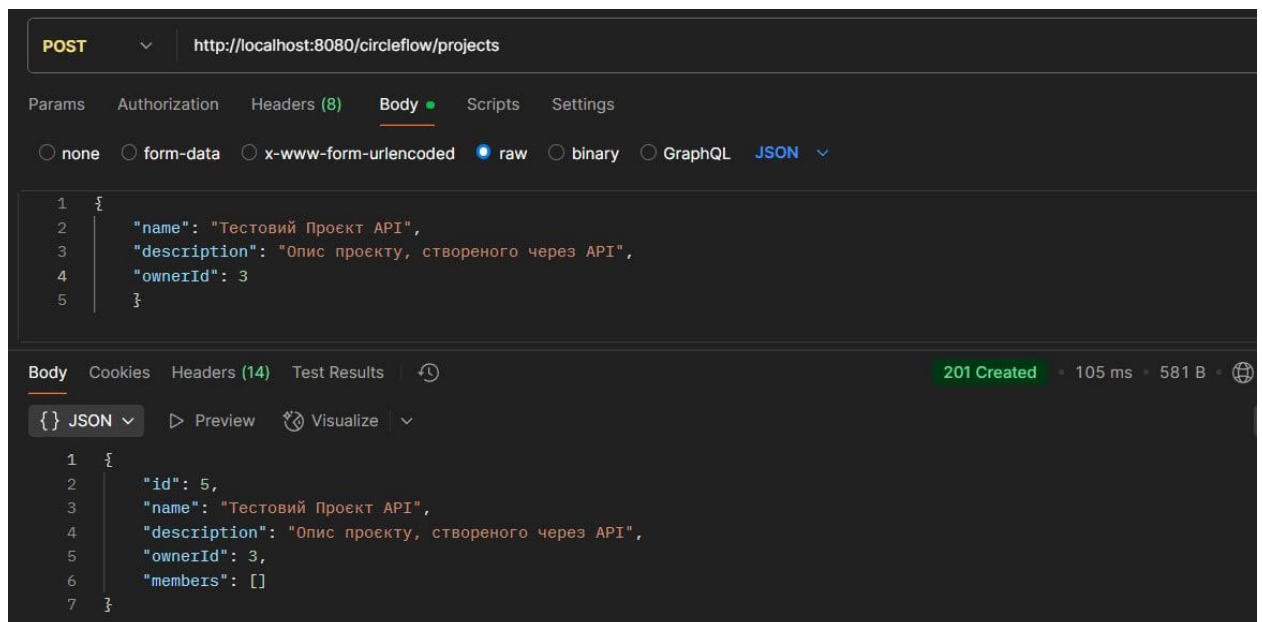


Рисунок 3.4.2 – тестування створення нового проєкту

На рисунку 3.4.2 зображено запит методом POST на URL `http://localhost:8080/circleflow/projects`. У заголовку `Authorization` передано `Bearer <JWT_TOKEN>`, а в тілі запиту вказано дані для нового проєкту. У відповідь отримано статус 201 Created. У тілі відповіді містяться дані створеного проєкту з присвоєним `id`, що свідчить про правильну роботу серверу.

3. Створення нового завдання у проєкті: Перевірка ендпоінту `/circleflow/tasks` на можливість створення нового завдання в

існуючому проєкті з декількома визначеними параметрами завдання (рис. 3.4.3).

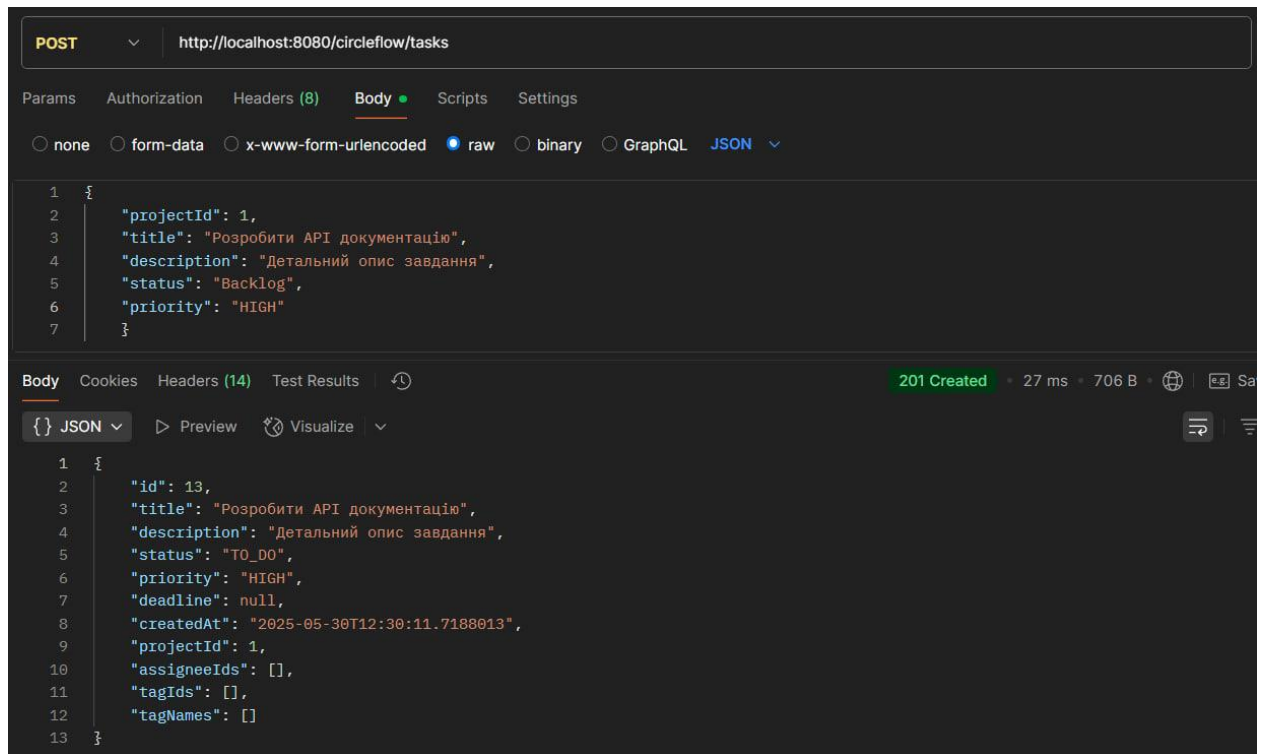


Рисунок 3.4.3 – тестування створення нового завдання

На рисунку зображено запит методом POST на URL `http://localhost:8080/circleflow/tasks`. У заголовку Authorization передано Bearer <JWT_TOKEN>. У відповідь отримано статус 201 Created. У тілі відповіді містяться дані створеного завдання, що свідчить про правильну роботу серверу.

- Отримання списку завдань для проєкту: Перевірка ендпоінту `/circleflow/tasks` з параметром `projectId` на можливість отримання всіх завдань, що належать конкретному проєкту (рис. 3.4.4).

