

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Інтелектуальна система підтримки прийняття
рішень для планування та оптимізації ресурсів ІТ-
проєктів»**

Виконав студент групи ІПЗм-24-2
ОБМІНЯНИЙ Денис Сергійович

Керівник роботи:
ЧИЖМОТРЯ Олексій Володимирович

Рецензент:
ТОЛСТОЙ Ігор Анатолійович

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«18» вересня 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ОБМІНЯНИЙ Денис Сергійович**

Керівник роботи: **ЧИЖМОТРЕ Олексій Володимирович**

Тема роботи: **«Інтелектуальна система підтримки прийняття рішень для планування та оптимізації ресурсів ІТ-проектів»**,

затверджена Наказом закладу вищої освіти від **«18» вересня 2025 р. № 491/с**

Вихідні дані для роботи: **моделі та алгоритми штучного інтелекту, що застосовуються для оптимізації планування задач, прогнозування дедлайнів та автоматичного вибору виконавця і процес управління людськими ресурсами в ІТ-проектах.**

Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.В.	18.09.2025р.	18.09.2025р.
2	Чижмотря О.В.	18.09.2025р.	18.09.2025р.
3	Чижмотря О.В.	18.09.2025р.	18.09.2025р.

РЕФЕРАТ

Кваліфікаційна робота освітнього ступеня «магістр» присвячена автоматизації прийняття рішень для планування та оптимізації ресурсів ІТ-проектів. Пояснювальна записка складається з вступу, трьох розділів та списку використаної літератури. Пояснювальна записка до кваліфікаційної магістерської роботи містить 88 сторінок, 26 ілюстрацій, 2 додатки та 40 посилань на зовнішні джерела.

Метою роботи є розробка інтелектуальної системи підтримки прийняття рішень для планування та оптимізації ресурсів ІТ-проектів, здатної аналізувати дані про виконавців, задачі та історію їх виконання і формувати рекомендації щодо оптимального розподілу робочого навантаження та прогнозування строків. Розроблена система дозволяє автоматизувати процес призначення задач виконавцям з урахуванням їхніх навичок, завантаженості та історичної продуктивності, підвищити точність прогнозування строків виконання та зменшити ризики перевантаження ресурсів, що робить її придатною до використання в реальних ІТ-командах. Проведено тестування програми.

КЛЮЧОВІ СЛОВА: ІНТЕЛЕКТУАЛЬНА СИСТЕМА, ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ, RANDOM FOREST, ПЛАНУВАННЯ РЕСУРСІВ ІТ-ПРОЄКТІВ.

ABSTRACT

The master's qualification thesis is devoted to the automation of decision-making processes for planning and optimizing resources in IT projects. The explanatory report consists of an introduction, three chapters, and a list of references. The thesis note contains 88 pages of main text, 26 illustrations, 2 appendices, and 40 references to external sources.

The aim of the work is to develop an intelligent decision support system for planning and optimizing resources in IT projects. The system is designed to analyze data about performers, tasks, and historical performance, as well as to generate recommendations for optimal workload distribution and deadline prediction. The developed system automates the process of assigning tasks to team members by considering their skills, current workload, and historical productivity. This significantly improves the accuracy of deadline forecasting and reduces the risk of resource overload, making the solution suitable for practical use in real IT teams. Software testing was conducted to verify the system's functionality and effectiveness.

KEYWORDS: INTELLIGENT SYSTEM, ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, RANDOM FOREST, IT PROJECT RESOURCE PLANNING.

ЗМІСТ

Вступ.....	7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	10
1.1. Особливості управління ІТ-проектами в сучасних умовах	10
1.2. Аналіз існуючих програмних рішень та інструментів.....	13
1.3. Методи прийняття рішень у проєктному менеджменті.....	15
1.4. Постановка задачі дослідження	20
Висновки до розділу 1	21
РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ.....	22
2.1. Архітектура та концептуальна модель системи	22
2.2. Вибір моделі машинного навчання та методів обробки даних.....	25
2.3. Проєктування бази даних.....	27
2.4. Алгоритм інтелектуального призначення задач.....	30
Висновки до розділу 2.....	39
РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ.....	41
3.1. Реалізація серверної частини системи.....	41
3.2. Реалізація клієнтської частини та інтерфейсу користувача	46
3.3. Модуль AI-рекомендацій та обробка результатів	57
3.4. Інтеграція AI-модуля в систему	59
3.5. Система логування, аналітики та зберігання історії взаємодій	62
Висновки до розділу 3	65
ВИСНОВКИ	67
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	69
ДОДАТКИ	74

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

- AI – (англ. artificial intelligence) – штучний інтелект
- API – (англ. application programming interface) – програмний інтерфейс взаємодії
- DB – (англ. database) – база даних
- ML – (англ. machine learning) – машинне навчання

ВСТУП

Актуальність теми. У сучасних умовах розвитку інформаційних технологій значна частина проєктів потребує ефективного управління ресурсами, точного планування задач, прогнозування строків виконання та раціонального розподілу навантаження між виконавцями. Зі збільшенням складності ІТ-проєктів, а також кількості задіяних фахівців, виникає потреба у системах, що можуть автоматично аналізувати людські ресурси, навички працівників, пріоритети задач та зовнішні фактори, які впливають на продуктивність.

Традиційно такі процеси виконуються вручну менеджерами проєктів або аналітиками. Проте цей підхід часто призводить до перевантаження персоналу, неефективних управлінських рішень та втрати часу на постійне ручне перепланування. Людський фактор залишається ключовою проблемою: неправильна оцінка навантаження, суб'єктивність рішень, неузгоджений тайм-менеджмент і труднощі прогнозування строків виконання задач знижують загальну ефективність управління ІТ-проєктами.

Одним із підходів є використання класичних систем управління проєктами. Проте такі інструменти часто є лише системами фіксації, а не прийняття рішень, і не містять механізмів адаптивного призначення задач відповідно до навичок фахівців, їх зайнятості чи історичної продуктивності.

Альтернативним та більш прогресивним підходом є впровадження інтелектуальних систем на основі штучного інтелекту, які здатні автоматично визначати оптимального виконавця, пропонувати терміни виконання, знаходити помилки в плануванні та прогнозувати ризики затримок. Завдяки використанню алгоритмів машинного навчання така система набуває здатності адаптуватися до змін у команді, специфіки проєктів та реальних результатів виконання задач.

Мета роботи – розробка інтелектуальної системи підтримки прийняття рішень для планування та оптимізації ресурсів ІТ-проєктів, здатної аналізувати дані про виконавців, задачі та історію виконання, а також формувати рекомендації щодо оптимального розподілу робочого навантаження.

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

1. Провести аналіз існуючих підходів та інструментів управління IT-проєктами.
2. Розробити архітектуру програмної системи з інтегрованим модулем рекомендацій.
3. Реалізувати базу даних для зберігання інформації про задачі, виконавців і продуктивність.
4. Розробити алгоритм призначення задач та прогнозування строків.
5. Інтегрувати механізми машинного аналізу даних у процес формування рекомендацій.
6. Створити програмний прототип системи та провести його тестування.

Об'єктом дослідження є процес управління людськими ресурсами в IT-проєктах.

Предметом дослідження є моделі та алгоритми штучного інтелекту, що застосовуються для оптимізації планування задач, прогнозування дедлайнів та автоматичного вибору виконавця.

Наукова новизна роботи полягає у побудові адаптивного рекомендаційного модуля на основі аналізу реальної історії продуктивності працівників, рівня їх навантаження, складності задач та бізнес-пріоритетів. Особливістю підходу є здатність системи навчатися на історичних даних і вдосконалювати точність рекомендацій у процесі використання, що відкриває можливості застосування в широкому спектрі IT-проєктів різної складності.

Практичне значення роботи полягає у розробці інтелектуальної системи підтримки прийняття рішень для планування та оптимізації ресурсів IT-проєктів, яка може бути інтегрована в існуючі корпоративні системи управління задачами. Запропонована модель поєднує класичні підходи проєктного менеджменту з методами машинного навчання, що дозволяє підвищити точність прогнозування строків виконання, зменшити ризики перевантаження працівників та

забезпечити більш ефективний розподіл задач між виконавцями. Результати роботи можуть бути використані у практичній діяльності проєктних менеджерів, а також у розробці комерційного програмного забезпечення для автоматизації планування.

Апробація. За темою кваліфікаційної магістерської роботи були підготовлені та подані до друку тези [1], [2].

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Особливості управління ІТ-проєктами в сучасних умовах

Управління ІТ-проєктами сьогодні має свої специфічні риси, що відрізняють його від проєктної діяльності в інших галузях. Стрімкий розвиток технологій, динамічність вимог, висока конкуренція та необхідність швидкого впровадження цифрових рішень формують складне середовище, у якому проєктні команди працюють у режимі постійних змін.

Однією з ключових особливостей сучасного управління ІТ-проєктами є високий рівень невизначеності. Багато рішень не можуть бути точно сплановані на початковому етапі через змінність бізнес-потреб та технологічних інструментів. Це зумовило активне поширення гнучких методологій - Agile, Scrum, Kanban, Lean Software Development, - які орієнтовані на поступове постачання цінності, можливість швидкої адаптації та короткі цикли зворотного зв'язку [1].

Ще однією важливою особливістю є складність і різноманітність команд. В ІТ-проєктах часто працюють фахівці різного профілю: розробники, дизайнери, тестувальники, DevOps-інженери, бізнес-аналітики, спеціалісти з кібербезпеки. При цьому вони можуть бути розподілені географічно, працювати в різних часових поясах і мати різний рівень досвіду. Це підсилює значення інструментів для планування завантаження, синхронізації роботи та комунікації.

Важливим аспектом є і швидка зміна технологій. Те, що актуально сьогодні, через кілька місяців може вже не відповідати ринку. Відповідно, системи управління ІТ-проєктами мають враховувати постійне оновлення компетенцій членів команди, а також динамічні зміни у складності та тривалості виконання завдань [2].

Крім того, на ефективність управління суттєво впливає людський фактор. Продуктивність спеціалістів може відрізнятися залежно від навичок, особистої мотивації, стрес-навантаження чи попереднього досвіду роботи з подібними

задачами. Це ускладнює прогнозування виконання задач у класичних моделях планування, де враховується лише час і ресурс.

Нарешті, сучасне управління IT-проєктами характеризується високою потребою у прийнятті рішень на основі даних. Інструменти, які використовують машинне навчання, аналітику, автоматизацію планування та прогнозування ризиків, все частіше стають необхідністю, а не додатковою функцією.

У сукупності ці фактори роблять управління IT-проєктами складним процесом, який потребує не лише класичних методів планування, а й інтелектуальних підходів, здатних адаптуватися до змін, прогнозувати результати та допомагати приймати управлінські рішення на основі реальних даних.

Однією з найскладніших задач у сучасному управлінні IT-проєктами є ефективне планування ресурсів [3]. На відміну від традиційних виробничих процесів, робота у сфері інформаційних технологій має високу мінливість, залежність від контексту, людського фактору та непередбачуваних ризиків. Навіть за наявності чітких бізнес-вимог, проєктних графіків і визначених компетенцій команди, розподіл задач між виконавцями рідко буває рівномірним.

Однією з ключових проблем є різниця між плановим та фактичним завантаженням команди. На етапі планування розробники, тестувальники або дизайнери отримують оціночні часові рамки виконання задач. У реальності ж виявляються фактори, які впливають на строки виконання: технічні блокери, неповні або змінені вимоги, залежності між командами, технічний борг, відсутність доступу до середовищ, людський фактор тощо.

На графіку нижче (рис. 1.1) представлено умовний приклад порівняння планових і фактичних витрат часу на виконання задач за різними ролями в команді:

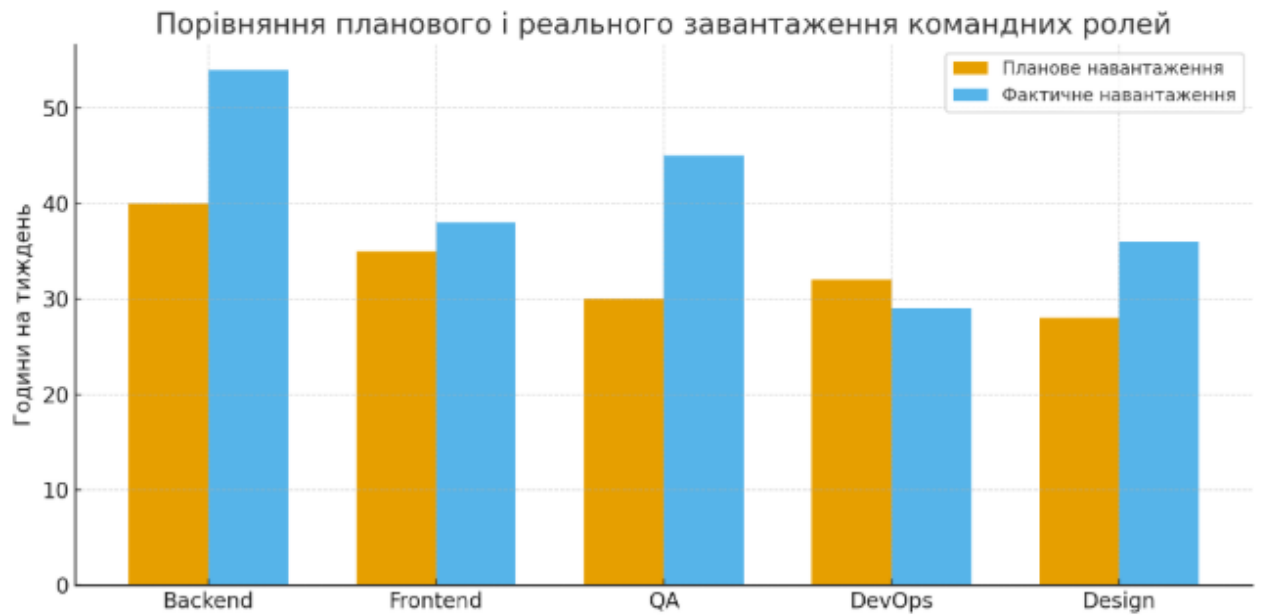


Рисунок 1.1 – Статистика за 2024 рік

Такий розрив показників є типовим для ІТ-проектів, у тестувальників та бекенд-розробників помітне перевищення фактичного навантаження порівняно з плановим. Це свідчить про те, що планування ресурсів у ручному форматі є недостатньо точним і часто не враховує прихованих залежностей та повторюваних шаблонів роботи.