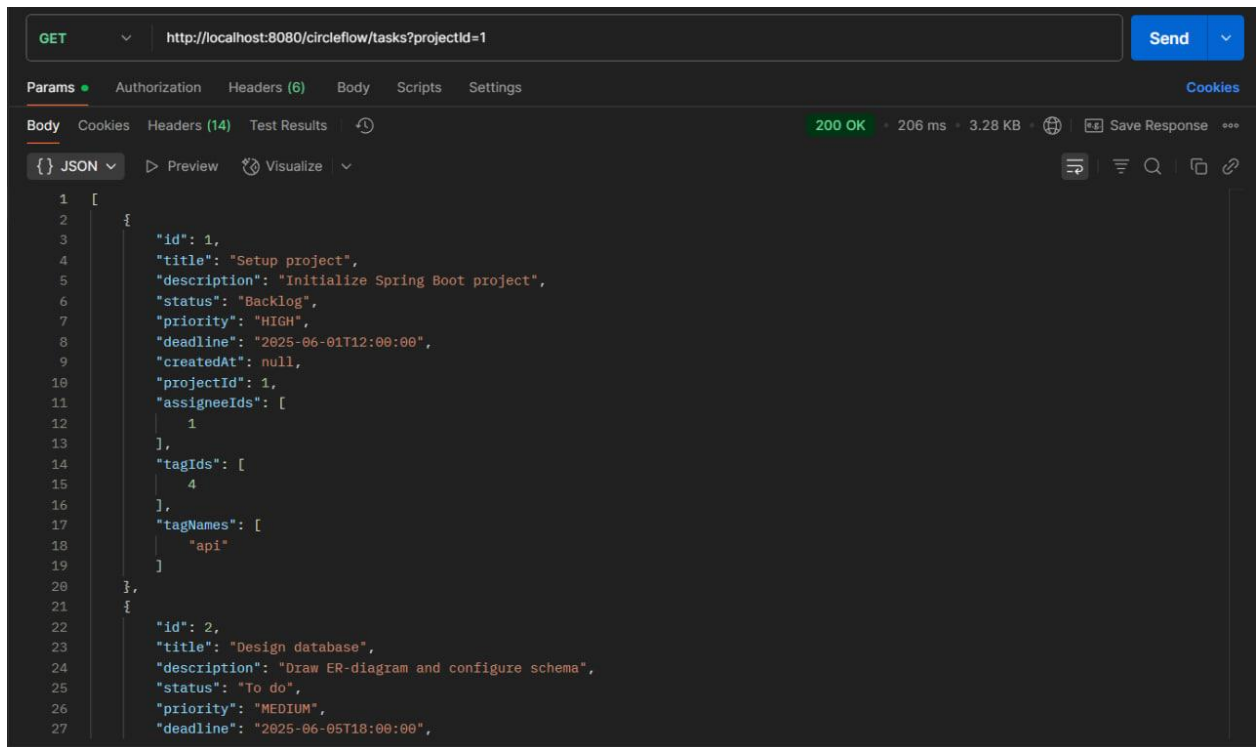


Рисунок 3.4.4 – отримання всіх завдань певного проєкту

На рисунку зображено запит методом GET на URL `http://localhost:8080/circleflow/tasks?projectId=1`. У заголовку Authorization передано Bearer <JWT_TOKEN>. У відповідь отримано статус 200 OK. У тілі відповіді міститься масив об'єктів завдань, що належать проєкту з `id=1`.

Тепер перейдемо до огляду фронтенду. Огляд користувацького інтерфейсу проводився шляхом виконання типових сценаріїв використання засобу "CircleFlow" з боку користувача. Це також дозволило перевірити коректність взаємодії клієнтської частини з бекендом та правильність відображення даних.

Сторінка `ProjectsPage` є головною після автентифікації користувача. Вона надає огляд усіх проєктів, до яких користувач має доступ. Кожен проєкт представлений у вигляді картки, що містить його назву, короткий опис та іншу релевантну інформацію. З цієї сторінки можна перейти до Kanban-дошки конкретного проєкту або ініціювати створення нового. Вигляд головної сторінки додатку зображено на рисунку 3.4.5.

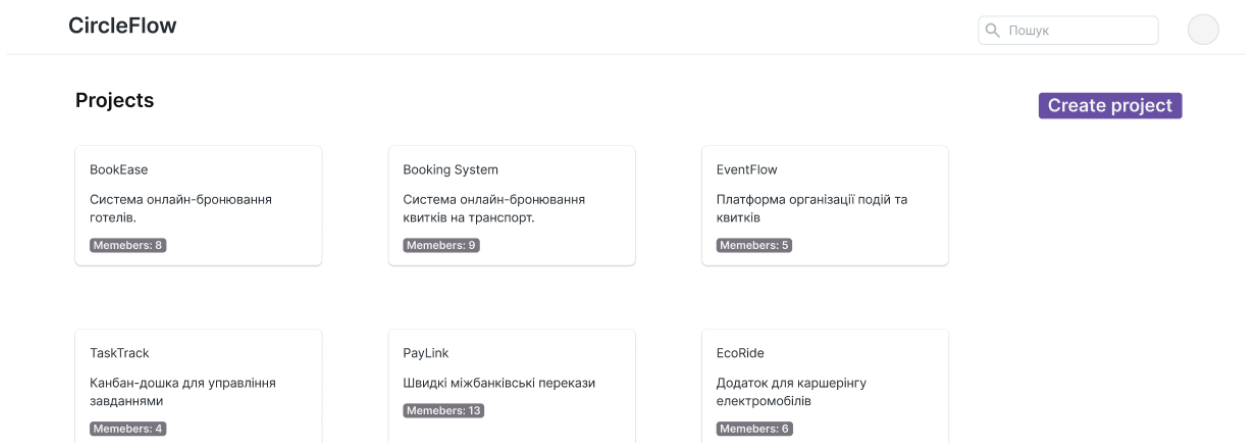


Рисунок 3.4.5 – головна сторінка додатку CircleFlow

Для створення нового проєкту користувач може використати спеціальну форму, яка відкривається у модальному вікні при натисканні кнопки “Create Project”, що знаходиться на головній сторінці. Форма містить поля для введення назви проєкту, його опису та вибору власника. Вигляд форми для створення нового проєкту можна побачити на рисунку 3.4.6.

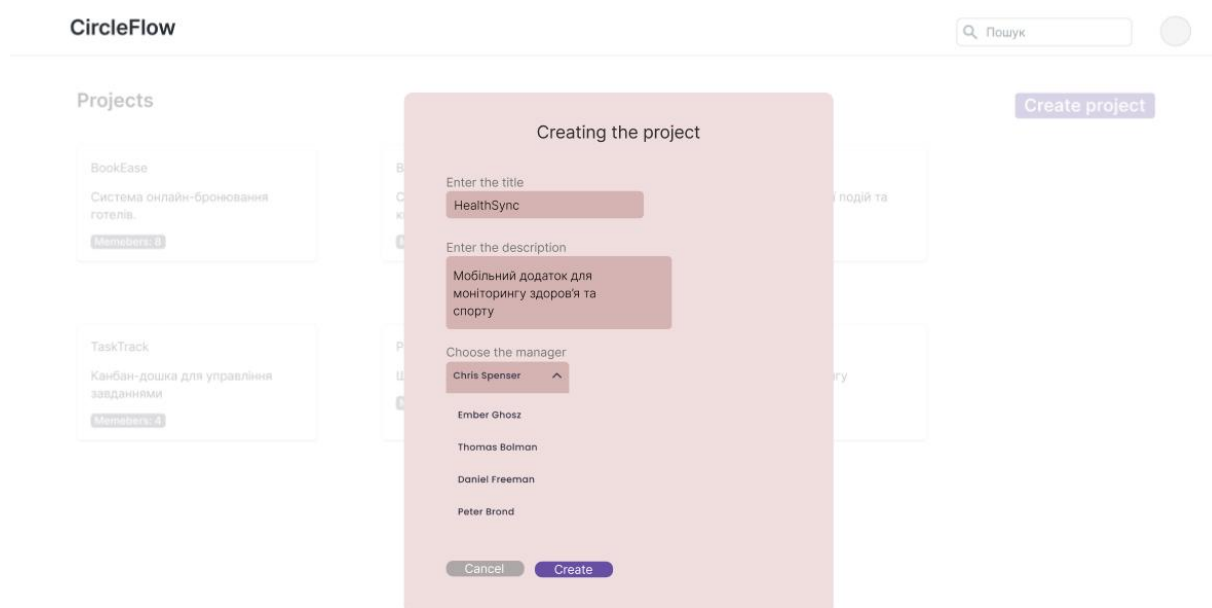


Рисунок 3.4.6 – форма створення нового проєкту

Далі йде Kanban-дошка (ProjectBoardPage), що є головним компонентом інтерфейсу користувача для візуалізації та управління завданнями. Вона складається з кількох колонок, визначаючи різні статуси завдань ("To Do", "In Progress", "QA/Review"). Завдання відображаються у вигляді карток, що можна перетягувати між колонками (рис. 3.4.7).

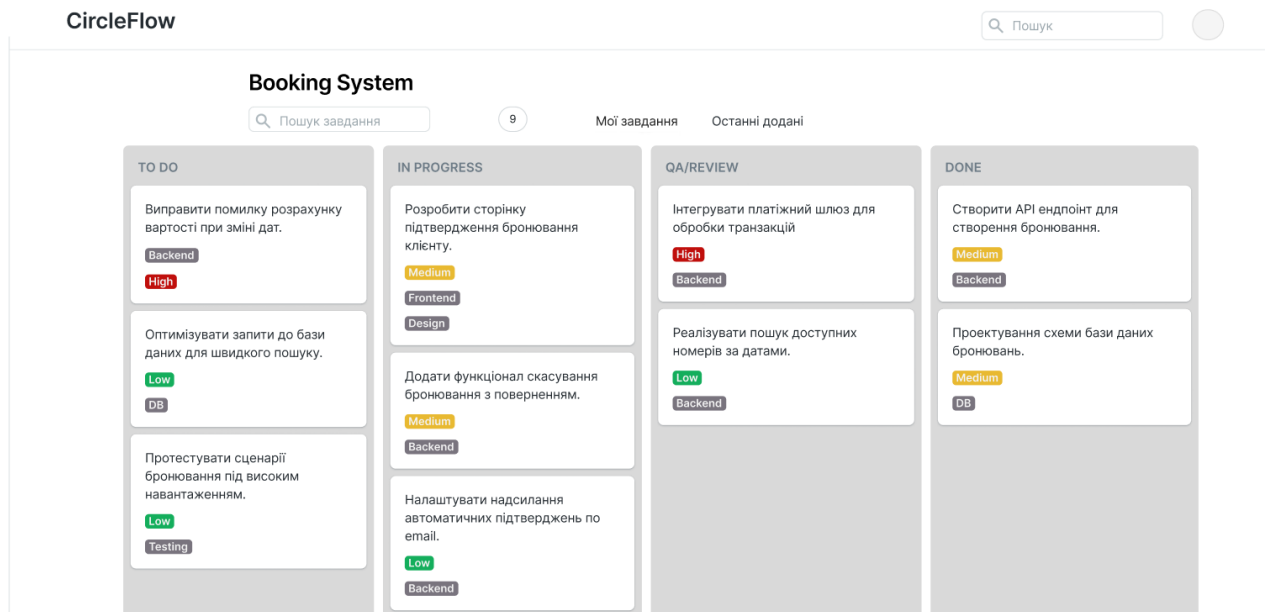


Рисунок 3.4.7 – Kanban – дошка

Рекомендації з ШІ-мікросервісу, такі як прогнозований час виконання завдання та найбільш підходящий виконавець, відображаються користувачеві у контексті конкретного завдання. Це реалізовано у модальному вікні кожного завдання. Вигляд вікна з рекомендаціями зображено на рисунку 3.4.8.

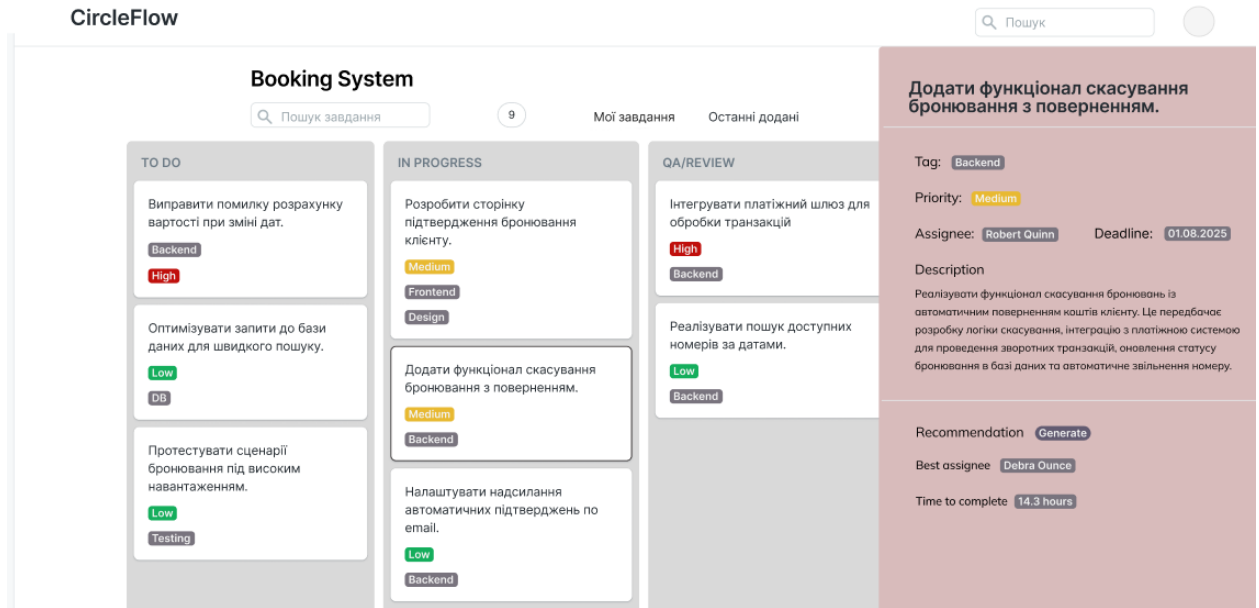


Рисунок 3.4.8 – Модальне вікно з рекомендаціями

Проведене тестування API та огляд ключових сторінок користувацького інтерфейсу підтверджують, що засіб "CircleFlow" реалізовано відповідно до поставлених раніше функціональних вимог. Та те, що він готовий до виконання основних завдань з управління проєктами з інтелектуальними рекомендаціями.