Ще однією проблемою є нерівномірність розподілу задач між членами команди [4]. Деякі спеціалісти отримують постійне перевантаження через унікальні компетенції або експертність, у той час як інші залишаються менш залученими. Це призводить до:

1. затримок у проекті;
2. нерівномірного вигорання фахівців;
3. падіння продуктивності;
4. зниження якості виконаних задач;
5. збільшення ризиків пропуску дедлайнів.

Також ускладнює ситуацію постійна динаміка пріоритетів. Проекти можуть змінювати напрямок розробки, бізнес-цілі або релізні дати. У результаті перерозподіл задач із урахуванням доступності та компетенцій виконавців потребує значних часових та когнітивних зусиль з боку менеджера.

Саме ці фактори підтверджують необхідність застосування інтелектуальних систем, здатних аналізувати історичні дані, прогнозувати навантаження, адаптуватися до змін, рекомендувати оптимальних виконавців, автоматично оцінювати строки виконання задач.

Таким чином, проблема планування людських ресурсів у командній розробці є не лише організаційним викликом, а й технологічним завданням, яке потребує впровадження моделей машинного навчання для забезпечення точності, передбачуваності та стабільності проєктних процесів.

1.2. Аналіз існуючих програмних рішень та інструментів

Сучасний ринок інструментів для управління ІТ-проєктами є досить розвиненим, і на сьогодні існує значна кількість програмних платформ, які пропонують функціональність планування задач, відстеження прогресу, призначення виконавців та контролю дедлайнів. Проте, попри широкий функціонал, більшість із цих систем все ще покладаються на ручне управління ресурсами та не використовують повноцінні адаптивні алгоритми штучного інтелекту, що аналізують поведінку команди та реальні сценарії виконання задач.

Серед найпоширеніших інструментів, що застосовуються у сфері управління проєктами, можна виділити такі платформи, як Jira, Trello, Asana, ClickUp, Monday.com, а також спеціалізовані системи типу Azure DevOps та YouTrack. Кожен із цих продуктів має свої переваги, але водночас демонструє характерні обмеження, які стали поштовхом до розробки нових рішень.

Jira є найбільш популярною системою в ІТ-командах, оскільки глибоко інтегрується в процеси Agile-розробки, підтримує Scrum-дошки, Kanban-моделі та гнучке налаштування робочих процесів. Однак аналіз призначення задач та прогнозування строків виконання залишається на відповідальності менеджера або Scrum-майстра, а рекомендаційні технології застосовуються лише частково.

Trello орієнтований на спрощену модель керування задачами й підходить невеликим командам, однак його можливості щодо стратегічного планування, аналітики та прогнозування обмежені. Подібні інструменти надають зручний інтерфейс, проте не виконують інтелектуального аналізу даних.

Asana, ClickUp та Monday.com роблять крок уперед та надають елементи автоматизації: автоматичне нагадування про завершення задач, можливість створення правил, інтеграцію з календарями та системами звітності. Однак ці механізми також базуються на попередньо визначених правилах, а не на моделі машинного навчання, що адаптується до поведінки користувачів.

Додатково варто відзначити, що більшість існуючих систем управління проєктами зосереджені переважно на фіксації поточного стану задач та забезпеченні зручного інструментарію для комунікації між учасниками команди. Незважаючи на наявність великої кількості метрик, звітів і діаграм, ці інструменти рідко надають можливість інтерпретувати отримані дані у вигляді рекомендацій, які могли б реально підтримувати прийняття рішень. Команди отримують значний обсяг інформації, але саме перетворення цієї інформації на практичні управлінські дії залишається переважно ручною роботою. Така ситуація сповільнює процеси планування, підвищує ризик помилок і зменшує здатність реагувати на зміни в умовах високої динамічності ІТ-проєктів. У результаті менеджер змушений виконувати аналітичні операції, які можуть бути автоматизовані засобами інтелектуального аналізу даних.

Ще однією важливою проблемою наявних платформ є відсутність індивідуалізованого підходу до виконавців. Сучасні інструменти дозволяють встановлювати дедлайни, вимірювати швидкість виконання задач або відстежувати завантаженість команди, але вони практично не враховують компетенції окремих працівників, специфіку їх стилю роботи чи залежність між особливостями задач та ефективністю їх виконання. Такі фактори, як різна швидкість реакції, досвід у певних типах завдань або схильність до перевантаження, не аналізуються системою глибоко. Унаслідок цього розподіл задач часто здійснюється за принципом черговості або рівномірного

навантаження, що не завжди призводить до оптимального результату. Інтелектуальні алгоритми могли б виявляти індивідуальні закономірності та забезпечувати персоналізований підхід до розподілу роботи, однак у більшості існуючих продуктів така функціональність перебуває на початковому рівні або зовсім відсутня.

Окремі сучасні системи впроваджують елементи штучного інтелекту, наприклад функції типу AI-assistant, які можуть формувати опис задачі, створювати структуру підзадач або проводити аналіз текстових вимог. Проте реальний автоматичний підбір виконавця, прогнозування строків виконання та адаптивне навчання на історії проєкту залишаються частково або повністю відсутніми.

Таким чином, аналіз ринку продемонстрував, що хоча існуючі платформи вирішують проблему організації роботи та відстеження прогресу, вони не забезпечують повноцінного автономного планування та оптимізації ресурсів на основі попереднього досвіду команди. Саме цей недолік став ключовою передумовою створення системи, яка поєднує класичні можливості систем управління проєктами з інтелектуальним модулем рекомендацій, здатним навчатися на реальних даних і підвищувати точність прогнозів у процесі експлуатації.

1.3. Методи прийняття рішень у проєктному менеджменті

Прийняття управлінських рішень у сфері IT-проєктів є складним і багатоетапним процесом, який включає оцінку ризиків, визначення пріоритетів, розподіл ресурсів, прогнозування строків виконання та контроль ключових показників ефективності. У сучасних умовах, коли команди працюють у швидкозмінному середовищі й використовують гнучкі підходи розробки, традиційні методи планування доповнюються або повністю замінюються автоматизованими інструментами підтримки прийняття рішень.

Існує декілька широко застосовуваних управлінських підходів, які вплинули на формування моделей планування в ІТ-галузі.

Першою групою є класичні детерміновані методи. До них належать мережеве планування, метод оцінки та аналізу програм, матричні таблиці відповідальності, а також принципи декомпозиції задач. Ці методи добре працюють у середовищі з передбачуваними процесами, де задачі мають чіткі залежності та фіксовані терміни. Проте в контексті ІТ-проектів, де вимоги та обсяг робіт можуть змінюватися, їх застосування часто вимагає постійного оновлення плану та особистого контролю менеджера.

Другою групою методів є підходи адаптивного управління, характерні для методологій Agile, Scrum та Kanban. У цьому випадку рішення приймаються на основі коротких циклів зворотного зв'язку: зборів команди, ретроспектив, оцінки спринтів і перегляду завантаження ресурсів. Прогнозування, як правило, здійснюється на основі velocity-метрик, burn-down діаграм та індикаторів пропускної здатності. Основна перевага цього підходу - його гнучкість, проте він продовжує покладатися на ручну аналітику, досвід РМ-менеджера або технічного лідера.

Третя група включає кількісні методи прийняття рішень, такі як багатофакторна оцінка, матриця вагових коефіцієнтів, аналіз сценаріїв та прогнозування на основі історичних трендів. Такі методи дозволяють об'єктивніше оцінити ризики, складність задач і рівень відповідності виконавців. Однак їх застосування зазвичай потребує значного обсягу даних та спеціалізованого програмного забезпечення.

Протягом останніх років формується четверта група - інтелектуальні автоматизовані системи прийняття рішень. Вони використовують машинне навчання, прогнозну аналітику та рекомендаційні алгоритми, що дозволяє системі не лише аналізувати минулі рішення, а й формувати нові на основі виявлених закономірностей. Такі системи поступово змінюють підходи управління, оскільки зменшують залежність від суб'єктивності людини,

скорочують час на розподіл завдань, підвищують точність прогнозів і сприяють стабільнішому плануванню.

Таким чином, еволюція управлінських методів у проєктному менеджменті проходить шлях від ручного планування до гібридних моделей, де ключову роль відіграють автоматизовані технології аналізу даних. Саме ця тенденція стала основою для створення системи, описаної у даній роботі, яка поєднує класичні управлінські принципи з адаптивною логікою машинного навчання.

Використання штучного інтелекту у сфері управління ІТ-проєктами вже давно перестало бути експериментом та поступово перетворюється на стандарт індустрії. Завдяки розвитку технологій машинного навчання, хмарної обробки даних та автоматизації бізнес-процесів ШІ починає виконувати не лише роль аналітичного інструмента, а й стає активним учасником процесу ухвалення рішень. У сучасних умовах він використовується для прогнозування строків виконання задач, оцінки завантаження ресурсів, оптимізації ризиків, виявлення залежностей між задачами та формування рекомендацій щодо планування.

Однією з ключових сфер застосування штучного інтелекту є автоматичне прогнозування дедлайнів. Традиційні методи розрахунку строків, що базуються на попередніх оцінках, середньому темпі виконання або експертній думці, часто можуть виявитися неточними через людський фактор або непередбачувані зміни в проєкті. ШІ використовує історичні дані, враховує реальні зміни навантаження працівників, їхню продуктивність, складність задач, пріоритети та інші фактори, щоб сформулювати більш реалістичний прогноз.

Іншим важливим напрямком застосування є рекомендаційні системи для розподілу задач. ШІ аналізує компетенції учасників команди, їх попередній досвід, швидкість виконання подібних задач, а також контекст проєкту, і пропонує найбільш оптимального виконавця. При цьому забезпечується баланс між справедливістю розподілу, ефективністю та швидкістю виконання. У таких системах реалізується адаптивний підхід, чим довше працює система, тим точнішими стають рекомендації.

Сучасні ІІІ-платформи також беруть участь у виявленні ризиків, аналізуючи затримки, переривання робочих процесів, перевищення навантаження та конфлікти ресурсів. Система здатна сформувати попередження або запропонувати альтернативне планування, перш ніж проблема проявиться критично.

Ще одним напрямом є інтелектуальна аналітика. ІІІ-модулі можуть обробляти великі обсяги даних, автоматично генерувати звіти, виявляти тренди, формувати прогнози продуктивності та пропонувати рекомендації для оптимізації процесів.

Варто зазначити, що інтеграція ІІІ у процес управління проєктами підтримує концепцію *proactive management*, коли система не просто фіксує стан проєкту, а активно впливає на процеси, допомагає передбачати проблеми і пропонує шляхи вирішення.

Планування ресурсів у проєктному менеджменті є однією з ключових задач, що потребує точного прогнозування, балансу між навантаженням та дедлайнами, а також оптимального розподілу робочого часу між членами команди. Існує низка математичних моделей, що застосовуються для вирішення подібних задач, і їх використання залежить від масштабу проєкту, кількості учасників, складності процесів та доступних даних.

Однією з найраніших і найпоширеніших моделей є лінійне програмування, яке дозволяє знайти оптимальне рішення шляхом математичного співставлення обмежень, наприклад: кількість доступних годин, дедлайни задач, кваліфікація виконавців. Такі моделі дозволяють мінімізувати час виконання або максимізувати ефективність використання ресурсів. Проте у сфері ІТ-розробки цей метод має обмеження - він погано адаптується до умов динамічних змін, неповних даних та різноманіття типів задач.

Інша група підходів ґрунтується на теорії графів, де задачі розглядаються як вузли, а залежності між ними - як ребра. Такі моделі застосовуються для створення критичних шляхів, оптимізації залежностей та визначення вузьких місць. Цей метод широко використовується в рамках підходу CPM/PERT, однак

він також не враховує поведінкові аспекти виконавців і можливі відхилення в реальному робочому процесі.

У міру розвитку проєктного менеджменту та зростання складності процесів виник попит на стохастичні моделі, які враховують випадковість, невизначеність та змінність. До таких моделей належать системи прогнозування на основі історичних даних, імовірнісні підходи до визначення строків виконання задач та аналіз сценаріїв. У сучасних інструментах ці методи стали основою для автоматичного аналізу ризиків.

Окрему групу складають методи евристичної оптимізації, зокрема генетичні алгоритми, алгоритми мурашиних колоній чи табуйованого пошуку. Вони застосовуються тоді, коли кількість можливих варіантів розподілу задач надто велика для прямого математичного моделювання. Такі алгоритми дозволяють наближено, але ефективно знаходити рішення у складних багатофакторних ситуаціях.

Однак найбільш перспективним напрямом у контексті сучасного управління ІТ-проєктами є гібридні моделі, що об'єднують класичні математичні методи з алгоритмами машинного навчання. Такий підхід дозволяє системі не лише оперувати існуючими обмеженнями, а й адаптуватися до нових даних, враховувати продуктивність виконавців, історичні відхилення та контекст проєкту.