3.4 Результати роботи ШІ-модуля

Для оцінки ефективності розробленого ШІ-модуля, що складається з моделі прогнозування тривалості завдань та моделі рекомендації виконавців, було проведено тестування на відкладених раніше тестових вибірках. Ці вибірки не використовувалися на етапі навчання моделей, що забезпечило об'єктивність отриманих результатів.

Оцінка моделей проводилася на синтетичних даних, згенерованих таким чином, щоб відображати характеристики реальних проєктних завдань та виконавців. Рейтинг ознак та коефіцієнти, що використовувалися моделями наведено в додатку Б.

Для моделі прогнозування тривалості завдань (регресійна модель) використовувалися такі метрики, як Середня абсолютна помилка (MAE) -

показує середню величину помилки прогнозу в одиницях цільової змінної (годинах), Середньоквадратична помилка (RMSE) - вимірює середню величину помилки і сильніше штрафує за відхилення, Коефіцієнт детермінації (R^2) - показує, яку частку варіативності цільової змінної (тривалості) може пояснити модель, Середня абсолютна відсоткова помилка (MAPE) - вимірює середню помилку у відсотках [41].

Для моделі рекомендації виконавців (класифікаційна модель) використовувалися такі метрики, як Матриця помилок (Confusion Matrix) - показує ефективність класифікатора, розділяючи прогнози на True Positives (TP), True Negatives (TN), False Positives (FP) та False Negatives (FN), Точність (Accuracy) - загальна частка правильно класифікованих прикладів, Прецизійність (Precision) - частка об'єктів, дійсно належних до позитивного класу, серед усіх об'єктів, які модель віднесла до такого, Повнота (Recall) - частка об'єктів позитивного класу, які модель правильно ідентифікувала, F1-міра (F1-score) - середнє між прецизійністю та повнотою, ROC-AUC - площа під ROC-кривою (Receiver Operating Characteristic), що характеризує загальну якість розрізнення класів моделлю [40].

Модель RandomForestRegressor була навчена на синтетичному датасеті розміром 1000 зразків, з яких 796 використовувалися для навчання (без викидів) та 200 – для тестування. Оптимізувалися гіперпараметри за допомогою GridSearchCV, що дозволило досягти середнього R^2 на крос-валідації - 0.8877.

Значення $R^2=0.8445$ свідчить про те, що модель має добру пояснювальну здатність щодо тривалості завдань. MAE на рівні 3.83 години є прийнятним для багатьох практичних сценаріїв УП, особливо для завдань середньої та великої тривалості. MAPE моделі було оцінене у 22.38%.

Для візуалізації коректності передбачень моделлю було створено діаграму розсіювання. У ній по осі X відкладено фактичні значення тривалості завдань з тестової вибірки. По осі Y відкладено прогнозовані моделлю значення тривалості. Також на графіку присутня діагональна лінія

$y=x$, що відповідає ідеальному прогнозу (рис. 3.4.9).

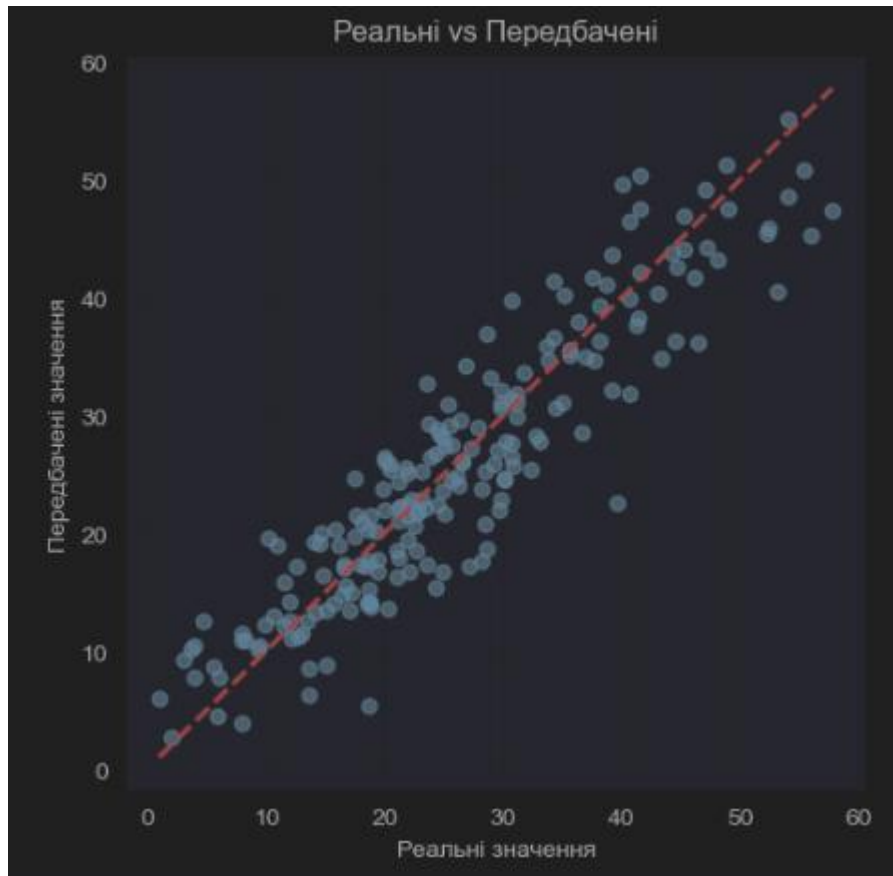


Рисунок 3.4.9 – діаграма розсіювання

Як можна побачити, точки на графіку групуються навколо діагональної лінії. Це і вказує на доволі високий відсоток правильний передбачень.

Також для візуалізації правильної роботи моделі можна використати графік залишків. У ньому осі X відкладено прогнозовані значення тривалості завдань. По осі Y відкладено залишки (різниця між фактичним та прогнозованим значенням). Горизонтальна лінія на рівні $y=0$ вказує на нульову помилку (рис. 3.4.10).

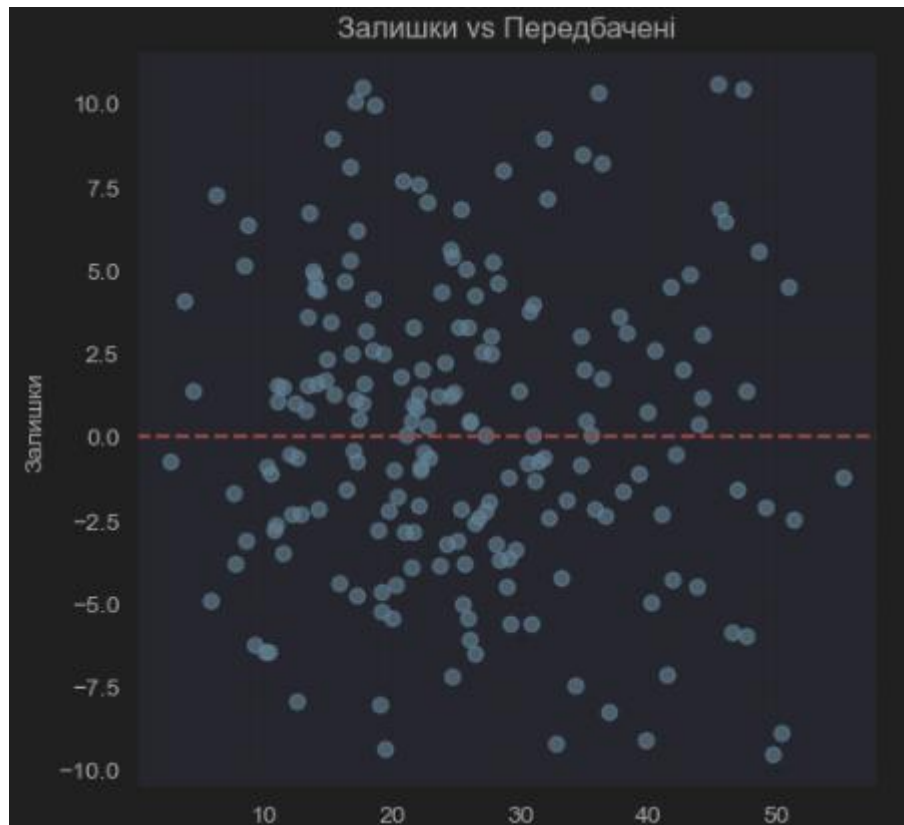


Рисунок 3.4.10 – Графік залишків

В ідеальному випадку залишки мають бути випадково розподілені навколо нуля без чітких патернів, що буде свідчити про відсутність нерівномірної дисперсії помилок. Як можна побачити на рисунку 3.4.10, жодних закономірностей на графіку немає, що свідчить про високу якість реалізованої моделі.

Модель RandomForestClassifier також оцінювалася на тестовій вибірці з 200 зразків.

Для візуалізації коректної роботи моделі можна зобразити матрицю помилок. На основі звіту про класифікацію, матриця помилок для тестової вибірки (200 зразків, з них 118 належать класу “1” – найкращий виконавець, та 82 – класу “0”). На рисунку 3.4.11 зображено матрицю помилок.

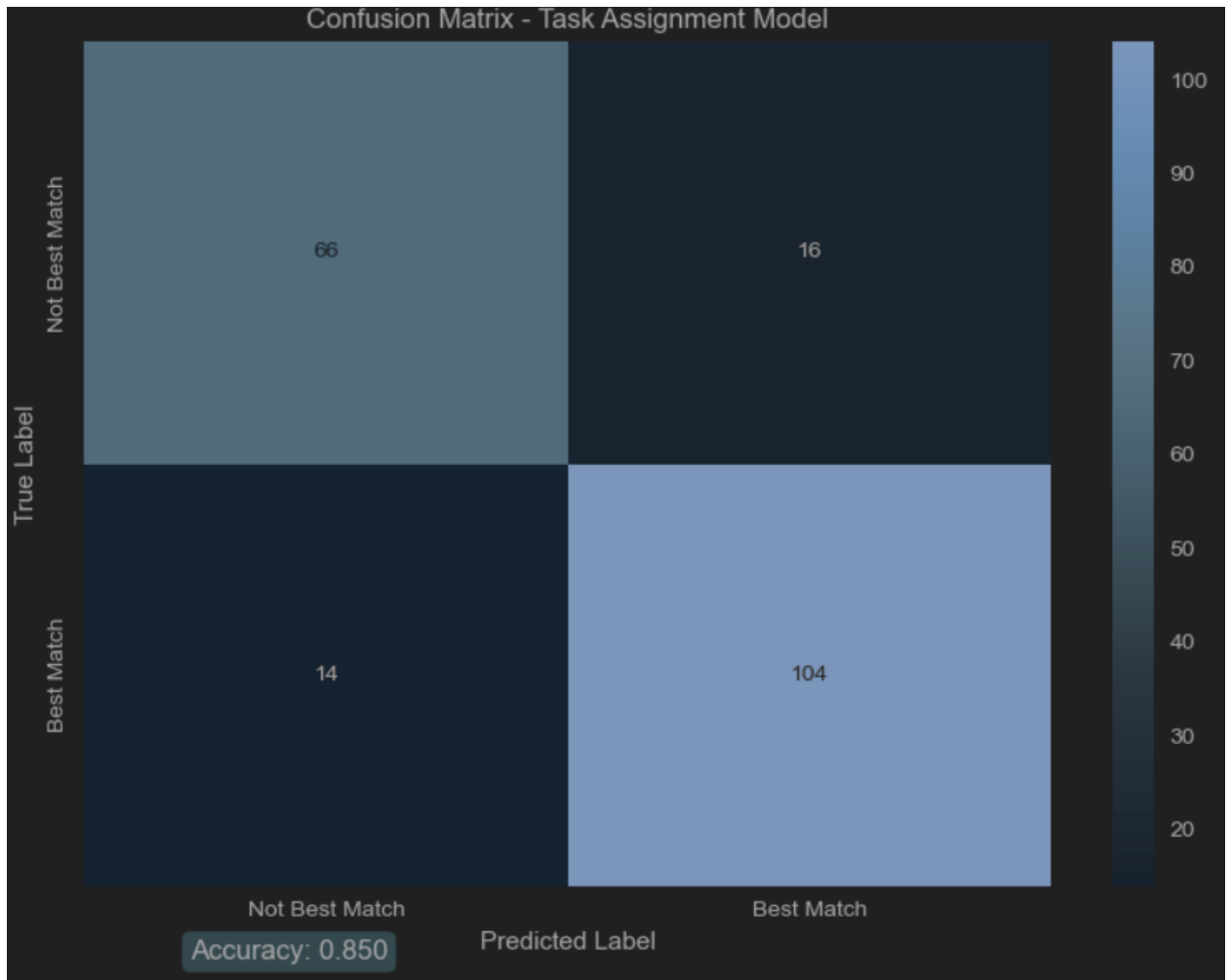


Рисунок 3.4.11 – матриця помилок

Поточний вигляд матриці помилок означає, що модель правильно ідентифікувала 104 найкращих виконавців та 66 не найкращих. Вона помилково віднесла 16 не найкращих виконавців до найкращих та пропустила 14 реально найкращих виконавців, що означає високу якість моделі.

Також ефективність навченої моделі можна визначити за ROC-кривою. На цій кривій на осі X відкладено частку хибно позитивних спрацювань. По осі Y відкладено частку істинно позитивних спрацювань. Також на графіку показана діагональна лінія (ROC-крива для випадкового класифікатора) та значення AUC (рис. 3.4.12).