У підсумку можна стверджувати, що математичні моделі планування й оптимізації ресурсів пройшли шлях від статичних формул і строгих обмежень до гнучких інтелектуальних систем. Ця еволюція стала основою для створення рішень, у яких ключова роль належить автоматизованому аналізу даних та адаптивному механізму прийняття рішень, що використовується у даній роботі.

1.4. Постановка задачі дослідження

На основі проведеного аналізу предметної області, існуючих інструментів управління проєктами, методів прийняття рішень і можливостей застосування штучного інтелекту в управлінні ресурсами стало очевидним, що сучасні системи планування мають суттєві обмеження у прогнозуванні, автоматичному розподіленні задач та адаптації до динамічних умов роботи команди.

Попри велику кількість існуючих платформ, жодна з них не реалізує повністю автоматизованого та адаптивного механізму призначення задач, який би враховував рівень кваліфікації виконавців, їхню поточну завантаженість, часові зони, наявність релевантних компетенцій, сторію виконання попередніх завдань, поведінкові та динамічні параметри ефективності.

Також відсутні системи, здатні автоматично пропонувати реалістичні дедлайни з урахуванням змінних показників продуктивності та непередбачуваності в командній роботі.

У зв'язку з цим була сформульована основна науково-практична задача дослідження - розробка інтелектуальної системи підтримки прийняття рішень для планування та оптимізації ресурсів ІТ-проєктів, яка здатна адаптивно призначати задачі виконавцям на основі даних та прогнозувати строки їхнього виконання.

Для досягнення мети необхідно було вирішити такі підзадачі:

1. Розробити структуру зберігання даних, що описує задачі, виконавців, їх навички та історію роботи.
2. Реалізувати математичну модель оцінки релевантності виконавця до задачі.
3. Створити механізм прогнозування дедлайнів на основі попередньої статистики.
4. Розробити модуль машинного навчання для адаптивного поліпшення рішення в процесі використання системи.

5. Забезпечити зручний інтерфейс для взаємодії користувача з рекомендаційним механізмом.

Таким чином, поставлена задача охоплює як теоретичні аспекти побудови інтелектуальних систем, так і практичну реалізацію програмного продукту для реального використання у сфері IT-менеджменту.

Висновки до розділу 1

У цьому розділі було проведено комплексний аналіз сучасних умов управління IT-проектами, ключових проблем ресурсного планування, а також огляд існуючих систем та методів, що застосовуються у сфері проектного менеджменту. Було встановлено, що попри значну кількість доступних інструментів, більшість із них не забезпечує гнучкої автоматизованої підтримки процесу розподілу задач і прогнозування дедлайнів з урахуванням індивідуальних характеристик виконавців.

Дослідження показало, що використання штучного інтелекту та методів машинного навчання у сфері управління проектами є перспективним напрямом, який дозволяє підвищити ефективність комунікацій, зменшити кількість помилок планування та оптимізувати використання людських ресурсів.

Виявлено, що існуючі системи переважно покладаються на ручні методи або статичні правила призначення, що робить їх недостатньо адаптивними для реальних бізнес-умов. На основі аналізу було сформульовано необхідність створення системи, здатної аналізувати дані про команду та задачі, адаптуватися до змін продуктивності, автоматично формувати рекомендації щодо виконавців, прогнозувати строки виконання на основі накопичених даних.

Ці висновки стали підґрунтям для постановки досліджуваної задачі та визначення подальших напрямів розробки інтелектуальної системи планування IT-проектів.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ

2.1. Архітектура та концептуальна модель системи

У процесі проєктування інтелектуальної системи підтримки прийняття рішень для планування та оптимізації ресурсів ІТ-проєктів було сформовано цілісну архітектурну модель, яка забезпечує узгоджену взаємодію всіх основних модулів, гарантуючи масштабованість, надійність та адаптивність системи до змін у проєктному середовищі.

Архітектура системи побудована за принципами багаторівневого розподілу логіки Multi-Layered System Architecture, що дозволяє розмежувати функціональність користувацького інтерфейсу, логіку обробки даних та механізми машинного аналізу. Виділено три ключові рівні: клієнтська частина, серверна логіка та інтелектуальний модуль рекомендацій[5].

Клієнтський рівень представлений веб-інтерфейсом, через який користувачі (керівники проєктів, розробники, аналітики) взаємодіють із системою. Інтерфейс забезпечує:

1. створення та редагування задач;
2. управління інформацією про виконавців;
3. перегляд навантаження команди;
4. зручні засоби моніторингу прогресу проєктів;
5. взаємодію з AI-модулем у вигляді рекомендацій.

Клієнтська частина побудована із використанням HTML5, CSS, Bootstrap, JavaScript та AJAX-запитів для асинхронної взаємодії без перезавантаження сторінки.

Серверна частина виконує основні функції обробки даних, валідації вхідних параметрів, керування бізнес-логікою, а також інтеграції з базою даних та AI-модулем.

Система реалізована на PHP із застосуванням RedBeanPHP ORM, що дозволяє гнучко керувати моделями, не прив'язуючись до жорсткої структури запитів.

Основні функції серверного рівня: керування проєктами, задачами та працівниками, ведення історії змін та логів, збереження прогнозів і рекомендацій AI, формування API-ендпоінтів для інтеграції інтелектуального модуля.

Цей рівень виступає посередником між користувачем та інтелектуальними алгоритмами прогнозування.

Структура даних розроблена таким чином, щоб підтримувати гнучку модель зберігання інформації про виконавців, їхні навички, задачі, проєкти та історію навантаження. Вона включає таблиці для зберігання стану задач, призначень, логів системи, рекомендацій та часових записів прогнозів.

Особливу роль відіграють таблиці історичних записів: вони використовуються як навчальний матеріал для удосконалення якості рекомендацій у майбутніх ітераціях використання системи.

AI-модуль відповідає за аналіз інформації з бази даних, визначення оптимального виконавця, прогнозування термінів виконання задач, формування рекомендацій для менеджера проєкту.

Модуль отримує дані через, обробляє їх, а потім повертає у вигляді структурованих відповідей. Інтелектуальний алгоритм використовує комбінацію правил, історичних метрик, рівня завантаженості, показників продуктивності та параметрів задачі.

Взаємодія між елементами побудована за принципом подієво-орієнтованої моделі, де кожна дія користувача запускає процес аналізу, логуювання або генерації рекомендацій.

Подана блок-схема (рис. 2.1) демонструє логічну структуру та процес взаємодії між ключовими компонентами інтелектуальної системи підтримки прийняття рішень при управлінні IT-проєктами.

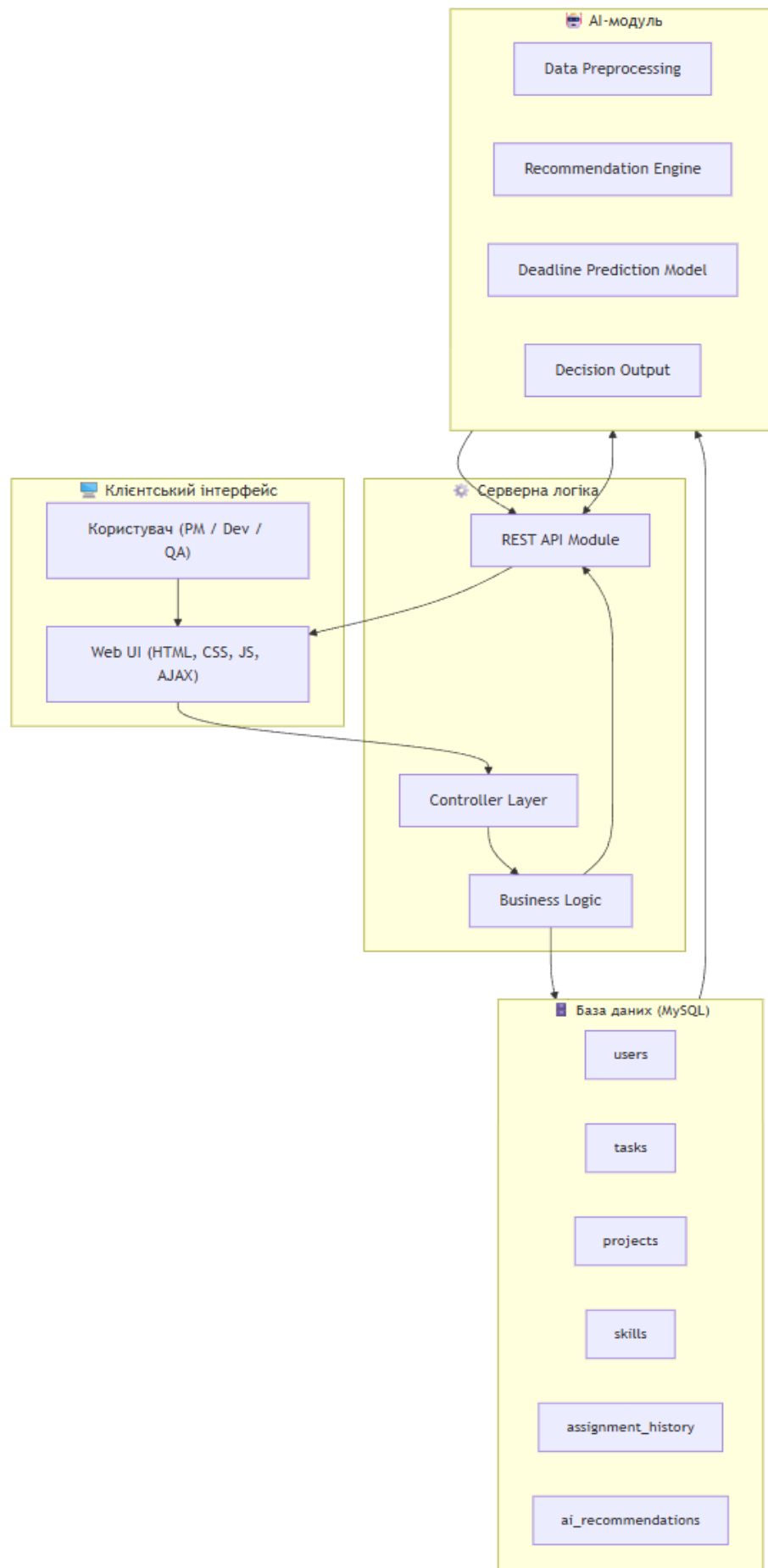


Рисунок 2.1 – Архітектура системи

Архітектура побудована за принципами багаторівневого підходу, де кожен рівень виконує окрему функціональну роль, але взаємодіє з іншими через стандартизовані інтерфейси.

Запропонована архітектура дозволяє забезпечити:

- модульність структури;
- легкість розширення функціональності;
- можливість подальшої інтеграції більш складних алгоритмів машинного навчання;
- стійкість до масштабування та збільшення обсягів даних.

Таким чином, архітектурна модель є фундаментом, що забезпечує функціональність системи, оптимізує роботу користувачів та створює основу для інтелектуального аналізу даних.

2.2. Вибір моделі машинного навчання та методів обробки даних

Під час проектування інтелектуального модуля системи постала необхідність вибрати такий підхід до машинного навчання, який можна інтегрувати у веб-середовище, побудоване на PHP, та який буде придатний для задач прогнозування строків виконання робіт і рекомендацій виконавців. Оскільки система працює зі структурованими даними (оцінка навичок, складність задач, часові показники, пріоритети, рівень завантаженості), було вирішено, що технологія повинна ефективно працювати з табличною інформацією та змішаними типами даних [6].

Після аналізу доступних рішень було визначено, що найраціональнішим є використання бібліотеки PHP-ML, оскільки вона є стабільною, достатньо зрілою, добре документованою та інтегрується у наявний технологічний стек без необхідності використання зовнішніх мов або окремих серверів машинного навчання.

На етапі вибору алгоритму було розглянуто кілька класів моделей, алгоритми класифікації, алгоритми регресії, алгоритми рекомендаційного типу.

З огляду на характер даних, їхню динаміку та вимоги системи, було вирішено застосувати метод Random Forest. Цей підхід є одним із найбезпечніших і найнадійніших у класичних машинних моделях, оскільки він добре працює з різнорідними параметрами, стійкий до статистичного шуму та не потребує складної попередньої підготовки даних. Random Forest дозволяє враховувати одночасно кілька факторів, таких як рівень спеціаліста, минула швидкість виконання задач, завантаженість, тип задачі, бізнес-важливість та інші параметри [7].

Перед використанням моделі було проведено попередню обробку даних. Вона включала очищення від пропусків і невалідних значень, нормалізацію числових полів, кодування категоріальних ознак та перетворення наборів навичок у цифрове подання, яке можна використовувати у навчанні моделі. Такий підхід дав змогу привести дані у єдиний формат, придатний для аналізу.

Важливою вимогою було те, що модель повинна вдосконалюватися в процесі експлуатації системи. Обраний метод дозволяє поетапно оновлювати знання моделі на основі нових даних, а саме - завершених задач, нових метрик продуктивності та змін у завантаженості команди. Це забезпечує адаптивність та поступове підвищення точності прогнозів.

Таким чином, вибір моделі Random Forest у поєднанні з бібліотекою RHP-ML виявився оптимальним. Це рішення забезпечує баланс між точністю, надійністю, масштабованістю та технічною реалізацією в обраному середовищі. Обрана технологія також передбачає подальший розвиток, оскільки в разі накопичення великого масиву даних система може бути розширена шляхом винесення моделювання на зовнішні сервіси або розширення інструментарію без зміни логіки системи.

Використаний у системі метод Random Forest працює як колективна модель, що складається з множини незалежних дерев рішень. Кожне дерево аналізує дані та формує власний прогноз, а кінцевий результат обчислюється як середнє значення або найбільш часто повторюваний варіант. Такий підхід робить модель стійкою до помилок, викликаних окремими випадками або

шумовими даними, і дозволяє уникнути переобучення, характерного для класичних дерев рішень [8].

У контексті розробленої системи модель отримує на вхід набір характеристик задачі та виконавців. До таких характеристик належать рівень навичок, пріоритет задачі, бізнес-важливість, орієнтовна тривалість, історична швидкість виконання, рівень зайнятості та інші параметри. На основі цих даних модель формує два основних результати: ймовірність вдалого виконання задачі певним працівником, орієнтовний прогноз часу завершення задачі в робочих годинах чи днях. Процес прийняття рішення відбувається у кілька етапів.

Спочатку модель аналізує кожен окремий параметр: складність задачі, наявність релевантних навичок, рівень продуктивності виконавця, а також його поточне навантаження. Далі кожне дерево у моделі формує власний прогноз. Оскільки система складається з великої кількості дерев, слабкі або випадкові рішення окремих дерев мають мінімальний вплив на загальний результат. Такий підхід дозволяє моделі враховувати складні кореляції та взаємозв'язки, які не завжди очевидні в логічних або ручних системах планування [9].

Після отримання результатів модель ранжує потенційних виконавців за рівнем відповідності задачі. Користувач отримує список із найкращими кандидатами, де перший у списку є найбільш оптимальним з точки зору алгоритму. Разом із цим модель також визначає прогнозований час виконання, що дозволяє системі формувати більш реалістичні дедлайни.

Завдяки здатності моделі вдосконалюватися на основі нових завершених задач її прогнози з часом стають точнішими. Такий механізм забезпечує адаптивність системи до реальних умов роботи команди, зміни навичок працівників, сезонних коливань навантаження та нових типів задач [10].

2.3. Проєктування бази даних

Проєктування бази даних є ключовим етапом у реалізації інтелектуальної системи підтримки прийняття рішень, оскільки саме структура даних визначає

якість, масштабованість та ефективність роботи всієї системи. Модель даних була розроблена відповідно до принципів нормалізованого реляційного зберігання інформації, з чітким розділенням сутностей, мінімізацією дублювання та забезпеченням логічних зв'язків між об'єктами.

Основна мета проєктування БД - забезпечити можливість зберігання інформації про користувачів системи та їх навички, проєктів та задач з повними атрибутами, необхідних компетенцій для виконання задач, історії призначень задач на працівників, прогнозів, рекомендацій та активностей системи штучного інтелекту, історії завантаження працівників для можливості подальшого аналітичного аналізу (рис. 2.2).

При проєктуванні використовувалась доменно-орієнтована модель, де кожна сутність відповідає реальному об'єкту системи управління. Для забезпечення інтегрованості й простоти масштабування було виділено окремі таблиці для довідників та таблиці взаємодії [11].

Всі сутності пов'язані між собою за допомогою первинних ключів та зовнішніх ключів, що дозволяє забезпечити референційну цілісність, спростити пошук виконавців для задач за компетенціями, формувати історичні вибірки для прогнозування та навчання AI-модуля.

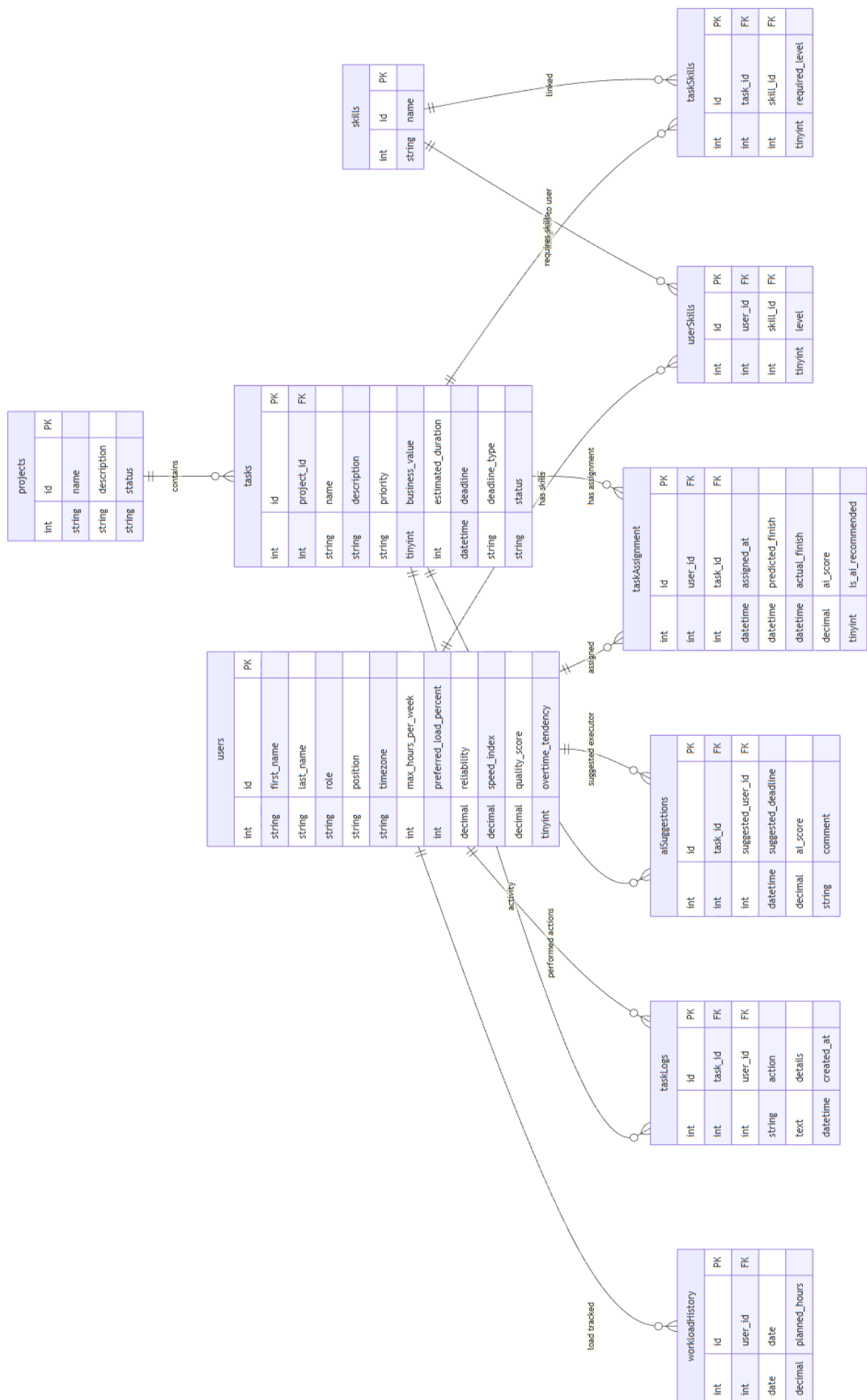


Рисунок 2.2 – ER-Діаграма бази даних

2.4. Алгоритм інтелектуального призначення задач

Алгоритм інтелектуального призначення задач у розробленій системі виконує роль автоматизованого інструменту підтримки прийняття рішень, який допомагає менеджеру проєкту швидко та ефективно визначити, кому саме варто доручити виконання певної задачі. Основна ідея полягає в тому, щоб мінімізувати ручний аналіз численних факторів та забезпечити більш об'єктивний, прогнозований та прозорий процес розподілу роботи всередині команди [12].

Процес роботи алгоритму починається одразу після створення або редагування задачі. Система аналізує ключові параметри: складність, бізнес-важливість, тип дедлайну, пріоритет, а також необхідні навички для виконання. На основі цих даних формується профіль задачі - структурований опис того, яким повинен бути оптимальний виконавець [13].

Після визначення характеристик самої задачі система переходить до аналізу доступних працівників. На цьому етапі враховуються кілька ключових аспектів. Першим фактором є відповідність компетенцій виконавця вимогам задачі. Якщо працівник не володіє ключовою навичкою або має значно нижчий рівень кваліфікації, система не розглядає його як потенційного кандидата [14].

Другим важливим параметром є поточне навантаження працівника. Система перевіряє, скільки годин занесено в його робочий графік, чи наближається він до максимально допустимого обсягу роботи, та чи має вільні часові інтервали для виконання додаткової задачі. Це дозволяє уникнути ситуацій, коли відповідальний, але перевантажений співробітник отримує ще одне завдання, що може негативно вплинути на строки виконання інших задач.

Третім аспектом виступають історичні показники продуктивності. Система використовує такі дані, як швидкість виконання попередніх задач, стабільність роботи, якість результатів та реакція виконавця на критичні зміни. Це дозволяє алгоритму не лише оцінювати формальні компетенції, а й враховувати реальну ефективність у контексті робочого середовища [15].

На основі сукупності цих факторів система формує список кандидатів та визначає найбільш оптимального. На відміну від фіксованих правил, алгоритм працює як гнучка модель, що може адаптуватися до змін у навантаженні, складі команди або пріоритетах проєкту. Якщо система виявляє однаково підходящих кандидатів, вона може запропонувати кілька варіантів або рекомендації на основі сценаріїв.

Особливістю алгоритму є також можливість врахування дедлайну. Якщо задача має фіксований і жорсткий строк виконання, система віддає перевагу тим працівникам, які потенційно зможуть завершити її вчасно. Якщо ж дедлайн гнучкий, алгоритм може рекомендувати більш рівномірний розподіл навантаження між співробітниками [16].

У результаті користувач отримує не лише автоматичне призначення, але й пояснення, чому саме цей працівник був рекомендований системою. Такий підхід підвищує довіру до автоматизованого планування, а також створює зручність для аналітики управлінських рішень.

Блок-схема (рис. 2.3) відображає логічну послідовність роботи інтелектуального алгоритму призначення задач. Процес починається з моменту створення або зміни задачі. Спочатку система аналізує її ключові параметри: пріоритет, дедлайн, тривалість та список необхідних компетенцій. На основі цього формується вимоговий профіль задачі [17].

Далі система перевіряє, чи існують працівники, які відповідають встановленим критеріям. Якщо таких виконавців немає, процес завершується повідомленням про відсутність кандидатів. Якщо потенційні виконавці знайдені, система переходить до аналізу їхнього поточного робочого навантаження, історичної ефективності та якості виконаних задач.

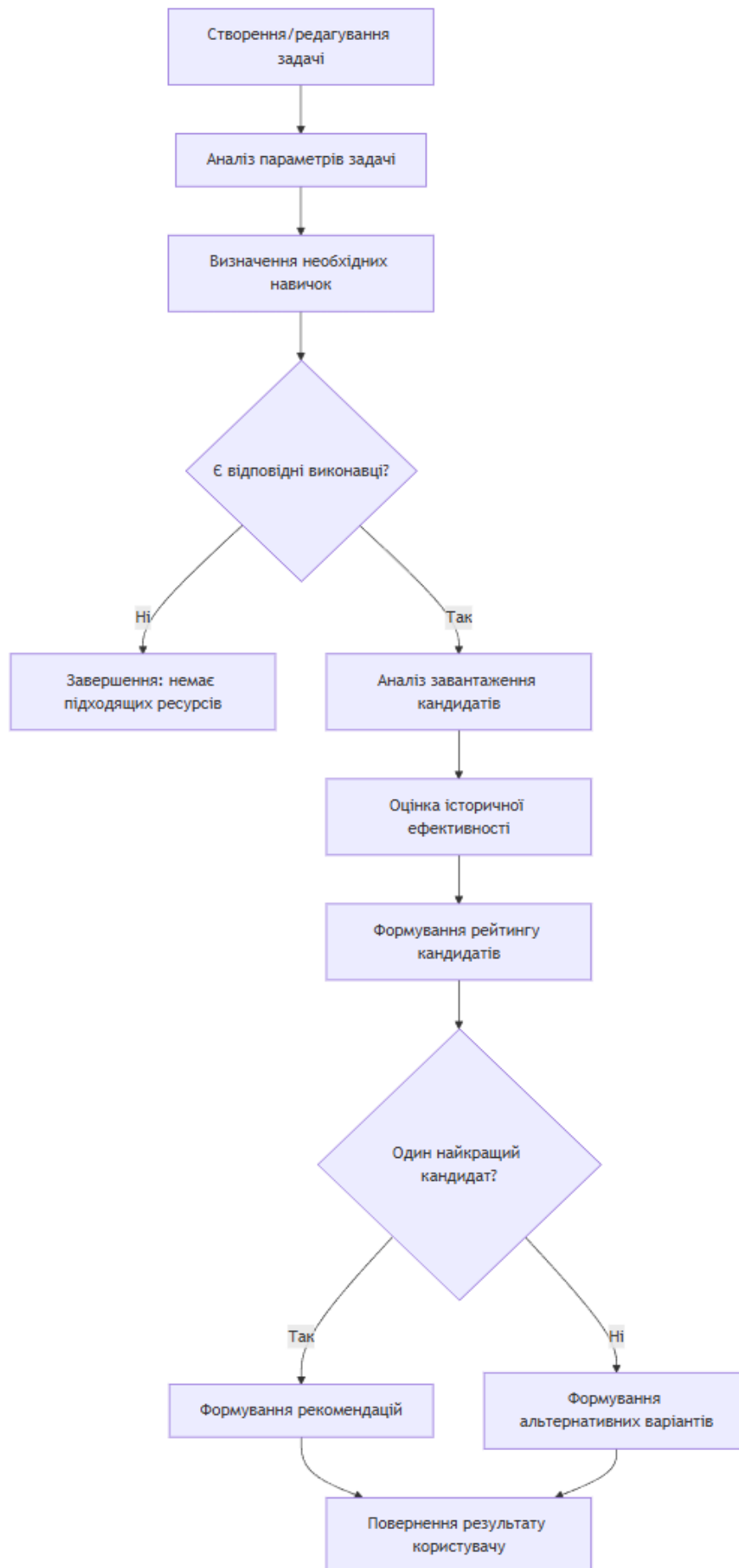


Рисунок 2.3 – Алгоритм призначення задач

Після обробки цих даних система формує рейтинг кандидатів. Якщо є один очевидно найкращий виконавець - він пропонується користувачу. Якщо можливі декілька близьких за ефективністю варіантів, система формує список рекомендацій, залишаючи користувачу можливість прийняти остаточне рішення.