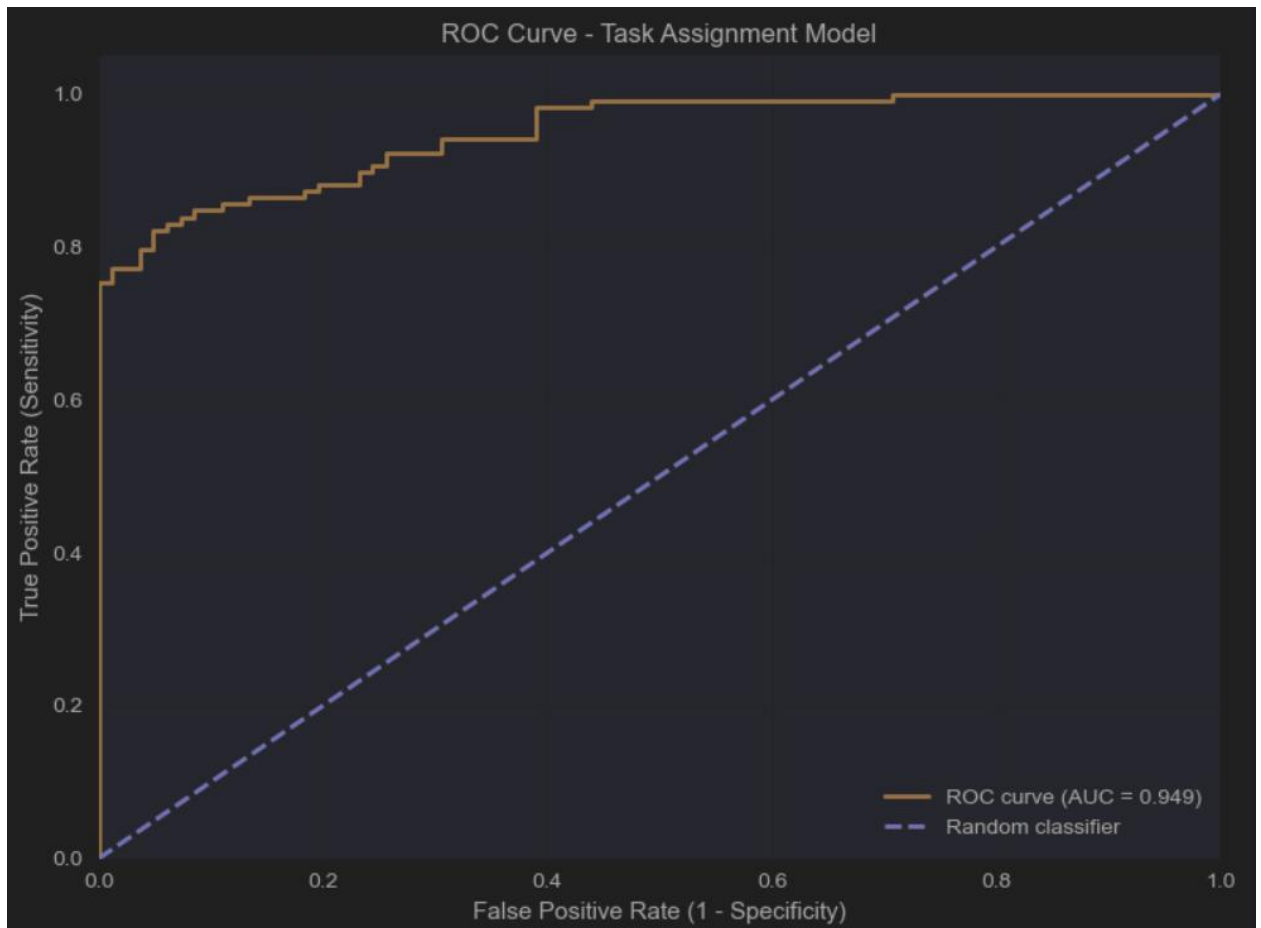


Рисунок 3.4.12 – ROC – крива

Як можна побачити - крива для розробленої моделі значно вище діагональної лінії та наближається до верхнього лівого кута. Це свідчить про високу якість класифікації, а значення $AUC = 0.949$ підтверджує це. Також високі значення Precision (0.87) та Recall (0.88) для класу "найкращий виконавець" свідчать про те, що модель не тільки точно рекомендує, але й знаходить більшість з тих, хто дійсно є найкращим. F1-score (0.87) підтверджує хороший баланс між цими двома метриками.

Загалом, розроблений ШІ-модуль показав хороші результати на синтетичних тестових даних. Модель прогнозування тривалості завдань здатна пояснити значну частину варіативності даних ($R^2 \approx 0.84$) з середньою абсолютною помилкою близько 3.83 години. А модель рекомендації виконавців демонструє високу точність (Accuracy=0.85) та відмінну розрізнявальну здатність (ROC-AUC=0.949), ефективно знаходячи баланс

між точністю рекомендацій та повнотою охоплення найкращих кандидатів.

Досягнутий рівень ефективності є достатнім для використання моделей у засобі "CircleFlow" для надання інтелектуальних рекомендацій.

ВИСНОВКИ

У кваліфікаційній роботі було успішно вирішено задачу розробки концепції та програмного засобу управління проєктами "CircleFlow", що інтегрує Kanban-методологію з інтелектуальними функціями на основі штучного інтелекту для оптимізації планування та розподілу ресурсів. В ході виконання роботи було проведено аналіз сучасних методологій УП (Waterfall, Agile, Scrum, Kanban) та інструментів з їх використанням, а також досліджено застосування ШІ в цій галузі, що підтвердило актуальність розробки; обґрунтовано вибір моделей машинного навчання – RandomForestRegressor для прогнозування тривалості та RandomForestClassifier для надання рекомендацій щодо вибору виконавців; розроблено концептуальну модель та трикомпонентну архітектуру CircleFlow: React SPA, Spring Boot REST API, Python/FastAPI ШІ-мікросервіс, спроектовано структуру бази даних (PostgreSQL); реалізовано програмний прототип засобу: серверна частина на Spring Boot з REST API та JWT-безпекою, ШІ-мікросервіс на FastAPI з ендпоінтами для прогнозування та рекомендацій, клієнтська частина на React з Kanban-дошкою та відображенням ШІ-рекомендацій; проведено функціональне тестування API та тестування інтерфейсу користувача; проведено оцінку моделей ШІ на тестових даних, що показала: $R^2=0.8445$ для моделі прогнозування тривалості та $ROC-AUC = 0.949$ для моделі рекомендації виконавців, що свідчить про їх задовільну якість для ефективної роботи.

Новизною роботи є обґрунтування доцільності інтеграції методів ШІ (Random Forest) в засоби управління проєктами для задач прогнозування та рекомендацій та практична реалізація програмного засобу. Таким чином, мета роботи була досягнута, а основні завдання - виконані.