Модуль прогнозування дедлайнів є важливим компонентом системи, оскільки він дозволяє автоматично визначати реалістичні строки виконання задач на основі історичних даних, попередньої продуктивності працівників, рівня складності задач та їх пріоритету. Такий підхід зменшує ризик помилок під час ручного планування та дозволяє адаптуватися до змін у навантаженні команди або динаміки проєкту [18].

На практиці задачі рідко виконуються в однакових умовах, і попередня статистика показує, що час, встановлений вручну, часто не відповідає реальним ресурсам виконавця або складності роботи. Тому модуль прогнозування дедлайнів був створений як автоматизований механізм, який допомагає керівнику проєкту отримати орієнтовну дату завершення задачі перед її призначенням.

Робота модуля базується на аналізі кількох груп даних. Спочатку система враховує заплановану тривалість задачі, яку вказав користувач під час створення. Потім алгоритм перевіряє поточне завантаження співробітника, його доступні робочі години на найближчі дні та звичний темп роботи. Якщо попередні задачі працівник виконував швидше або повільніше за прогнозований час, система коригує оцінку, адаптуючись до індивідуальних особливостей співробітника.

Наступним етапом є врахування пріоритетності задачі. Якщо робота має критичний статус або високу бізнес-цінність, алгоритм намагається зменшити розрив між поточними подіями у графіку та початком виконання задачі, вважаючи її більш терміною. Для низькопріоритетних завдань система, навпаки, розподіляє прогнозований обсяг у більш розслабленому темпі, зменшуючи навантаження у пікові періоди.

Окрему роль відіграє аналіз попередніх рекомендацій AI та історичної

статистики успішних або невдалих дедлайнів. Якщо система фіксує patterns, наприклад певний тип задач завжди потребує більше часу - прогноз автоматично підлаштовується.

Після завершення усіх етапів обчислення модуль повертає орієнтовну дату завершення задачі з текстовим поясненням. Це пояснення дозволяє користувачу зрозуміти, чому саме система прийняла таке рішення, що підвищує прозорість процесу планування та довіру до інтелектуальних функцій системи.

Блок-схема (рис. 2.4) демонструє послідовність роботи модуля прогнозування дедлайнів. Спочатку система отримує дані задачі, включаючи очікувану тривалість. Потім проводиться аналіз історичних результатів виконавця, його актуального навантаження та важливості задачі. Після цього система перевіряє, чи потрібно скоригувати попередній прогноз відповідно до статистики виконання подібних задач у минулому.

Якщо корекція потрібна - система оновлює прогноз. На завершальному етапі користувач отримує орієнтовну дату завершення задачі разом із коментарем, який пояснює логіку прийнятого рішення.

Оскільки система працює в умовах динамічного середовища - де виконання задач, зміна навичок, підвищення кваліфікації працівників та коливання навантаження відбуваються постійно - важливо, щоб модель машинного навчання не залишалася статичною. Тому в архітектурі системи передбачено механізм поступового накопичення нових даних і періодичного донавчання моделі.

Механізм працює автоматично і базується на тому, що кожен завершений робочий цикл створює нову точку даних у системі. Ці дані містять інформацію про те, наскільки прогноз, згенерований моделлю, збігся з реальним результатом. Основною перевагою такого підходу є те, що система не потребує повторного повного навчання моделі з нуля - достатньо додавати нові записи до існуючої бази та періодично оновлювати модель, що суттєво зменшує обчислювальне навантаження.

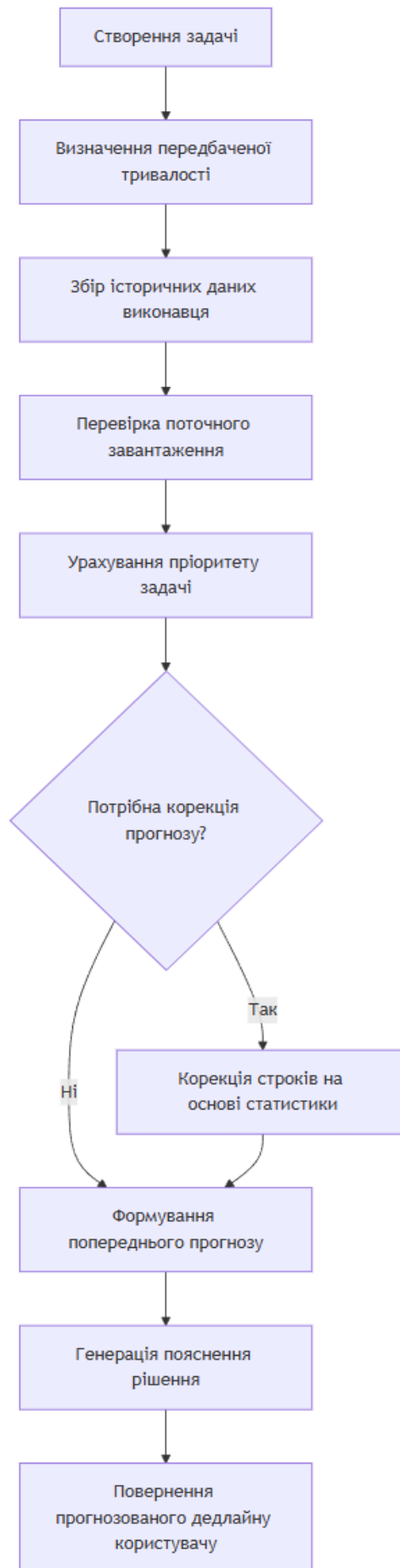


Рисунок 2.4 – алгоритм роботи

Під час повторного навчання модель аналізує як старі, так і нові записи, що дає можливість адаптуватися до змін у поведінці команди. Якщо один із працівників суттєво підвищив продуктивність або, навпаки, почав затримувати дедлайни, модель почне враховувати цю зміну й коригувати майбутні прогнози.

Також у системі передбачено механізм контролю якості моделі. Перед тим як нова версія моделі буде використана в реальній роботі, вона порівнюється зі старою. Якщо нова версія демонструє кращу точність, вона замінює попередню. У протилежному випадку система продовжує використовувати стару модель, а процес навчання повторюється у майбутньому.

Таким чином, модель у системі функціонує як самонавчальний компонент, (рис. 2.5) який стає точнішим у процесі реального використання, не вимагає ручного втручання та продовжує вдосконалювати рекомендації й прогнозування дедлайнів відповідно до накопичених даних.

AI-модуль у системі виконує роль інтелектуального помічника, який допомагає керувати проєктами, прогнозувати строки виконання задач та рекомендувати найбільш відповідних виконавців. Його робота побудована таким чином, щоб користувач не мав справи зі складною технічною логікою, а отримував результат у максимально зрозумілому вигляді.

Взаємодія між модулем і користувачем відбувається у кілька послідовних етапів. Спочатку користувач створює або редагує задачу, задаючи її параметри: пріоритет, оцінку складності, тривалість, необхідні навички та інші атрибути. На цьому етапі AI-модуль ще не втручається - система лише фіксує вихідні дані.

Після цього користувач має можливість на вимогу запустити AI-аналіз. Запуск здійснюється натисканням відповідної кнопки у формі задачі, тому процес є інтуїтивно зрозумілим і не вимагає спеціальних знань. У момент виклику AI-модуль отримує всі введені користувачем параметри і доповнює їх поточними даними з системи - наприклад, рівнем навантаженості виконавців, їхньою продуктивністю та попередніми результатами виконання схожих задач.

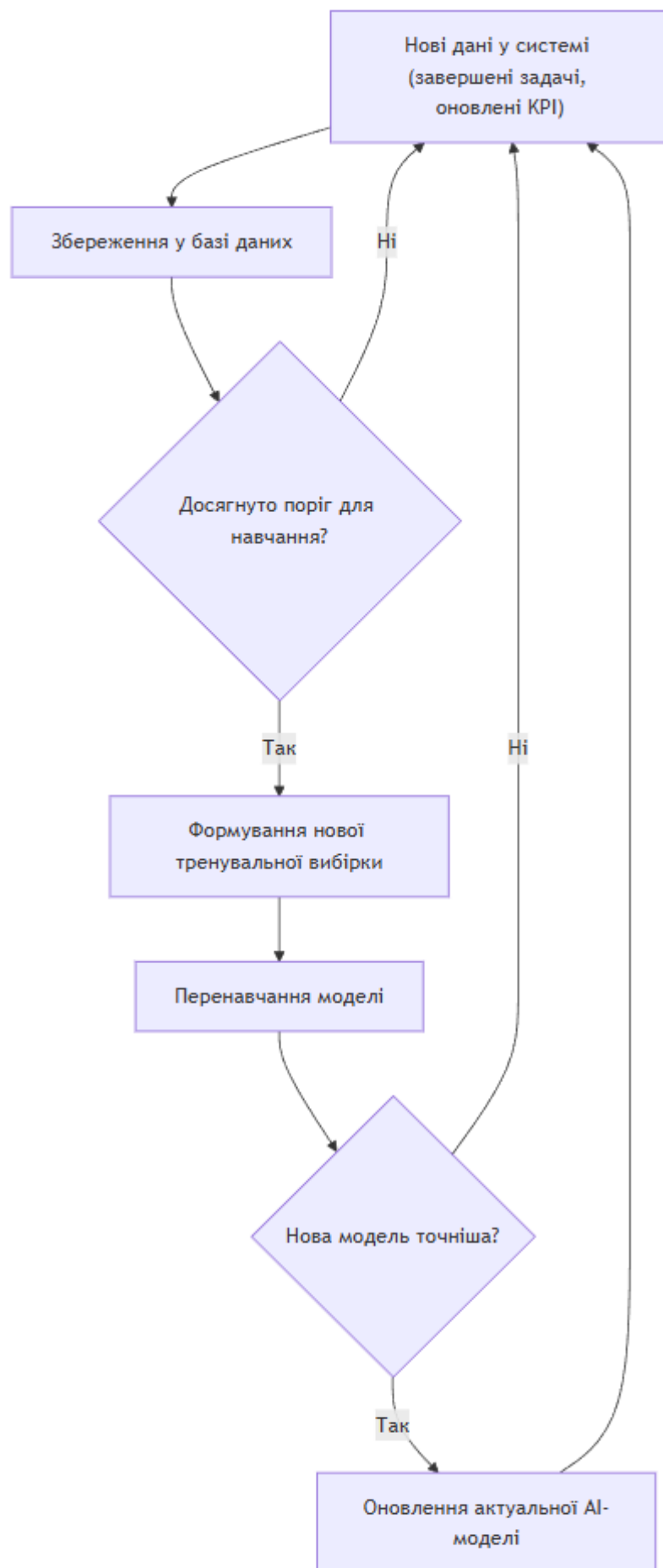


Рисунок 2.5 – Механізм донавчання

Після отримання даних AI-модуль виконує прогнозування. Модель генерує два основних результати, оцінку найбільш імовірного виконавця і прогноз строку завершення задачі.

Якщо система не може впевнено визначити одного кандидата, вона повертає декілька варіантів з рейтингом відповідності.

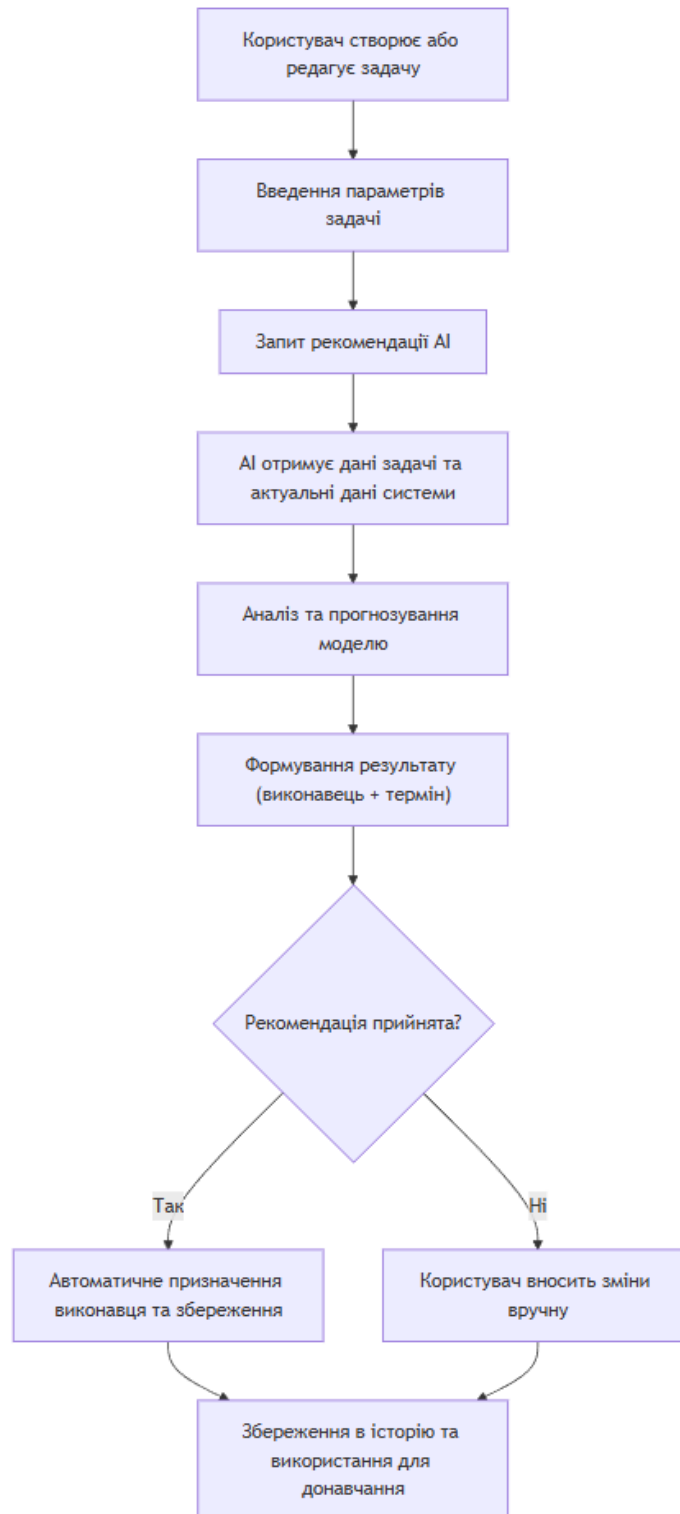


Рисунок 2.6 – Алгоритм взаємодії

Кінцевий результат подається користувачу у компактному та інформативному вигляді: ім'я рекомендованого виконавця, значення прогнозу, рівень впевненості системи та коротке пояснення, чому був запропонований саме цей варіант. Це дозволяє користувачу не лише отримати готовий результат, але й розуміти логіку рекомендації [19].

Усі взаємодії з AI-модулем автоматично фіксуються в системі. Це дозволяє проводити подальший аналіз, оцінювати точність рекомендацій і навчати модель на основі реальних рішень користувачів.

Таким чином, взаємодія між користувачем і AI-модулем (рис. 2.6) побудована так, щоб не замінювати людину, а підтримувати її рішення, роблячи процес планування швидшим, точнішим і менш залежним від людського фактора.

Висновки до розділу 2

У другому розділі було проведено проєктування ключових компонентів інтелектуальної системи підтримки прийняття рішень для планування та оптимізації ресурсів IT-проєктів. На основі аналізу предметної області було сформовано концептуальну архітектуру рішення, що включає серверну частину, клієнтський інтерфейс, базу даних, інтелектуальний модуль рекомендацій та механізми автоматичного аналізу даних.

У процесі проєктування було розроблено структурну модель бази даних, яка забезпечує зберігання усіх необхідних компонентів для роботи системи: даних про працівників, задачі, навички, робоче навантаження, зв'язки між елементами, а також історичні записи для подальшого навчання моделі та формування рекомендацій. Структура побудована таким чином, щоб забезпечити масштабованість, стійкість та можливість подальшого розширення, включаючи додавання нових категорій даних або модулів.

Окрему увагу було приділено вибору моделі машинного навчання. З урахуванням характеру даних, технічних обмежень і вимог до інтеграції було обрано підхід на базі бібліотеки PHP-ML, який дозволяє реалізувати прогнозування строків виконання задач і рекомендацію відповідних виконавців без зовнішніх сервісів. Алгоритм Random Forest був визначений як оптимальний завдяки стійкості до шуму, здатності працювати зі змішаними типами ознак та можливості навчання на кумулятивних даних.

Також було описано механізм автоматичного накопичення даних і періодичного донавчання моделі, що забезпечує адаптивність системи та підвищення точності прогнозів у процесі реальної експлуатації. Алгоритми взаємодії з користувачем побудовано відповідно до принципів юзабіліті: система не замінює людину у прийнятті рішень, але оптимізує процес планування й забезпечує обґрунтовані рекомендації.

Таким чином, результати проектування свідчать, що розроблена модель системи є логічно узгодженою, технічно здійсненою та готовою до реалізації. Вона забезпечує основу для подальшого впровадження, тестування та інтеграції інтелектуальних інструментів, що підвищують ефективність управління ІТ-проектами.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

3.1. Реалізація серверної частини системи

У рамках створення інтелектуальної системи підтримки прийняття рішень була реалізована серверна частина, яка забезпечує логіку збереження даних, виконання бізнес-процесів, обробку запитів користувача та взаємодію з модулем штучного інтелекту. Серверна частина виступає центральною ланкою системи, оскільки саме вона відповідає за узгодженість даних, коректність операцій і сервісну логіку, необхідну для функціонування всіх підсистем.

Розробка серверного функціоналу здійснювалася з використанням архітектурної моделі MVC, що дозволило відокремити відображення інтерфейсу від логіки обробки даних. Такий підхід забезпечив чисту структуру проєкту, можливість подальшого масштабування та зручність підтримки [20].

Одним із ключових етапів було створення системи маршрутизації, яка визначає, який контролер і функція відповідають за обробку конкретного запиту. Завдяки цьому структуру проєкту вдалося зробити логічною: кожен функціональний модуль отримав власний контролер і набір дій.

Під час реалізації були визначені основні доменні сутності системи:

- працівники;
- навички;
- задачі;
- призначення виконавців;
- історія навантаження;
- проєкти;
- лог подій;
- AI-рекомендацій;

Для роботи з цими даними використовувався ORM-підхід, що дозволив працювати з базою даних на рівні об'єктів. Такий метод усунув необхідність

вручну писати SQL-запити для базових операцій, що позитивно вплинуло на швидкість розробки.

Одним із важливих етапів стала реалізація функціоналу управління працівниками. Система дозволяє створювати, редагувати та переглядати їх характеристики, включно з рівнем навичок, графіком роботи, індивідуальними показниками продуктивності та історією призначених задач. Усі ці параметри використовуються AI-модулем під час формування рекомендацій [21].

Окрему увагу було приділено модулю керування задачами. При створенні задачі користувач може вказувати її характеристики: пріоритет, тип дедлайну, важливість для бізнесу, необхідні компетенції тощо. Серверна частина забезпечує перевірку коректності введених даних, запис інформації до бази даних, фіксацію змін та можливість прив'язувати задачі до конкретних працівників.

Особливим блоком стала реалізація механізмів автоматичного збору аналітики. Система зберігає історію призначень задач, відстежує очікувані і фактичні дедлайни, а також формує статистику завантаженості працівників на основі розрахованих і зафіксованих трудових витрат. Це дозволило побудувати базовий фундамент для функціонування прогнозних моделей.

Важливим компонентом серверної частини є взаємодія з модулем штучного інтелекту. Було реалізовано окремий процес обробки запитів від інтерфейсу користувача, де система враховує параметри задачі та історичні дані, після чого формує рекомендацію щодо виконавця та прогнозує терміни завершення. Такий підхід дозволив перевести ухвалення рішень із суто ручного режиму у автоматизований, заснований на даних [22].

Завдяки комплексній реалізації серверної частини система отримала можливість:

- централізовано керувати бізнес-логікою;
- забезпечувати коректність операцій із даними;
- гарантувати масштабованість та розширюваність функціоналу;
- підтримувати двосторонню інтеграцію з AI-модулем;

- створювати й підтримувати повний цикл управління задачами, проєктами та ресурсами.

У додатку реалізовано простий механізм маршрутизації HTTP-запитів (рис. 3.1), що зіставляє URL-шлях із відповідним контролером та його екшеном. Для основних сутностей системи (головна панель, проєкти, задачі, працівники, AI-модуль) налаштовані окремі маршрути.

```
return [
    ['GET', '/', 'main/index'],
    ['GET', '/main/index', 'main/index'],

    // Проєкти
    ['GET', '/main/list', 'main/listProjects'],
    ['GET', '/main/projectView', 'main/projectView'],
    ['POST', '/main/projectStatus', 'main/projectStatus'],

    // Задачі
    ['GET', '/main/add', 'main/addTask'],
    ['POST', '/main/add', 'main/addTask'],
    ['GET', '/main/tasksView', 'main/tasksView'],
    ['POST', '/main/tasksAssig', 'main/tasksAssig'],

    // Працівники
    ['GET', '/main/workers', 'main/listWorkers'],
    ['GET', '/main/create', 'main/create'],
    ['POST', '/main/create', 'main/create'],
    ['GET', '/main/workerView', 'main/workerView'],
    ['POST', '/main/workerUpdate', 'main/workerUpdate'],

    // AI модуль
    ['GET', '/main/ai', 'main/ai'],
    ['POST', '/ai/suggest-user', 'ai/suggestUser'],
    ['POST', '/ai/suggest-deadline', 'ai/suggestDeadline'],
];
```

Рисунок 3.1 – Маршрутизація в додатку

Кожен запис визначає HTTP-метод, URL-шлях та пару контролер і екшен. Наприклад, запит GET /main/workers буде оброблено методом listWorkersAction() класу MainController.

Контролери згруповані у просторі імен controllers і успадковують базовий клас (рис. 3.2), який забезпечує механізми встановлення змінних для подальшого відображення у view [23].

```

namespace controllers;

use R;

class MainController extends ApplicationController
{
}

```

Рисунок 3.2 – Наслідування контролерів

Центральною точкою входу для користувача є головна сторінка системи, на якій відображаються ключові показники, завантаженість працівників, задачі з очікуваним завершенням сьогодні, а також останні рекомендації AI. Уся ця інформація готується в екшені `indexAction()` (рис. 3.3).

```

public function indexAction(){

    $projects = \R::findAll('projects', ' ORDER BY created_at DESC ');

    $tasks = \R::findAll('tasks', ' status != ? ORDER BY created_at DESC LIMIT 20 ', ['done']);

    $users = \R::findAll('users', ' ORDER BY last_name ASC ');

    // Історія завантаження за останні 14 днів
    $workload = \R::getAll("
        SELECT user_id, date, planned_hours
        FROM workload_history
        WHERE date >= DATE_SUB(CURDATE(), INTERVAL 14 DAY)
        ORDER BY date ASC
    ");

    // KPI по задачах
    $totalTasks = \R::count('tasks');
    $activeTasks = \R::count('tasks', ' status = ? ', ['in_progress']);
    $criticalTasks = \R::count('tasks', ' priority = ? ', ['critical']);
    $doneTasks = \R::count('tasks', ' status = ? ', ['done']);

    $kpi = [
        'total_tasks' => $totalTasks,
        'active_tasks' => $activeTasks,
        'critical_tasks' => $criticalTasks,
        'done_tasks' => $doneTasks,
    ];

    // Останні рекомендації AI
    $aiSuggestions = \R::findAll('ai_suggestions', ' ORDER BY created_at DESC LIMIT 5 ');

    // Завантаженість працівників за 7 днів
    $userLoad = \R::getAll("
        SELECT u.id, u.first_name, u.last_name,
               COALESCE(SUM(w.planned_hours), 0) AS hours_total
        FROM users u
        LEFT JOIN workload_history w ON u.id = w.user_id
        AND w.date >= DATE_SUB(CURDATE(), INTERVAL 7 DAY)
        GROUP BY u.id
        ORDER BY u.last_name ASC
    ");
}

```

Рисунок 3.3 – Початковий екшен

Окремий модуль відповідає за створення, перегляд списку та редагування працівників. Це важливий елемент системи, оскільки всі подальші AI-

рекомендації базуються на параметрах виконавців: ролі, робочому графіку, продуктивності та якості. Список працівників формується в екшені `listWorkersAction()` (рис. 3.4).

```
public function createAction()
{
    // якщо POST — створюємо робітника
    if (!empty($_POST)) {

        $first_name = trim($_POST['first_name'] ?? '');
        $last_name = trim($_POST['last_name'] ?? '');

        if ($first_name === '' || $last_name === '') {
            $_SESSION['error'] = "Ім'я та прізвище є обов'язковими.";
            redirect('/main/create');
        }

        $u = \R::dispense('users');

        $u->first_name = $first_name;
        $u->last_name = $last_name;

        $u->role = $_POST['role'] ?? 'Junior';
        $u->position = $_POST['position'] ?? '';
        $u->timezone = $_POST['timezone'] ?? 'UTC+2';

        $u->work_hours_start = $_POST['work_hours_start'] ?? '10:00:00';
        $u->work_hours_end = $_POST['work_hours_end'] ?? '19:00:00';

        $u->max_hours_per_week = (int)($_POST['max_hours_per_week'] ?? 40);
        $u->preferred_load_percent = (int)($_POST['preferred_load_percent'] ?? 80);

        $u->reliability = (float)($_POST['reliability'] ?? 1.00);
        $u->speed_index = (float)($_POST['speed_index'] ?? 1.00);
        $u->quality_score = (float)($_POST['quality_score'] ?? 4.50);
        $u->overtime_tendency = (int)($_POST['overtime_tendency'] ?? 0);

        \R::store($u);

        $_SESSION['success'] = "Працівника успішно створено!";
        redirect('/main/workers');
    }

    // GET — просто показати форму
    $this->set([]);
}
```

Рисунок 3.4 – Лістинг модуля працівників

Серверна частина системи:

- реалізує повний набір CRUD-операцій для основних сутностей;
- побудована на основі MVC-архітектури, що забезпечує

структурованість коду;

- використовує ORM RedBeanPHP, що спрощує роботу з базою даних та пришвидшує розробку;
- містить спеціалізований AI-модуль, який доступний як через внутрішні сервіси, так і через зовнішні REST-ендпоїнти для клієнтської частини;
- забезпечує логування ключових дій, що є основою для подальшого аналізу та навчання моделей.