Розроблене програмне рішення буде основою для розробки повноцінної системи, корисної для малих команд та стартапів. В подальшому планується розширення функціоналу (реалізація модулі звітності та

інтеграції з іншими сервісами), використання додаткових алгоритмів машинного навчання та проведення заходів (збір та використання реальних проєктних даних) для підвищення точності моделей ШІ.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK® Guide). 5th ed. Newtown Square, PA: Project Management Institute, 2013. 589 p. URL: <https://ia800105.us.archive.org/18/items/a-guide/a-guide.pdf>. Дата звернення: 24.03.2025.
2. Sweeney M. Agile vs Waterfall: Which Method is More Successful? // Clearcode. URL: <https://clearcode.cc/blog/agile-vs-waterfall-method/>. Дата звернення: 25.03.2025
3. Flexi-Project. Маніфест Agile: ключові цінності та принципи. URL: <https://shorturl.at/P2Bdi>. Дата звернення: 25.03.2025.
4. Schwaber K., Sutherland J. Посібник зі Скраму. URL: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Ukrainian.pdf>. Дата звернення: 27.03.2025.
5. Trasy. Kanban board. URL: <https://tra.cy/uk/blog/kanban-board>. Дата звернення: 28.03.2025.
6. Atlassian. Working with WIP limits for kanban. URL: <https://www.atlassian.com/agile/kanban/wip-limits>. Дата звернення: 24.05.2025
7. Silicon IT Hub. Jira vs Asana vs Trello: How to Choose the Best Software. URL: <https://www.siliconithub.com/blog/jira-vs-asana-vs-trello/>. Дата звернення: 30.03.2025.
8. Unique Software Development. Pros and Cons of Waterfall Methodology in Software Development. URL: <https://shorturl.at/fk5yF>. Дата звернення: 31.03.2025.
9. University of Minnesota, College of Continuing and Professional Studies (CCAPS). Agile Methodology: Advantages and Disadvantages. URL: <https://ccaps.umn.edu/story/agile-methodology-advantages-and-disadvantages>. Дата звернення: 31.03.2025.

10. The Digital Project Manager. Why is it Difficult to Use Project Management Software Well?. URL: <https://shorturl.at/Ix4sV>. Дата звернення: 02.04.2025.
11. Ghosh S. Leveraging Artificial Intelligence for Optimized Project Management // World Journal of Advanced Research and Reviews 2024. Vol. 18 No. 2. P. 45–52. URL: <https://wjarr.com/sites/default/files/WJARR-2024-4026.pdf>. Дата звернення: 02.04.2025.
12. Thompson D. The Human Factor in Project Management. New York, 2018, 198 p.
13. Heidrich V. The Power of Artificial Intelligence in Project Management: A Review and Evaluation Study // Proceedings of the 5th International Conference on Big Data, IoT and Blockchain. 2024. P. 35–48. DOI: 10.5121/csit.2024.142403. URL: <https://shorturl.at/Vyl56>. Дата звернення: 05.04.2025.
14. Birdview. Aujla A. How Machine Learning Will Change the World of Project Management. 16.05.2025. URL: <https://shorturl.at/dSco5>. Дата звернення: 07.04.2025.
15. BannerBoo. Летюга Ю. Штучний інтелект в управлінні проєктами. 30.10.2024. URL: <https://bannerboo.com/ua/blog/shtuchnyy-intelekt-v-upravlinni-proyektamy/>. Дата звернення: 08.04.2025.
16. Sarkar S., Gupta R., Mehta P. Integrating AI into Project Management: Challenges and Opportunities // Production Planning & Control. 2024. Vol. 35, No. 7. P. 567–578. DOI: 10.1080/09537287.2024.2320790. URL: <https://www.tandfonline.com/doi/full/10.1080/09537287.2024.2320790>. Дата звернення: 09.04.2025.
17. IBM. What Is NLP (Natural Language Processing)?. 11.08.2024. URL: <https://www.ibm.com/think/topics/natural-language-processing>. Дата звернення: 10.04.2025.
18. Thakur R., Manikandan N., Kumar S.G., Jayaraj R. Task Management System using AI Prioritizations // International Journal of Novel Research and Development. 2024. Vol. 9, Issue 3. P. e1–e8. URL:

<https://www.ijnrd.org/papers/IJNRD2403401.pdf>. Дата звернення:
13.04.2025.

19. Wang Y., Li X., Zhang Q. Applying Machine Learning to Estimate the Effort and Duration of Individual Tasks in Software Projects // ResearchGate. 2024. URL: <https://shorturl.at/RVIhh>. Дата звернення: 14.04.2025.
20. ClickUp. How to Use AI in Project Management. URL: <https://clickup.com/blog/how-to-use-ai-project-management/>. Дата звернення: 14.04.2025.
21. Celoxis. AI & ML in Project Management: A Comprehensive Guide. URL: <https://www.celoxis.com/article/ai-ml-project-management>. Дата звернення: 19.04.2025.
22. ProjectManagement.com. Seven AI-Based Ethical Issues for Project Managers. URL: <https://www.projectmanagement.com/blog-post/75502/seven-ai-based-ethical-issues-for-project-managers>. Дата звернення: 20.04.2025.
23. AltexSoft. Functional and Nonfunctional Requirements: Specification and Types. URL: <https://www.altexsoft.com/blog/functional-and-non-functional-requirements-specification-and-types/>. Дата звернення: 23.04.2025.
24. Syncfusion. Secure JWT Storage Best Practices. URL: <https://www.syncfusion.com/blogs/post/secure-jwt-storage-best-practices/amp>. Дата звернення: 25.04.2025.
25. HelpWare.Tech. Zavgorodniy A. ISO 25010: Enhancing Our Software Quality Management Process. URL: <https://tech.helpware.com/blog/iso-25010-enhancing-our-software-quality-management-process>. Дата звернення: 26.04.2025.
26. Legit Security. Web Application Security Requirements. URL: <https://www.legitsecurity.com/aspm-knowledge-base/web-application-security-requirements>. Дата звернення: 26.04.2025.
27. Software Sustainability Institute. Developing Maintainable Software. URL: <https://www.software.ac.uk/guide/developing-maintainable-software>. Дата звернення: 28.04.2025.

28. Timescale. Optimizing Your Database: A Deep Dive into PostgreSQL Data Types. URL: <https://www.timescale.com/learn/optimizing-your-database-a-deep-dive-into-postgresql-data-types>. Дата звернення: 27.05.2025.
29. CelerData. Composite Keys. 29.04.2024. URL: <https://celerddata.com/glossary/composite-keys>. Дата звернення: 27.05.2025.
30. DataCamp. Third Normal Form (3NF) Explained. URL: <https://www.datacamp.com/tutorial/third-normal-form>. Дата звернення: 30.04.2025.
31. GitHub. System Design. URL: <https://github.com/karanpratapsingh/system-design>. Дата звернення: 01.05.2025.
32. Semaphore. Why You Need an API Layer in Your Spring Boot App and How to Implement It. URL: <https://semaphore.io/blog/api-layer-spring-boot>. Дата звернення: 02.05.2025.
33. EverSQL. Java ORM – Hibernate vs Spring Data JPA vs EclipseLink vs JPA vs JDBC. URL: <https://www.eversql.com/java-orm-hibernate-vs-spring-data-jpa-vs-eclipselink-vs-jpa-vs-jdbc/>. Дата звернення: 04.05.2025.
34. Stack Overflow Blog. Best practices for REST API design. URL: <https://stackoverflow.blog/2020/03/02/best-practices-for-rest-api-design/>. Дата звернення: 08.05.2025.
35. eSparkInfo. React Component-based Architecture: Build Modern UIs Easily. URL: <https://www.esparkinfo.com/software-development/technologies/reactjs/component-based-architecture>. Дата звернення: 10.05.2025.
36. scikit-learn. `sklearn.ensemble.RandomForestRegressor` — scikit-learn 1.6.1 documentation. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>. Дата звернення: 11.05.2025.
37. scikit-learn. `sklearn.ensemble.RandomForestClassifier` — scikit-learn 1.6.1 documentation. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier>.

html. Дата звернення: 11.05.2025.