Таким чином, реалізація серверної частини стала фундаментом усієї системи і забезпечила її працездатність, керованість та можливість подальшого розвитку у напрямі глибшої автоматизації та застосування передбачувальних алгоритмів.

3.2. Реалізація клієнтської частини та інтерфейсу користувача

Під час розробки клієнтської частини інтелектуальної системи основним завданням було створення інтерфейсу, який забезпечує зручність взаємодії з даними, простоту навігації та наочне відображення важливої інформації - завдань, ресурсів, дедлайнів і рекомендацій штучного інтелекту. Інтерфейс мав бути інтуїтивно зрозумілим, навіть для користувача без технічної підготовки, а також адаптивним, щоб система могла коректно працювати на різних типах пристроїв [24].

На етапі проєктування було визначено ключовий принцип: користувач повинен мінімально вводити інформацію вручну - система має пропонувати, підказувати або автоматично заповнювати значну частину даних на основі аналізу попередніх дій, історії завдань та оцінок ефективності працівників.

Для розробки інтерфейсу використовувався підхід компонентної побудови. Було створено окремі повторно використовувані елементи: картки задач, таблиці зі статусами, інтерактивні списки працівників, календарні блоки, прогрес-бари завантаженості, а також спеціальні індикатори для позначення

важливості, критичності чи прострочених задач.

Першим елементом, який було реалізовано, стала головна інформаційна панель. Саме вона дозволяє користувачу швидко оцінити поточний стан проєктів та ресурсів. На дашборді відображаються: загальна кількість задач, кількість критичних задач, завершені задачі, а також інформація про завантаженість працівників. Цей блок був доповнений інтерактивним таймлайном, який дозволяє візуально оцінити, коли саме планується завершення кожного завдання.

Наступним ключовим елементом стала система форм введення даних. Було розроблено форми створення задач, працівників та проєктів, які містять структурований набір полів із логічною групацією: загальні відомості, навички, параметри дедлайну, показники продуктивності тощо. На кожному етапі реалізації форм робився акцент на автоматизацію: частина полів заповнюється за замовчуванням, деякі - формуються залежно від попереднього вибору.

Особливу увагу було приділено взаємодії користувача з модулем штучного інтелекту. У формі створення задачі розміщено спеціальні кнопки: «Запропонувати виконавця» і «Запропонувати дедлайн». Після натискання система без перезавантаження сторінки надсилає запит до серверної частини, отримує рекомендацію, і відображає її безпосередньо у формі. Такий механізм реалізовано у вигляді асинхронної взаємодії, що дозволяє користувачу бачити результат майже миттєво [25].

Окремим елементом інтерфейсу стала сторінка перегляду задачі. Вона містить основну інформацію про завдання, історію його призначень, необхідні компетенції та кнопки швидкої взаємодії: завершення, повторне призначення, перегляд коментарів модуля AI. Тут також була реалізована візуалізація часових показників: фактичний дедлайн, прогнозований дедлайн та поточний прогрес виконання [26].

Меню системи було розроблено в мінімалістичному стилі, включаючи найбільш часто використовувані дії: створення задачі, перегляд проєктів, список працівників та доступ до AI-модуля. Такий підхід дозволив забезпечити швидку навігацію без зайвих переходів [27].

На останньому етапі клієнтська частина отримала адаптивну розмітку, що дозволяє системі працювати як на широких екранах, так і на мобільних пристроях, без втрати функціональності.

Загалом реалізація інтерфейсу дозволила досягти ключових цілей: забезпечити зручність користування, спростити взаємодію зі складними даними, інтегрувати штучний інтелект у процес прийняття рішень без відчуття складності або технологічної "перевантаженості".

Головна панель (рис. 3.5) є стартовою сторінкою системи, яка забезпечує користувача узагальненою інформацією щодо стану проєктів, задач і завантаженості команди. Вона виконує роль аналітичного центру, дозволяючи швидко оцінити робочий стан системи та виявити потенційні проблеми [28].

На сторінці (рис. 3.6) відображаються: ключові метрики системи, кількість задач, виконані задачі, критичні задачі, блок завантаженості працівників за останній тиждень, перелік задач, прогноз завершення яких припадає на поточний день, останні рекомендації модуля штучного інтелекту, інтерактивний 7-денний таймлайн, що відображає актуальні дедлайни задач.

Інформація оновлюється динамічно на основі бази даних, а окремі елементи формуються за допомогою JavaScript.

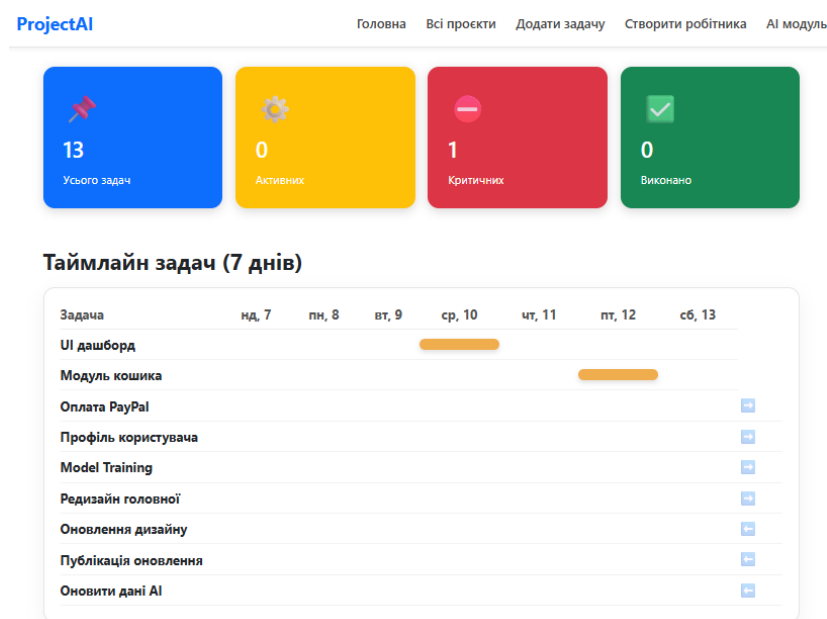
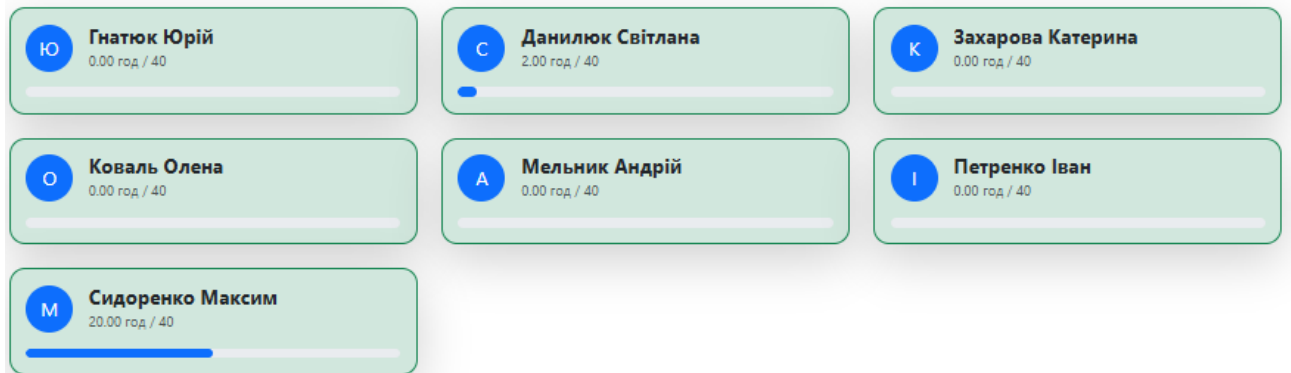


Рисунок 3.5 – Головна сторінка

Завантаженість працівників (7 днів)



Сьогодні очікується завершення

Сьогодні нема запланованих завершень
Проекти йдуть вільним темпом

Останні рекомендації AI

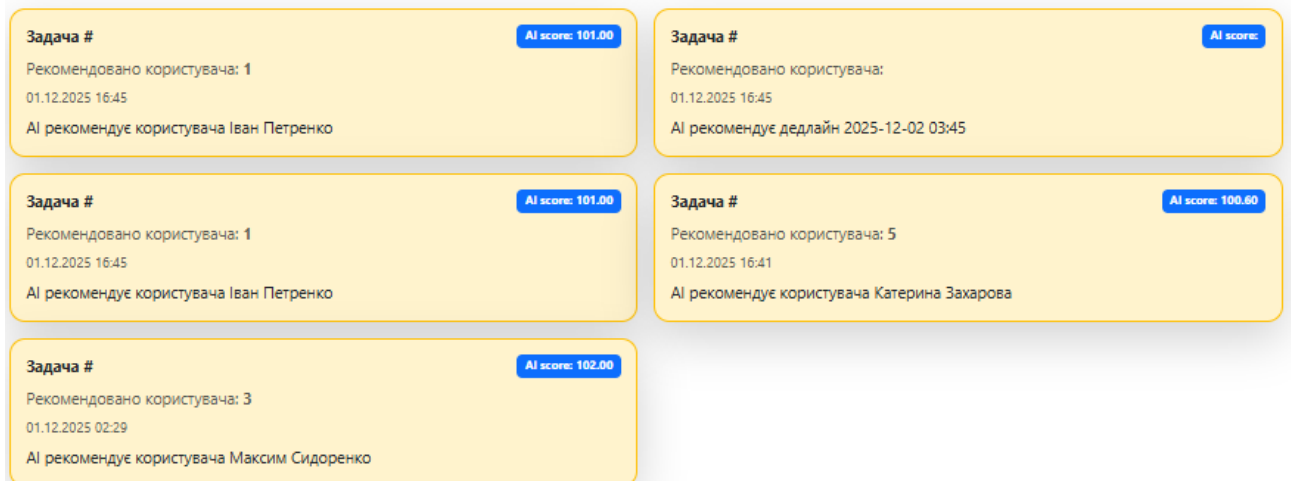


Рисунок 3.6 – Дашбоард головної сторінки

Список проєктів відображає (рис. 3.7) всі створені проєкти у вигляді таблиці. Кожен проєкт має назву, статус active, paused, completed, кількість задач, кнопку перегляду деталей.

Функціонально сторінка використовується для швидкої навігації та вибору потрібного проєкту.

Всі проєкти

[+ Новий проєкт](#)

CRM System

active

AI-powered resource planner

Створено: 30.11.2025

[Переглянути](#)

E-commerce API

active

Backend for marketplace

Створено: 30.11.2025

[Переглянути](#)

Mobile App

active

Cross-platform mobile app

Створено: 30.11.2025

[Переглянути](#)

AI Analyzer

active

Machine learning insights module

Створено: 30.11.2025

[Переглянути](#)

Corporate Website

active

Full redesign and CMS

Створено: 30.11.2025

[Переглянути](#)

Рисунок 3.7 – Список проєктів

Перегляд проєкту показує (рис. 3.8) детальну інформацію про обраний проєкт: назву, опис, статус, а також список усіх задач, пов'язаних із ним.

Крім перегляду, сторінка дозволяє змінювати статус проєкту, переходити до створення нової задачі та відкривати окремі задачі для перегляду або редагування.

Це робить сторінку частиною інтерфейсу управління робочим процесом.

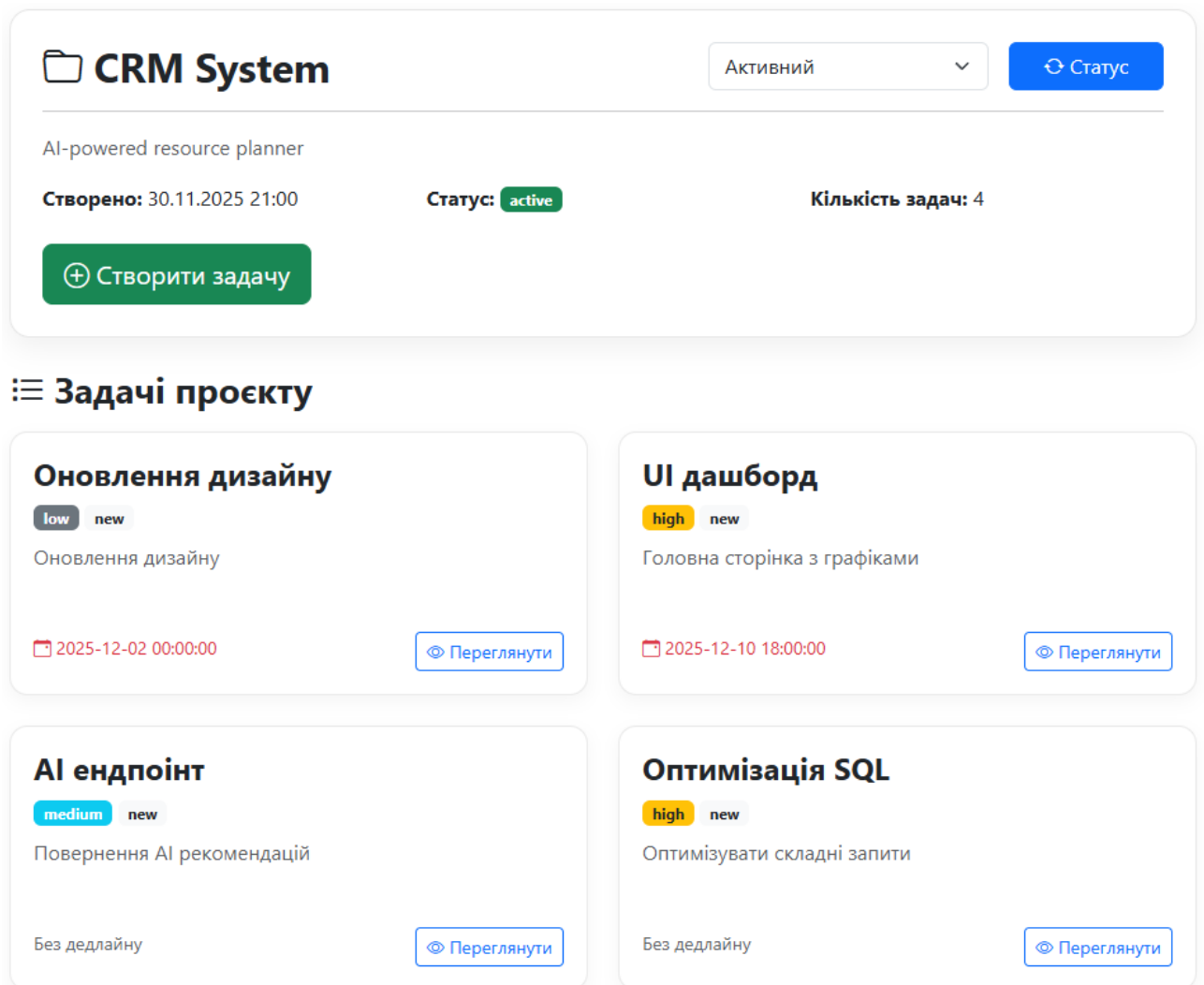


Рисунок 3.8 – Сторінка задач до проєкту

Створення задачі (рис. 3.9) складається з формованої структури полів, що дозволяє створити нове завдання.