38. Analytics Vidhya. Understanding Random Forest. URL:

<https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/>.

Дата звернення: 12.05.2025.

39. Google Developers. Classification: Accuracy, recall, precision, and related metrics. URL: <https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall>. Дата звернення: 12.05.2025.

40. Machine Learning Mastery. Regression Metrics for Machine Learning. URL: <https://www.machinelearningmastery.com/regression-metrics-for-machine-learning/>. Дата звернення: 12.05.2025.

41. IBM. What is a feature engineering? URL:

<https://www.ibm.com/think/topics/feature-engineering>. Дата звернення: 02.06.2025.

42. scikit-learn. Cross-validation: evaluating estimator performance. URL:

https://scikit-learn.org/stable/modules/cross_validation.html. Дата звернення: 15.05.2025.

43. LinkedIn. The Controller-Service-Repository Pattern: A Comprehensive Guide. URL: <https://www.linkedin.com/pulse/controller-service-repository-pattern-comprehensive-guide-shikder-azicc>. Дата звернення: 16.05.2025

ДОДАТОК А

СКРИПТИ ДЛЯ СТВОРЕННЯ ВСІХ ТАБЛИЦЬ БАЗИ ДАНИХ ЗАСОБУ “CIRCLEFLOW”

Лістинг А.1 – SQL-скрипти для створення таблиць в БД засобу “CircleFlow”

```
CREATE TABLE users (  
    id SERIAL PRIMARY KEY,  
    username VARCHAR(100) NOT NULL,  
    email VARCHAR(100) UNIQUE NOT NULL,  
    password_hash TEXT NOT NULL,  
    role VARCHAR(50) NOT NULL  
);  
  
CREATE TABLE user_skills (  
    user_id INT NOT NULL REFERENCES users(id),  
    skill_id INT NOT NULL REFERENCES skills(id),  
    PRIMARY KEY (user_id, skill_id)  
);  
  
CREATE TABLE projects (  
    id SERIAL PRIMARY KEY,  
    name VARCHAR(255) NOT NULL,  
    description TEXT,  
    owner_id INT NOT NULL REFERENCES users(id)  
);  
  
CREATE TABLE project_members (  
    id SERIAL PRIMARY KEY,  
    project_id INT NOT NULL REFERENCES projects(id),  
    user_id INT NOT NULL REFERENCES users(id),  
    role VARCHAR(50)  
);  
  
CREATE TABLE tasks (  
    id SERIAL PRIMARY KEY,  
    project_id INT NOT NULL REFERENCES projects(id),
```

```

    title VARCHAR(255) NOT NULL,
    description TEXT,
    status VARCHAR(50) NOT NULL,
    priority VARCHAR(20),
    deadline TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE task_assignees (
    task_id INT NOT NULL REFERENCES tasks(id),
    user_id INT NOT NULL REFERENCES users(id),
    PRIMARY KEY (task_id, user_id)
);

CREATE TABLE task_tags (
    task_id INT NOT NULL REFERENCES tasks(id),
    tag_id INT NOT NULL REFERENCES tags(id),
    PRIMARY KEY (task_id, tag_id)
);

CREATE TABLE recommendations (
    task_id INT NOT NULL REFERENCES tasks(id),
    recommended_user_id INT NOT NULL REFERENCES users(id),
    predicted_time_hours FLOAT,
    confidence_score FLOAT,
    explanation TEXT,
    PRIMARY KEY (task_id, recommended_user_id)
);

CREATE TABLE activity_log (
    id SERIAL PRIMARY KEY,
    project_id INT NOT NULL REFERENCES projects(id),
    user_id INT NOT NULL REFERENCES users(id),
    task_id INT REFERENCES tasks(id),
    action VARCHAR(255),
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

```

ДОДАТОК Б

ДІАГРАМА РЕЙТИНГУ ТА КОЕФІЦІЄНТІВ ВИКОРИСТАНИХ ПРИ НАВЧАННІ МОДЕЛЕЙ МН

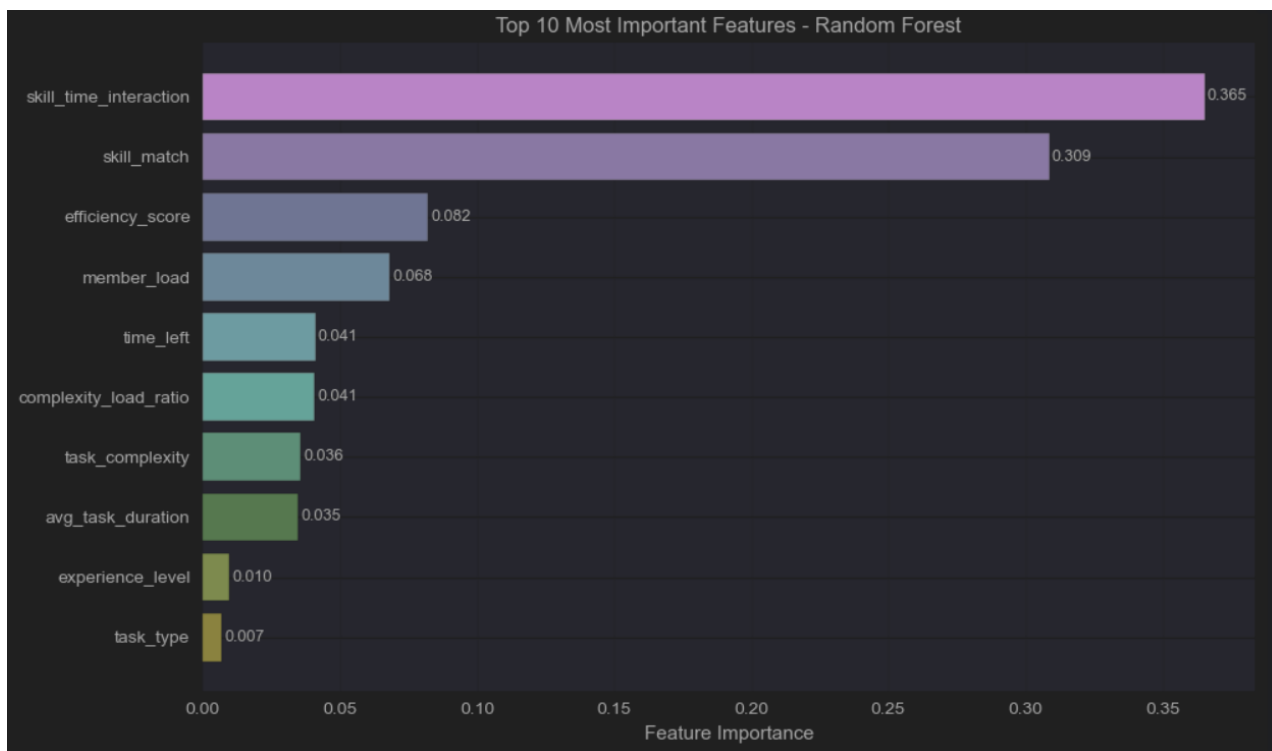


Рисунок Б.1 – діаграма рейтингу використаних ознак