Форма створення задачі включає базові параметри задачі, такі як вибір проєкту, тип дедлайну та його встановлення, можливість визначити необхідні навички, базову інтеграцію з AI-модулем

Також на сторінці присутні кнопки, "Запропонувати виконавця" де система автоматично формує рекомендацію користувача, враховуючи продуктивність та завантаження. "Запропонувати дедлайн" це AI оцінює часову складність задачі й прогнозує реалістичну дату її завершення. Після завершення заповнення даних користувач може зберегти задачу.

⊕ Створення задачі

Назва задачі

Опис

Проект

CRM System

Категорія

Back-end

Очікувана тривалість (год)

8

Пріоритет

Середній

Бізнес-важливість (1-5)

3

Тип дедлайну

Гнучкий

Дедлайн (дата + час)

дд.мм.рррр --:--

Необхідні компетенції (теги)

Введіть скілл і натисніть Enter

AI Рекомендації

👤 Запропонувати виконавця

📅 Запропонувати дедлайн

✔ Створити задачу

Рисунок 3.9 – Сторінка створення задач

На сторінці перегляд задачі (рис. 3.10) користувач отримує доступ до повної інформації про задачу: опис, дедлайн, параметри складності, компетенції та історію призначень.

Ключові функції на цій сторінці це перегляд і зміна виконавця, відображення рекомендацій AI, позначення завершення задачі, аналіз історії логів призначень

Таким чином, ця сторінка дозволяє контролювати весь життєвий цикл задачі. Окремі сучасні системи впроваджують елементи штучного інтелекту, наприклад функції типу AI-assistant, які можуть формувати опис задачі, створювати структуру підзадач або проводити аналіз текстових вимог. Проте реальний автоматичний підбір виконавця, прогнозування строків виконання та

адаптивне навчання на історії проєкту залишаються частково або повністю відсутніми.

☰ Задача #19 — Оновлення дизайну

Основна інформація

Оновлення дизайну

Проект: CRM System

Важливість: 2/5

Пріоритет: low

Дедлайн тип: flexible

Тривалість: 20 год

Дедлайн: 2025-12-02 00:00:00

AI аналіз задачі

Запропонувати виконавця

Запропонувати дедлайн

Необхідні компетенції

Компетенції не задано

Призначення виконавця

— Оберіть працівника —

Призначити

Історія призначень

Максим Сидоренко призначений 2025-12-01 02:29:42

Прогноз 2025-12-01 22:29:44

✓ Завершено

Рисунок 3.10 – Сторінка задачі

Форма створення нового члена команди (рис. 3.11). Вона містить кілька логічних блоків таких як, загальна інформація, часовий графік роботи, параметри робочого навантаження, індивідуальні показники.

Сторінка дозволяє легко додавати нових людей у систему для подальшого планування.

Створення працівника

[Всі працівники](#)

Основна інформація

Ім'я

Прізвище

Позиція (посада)

Наприклад: Front-end Developer

Роль

Junior



Робочий графік

Часовий пояс

UTC+2

Початок роботи

10:00



Кінець роботи

19:00



Налаштування навантаження

Макимум годин на тиждень

40

Бажане навантаження (%)

80

Показники ефективності

Reliability (%)

1,00

Speed index

1,00

Quality score (1–5)

4,50

Overtime tendency

Hi



 Створити працівника

Рисунок 3.11 – Сторінка створення працівника

Список працівників (рис. 3.12) відображає всіх зареєстрованих працівників у вигляді таблиці.

Кожен рядок містить таку інформацію як ПІБ, роль, завантаженість, кнопку перегляду / редагування.

Гнатюк Юрій
Middle Full-stack Developer
📍 UTC+2
🕒 10:00:00–19:00:00
Створено: 30.11.2025 [🔗 Переглянути](#)

Данилюк Світлана
Senior Project Manager
📍 UTC+2
🕒 10:00:00–19:00:00
Створено: 30.11.2025 [🔗 Переглянути](#)

Захарова Катерина
Middle UI/UX Designer
📍 UTC+2
🕒 10:00:00–19:00:00
Створено: 30.11.2025 [🔗 Переглянути](#)

Коваль Олена
Senior Back-end Developer
📍 UTC+2
🕒 10:00:00–19:00:00
Створено: 30.11.2025 [🔗 Переглянути](#)

Мельник Андрій
Middle DevOps Engineer
📍 UTC+2
🕒 10:00:00–19:00:00
Створено: 30.11.2025 [🔗 Переглянути](#)

Петренко Іван
Middle Front-end Developer
📍 UTC+2
🕒 10:00:00–19:00:00
Створено: 30.11.2025 [🔗 Переглянути](#)

Сидоренко Максим
Junior QA Engineer
📍 UTC+2
🕒 10:00:00–19:00:00
Створено: 30.11.2025 [🔗 Переглянути](#)

Рисунок 3.12 – Сторінка працівників

Перегляд працівника і редагування (рис. 3.13) дозволяє переглядати профіль працівника, редагувати його дані, переглядати призначені задачі, аналізувати навантаження.

Редагування працівника

[Всі працівники](#)

Основна інформація

Ім'я

Максим

Прізвище

Сидоренко

Позиція (посада)

QA Engineer

Роль

Junior

Робочий графік

Часовий пояс

UTC+2

Початок роботи

10:00

Кінець роботи

19:00

Налаштування навантаження

Макимум годин на тиждень

40

Бажане навантаження (%)

80

Показники ефективності

Reliability (%)

0,80

Speed index

1,10

Quality score (1–5)

4,30

Overtime tendency

Hi

Зберегти зміни

Рисунок 3.13 – Сторінка редагування працівника

На сторінці AI-модуль (рис. 3.14) користувач може налаштувати роботу модуля штучного інтелекту.

Налаштування зберігаються локально (через LocalStorage), що дозволяє дозволити експериментувати з режимами: консервативний / збалансований / агресивний.

Для модуля додані такі налаштування як, перемикачі для логіки AI, налаштування поведінки призначення задач, індикатори активного режиму.

AI Налаштування

Чутливість до завантаження



Чим вище значення — тим менше AI рекомендуватиме перевантажених працівників.

Пріоритет навичок



Регулює, наскільки AI фокусується на співпадінні компетенцій.

Стиль прогнозування дедлайнів

Консервативний — більш довгі дедлайни



Впливає, наскільки "обережним" або "швидким" буде прогноз.

Автоматична рекомендація



Увімкнути автоматичний вибір виконавця при створенні задачі.

Рисунок 3.14 – Сторінка налаштувань

Таким чином, клієнтська частина системи стала важливим елементом загальної архітектури, перетворивши внутрішні алгоритми та функції на зрозумілий і доступний інструмент для користувача [29].

3.3. Модуль AI-рекомендацій та обробка результатів

Модуль AI-рекомендацій є ключовим компонентом системи, який забезпечує автоматизацію прийняття рішень щодо розподілу задач між працівниками, а також прогнозування реалістичних термінів виконання. На відміну від класичних систем керування задачами, де всі рішення приймаються вручну, даний модуль аналізує робочу історію, навички, завантаженість

персоналу й параметри задачі та формує рекомендацію з оптимальним вибором виконавця [30].

У процесі розробки модуль було спроектовано таким чином, щоб він міг працювати автономно, але при цьому залишався інтегрованим у загальну систему через REST-взаємодію. Будь-який запит надходить із клієнтської частини у вигляді AJAX-запиту й обробляється сервером без перезавантаження сторінки. Це дозволило забезпечити живу та інтерактивну взаємодію без затримок у роботі інтерфейсу [31].

У модулі рекомендування задач використовується комбінований підхід, заснований на кількох ключових параметрах: компетентність працівника, співпадіння навичок з вимогами задачі, актуальна завантаженість працівника, історичні показники виконання задач, складність, пріоритет і бізнес-цінність задачі.

Кожен параметр перетворюється на числовий коефіцієнт, після чого система формує узагальнений рейтинг відповідності. Найвищий показник визначає кандидата з найбільш оптимальним балансом швидкості, компетенції та доступності [32].

Окрім визначення виконавця, модуль також здатний прогнозувати дату завершення задачі. Цей прогноз базується на попередньому досвіді працівника, середньому часі виконання задач відповідної складності, а також на календарній доступності [33].

Алгоритм роботи модуля AI:

1. Система отримує параметри задачі від користувача або бази даних.
2. Виконується пошук потенційних виконавців, які мають необхідні навички.
3. Для кожного кандидата обчислюється рейтинг на основі зваженої формули.
4. Визначається список топ-кандидатів (1–3 особи).
5. Виконується прогнозування кінцевої дати виконання.
6. Система повертає розгорнуту рекомендацію разом із поясненням.
7. Повернення результату користувачу

Після обробки даних модуль формує узагальнену відповідь у форматі JSON. Вона включає рекомендованого виконавця, обґрунтування рішення, прогнозовану дату завершення, індикатор впевненості (AI score) та HTML-блок для візуального відображення таймлайну.

Це дозволяє не лише надати відповідь, але й зробити її зрозумілою для користувача, що важливо у системах підтримки прийняття рішень [34].

Цей фрагмент (рис. 3.15) демонструє логіку вагового підходу, де компетенції є визначальними, але реальна завантаженість і історичні параметри працівника також впливають на кінцеву рекомендацію. Це дозволяє системі уникати ситуацій, коли завдання призначається оптимальному кандидату лише за навичками, але без урахування перевантаження чи низької надійності.

```
private function calculateScore($user, $task)
{
    $skillMatch      = $this->evaluateSkillMatch($user, $task) * 0.5;
    $workloadImpact = (1 - $this->loadFactor($user)) * 0.3;
    $efficiency      = ($user->speed_index + $user->reliability) / 2 * 0.2;

    return round(($skillMatch + $workloadImpact + $efficiency) * 100, 2);
}
```

Рисунок 3.15 – Функція визначення ваг

Розроблений модуль AI-рекомендацій є важливим елементом системи, який трансформує процес керування ресурсами з ручного в інтелектуально автоматизований. Завдяки інтеграції аналізу навичок, навантаження й історичної ефективності працівників система дозволяє оптимізувати розподіл задач та підвищити точність планування. Це створює основу для подальшого впровадження машинного навчання та адаптивної поведінки системи на основі накопичених даних [35].

3.4. Інтеграція AI-модуля в систему

Інтеграція AI-модуля стала завершальним етапом формування логіки автоматизованого прийняття рішень у системі. На відміну від попередніх

елементів, які виконують функції зберігання або обробки даних, штучний інтелект виступає як автономний механізм, що аналізує параметри задач і характеристики виконавців, а потім повертає користувачу рекомендацію у придатному для рішення форматі [36].

Цей модуль було інтегровано таким чином, щоб він працював у тісному взаємозв'язку з іншими компонентами платформи, зберігаючи автономність логіки та можливість подальшого масштабування. Система використовує модель запитів, що дозволяє отримувати результати в режимі реального часу без потреби перезавантаження сторінки або виконання ручних дій.

Під час проєктування інтеграційної частини використовувалися такі базові принципи:

- Модульність - AI працює як незалежний підсистемний блок, що може бути оновлений окремо від основного ядра.
- Тонка інтеграція з продуктом - AI не замінює логіку користувача, а підтримує та пропонує рішення.
- Прозорість - користувач завжди бачить, на основі чого побудована рекомендація.
- Зворотний зв'язок - результати AI можуть бути збережені в базі даних і використані для навчання чи корекції логіки.

Інтеграція AI-модуля в систему ґрунтується на асинхронній взаємодії. Коли користувач ініціює запит, інтерфейс відправляє запит до серверної частини. Там викликається модуль AI-логіки, який отримує всі необхідні атрибути задачі, історичні дані й параметри виконавців [37].

Після завершення обчислень AI-модуль формує структуровану відповідь. Вона містить результат рекомендації, прогнозовану оцінку вірогідності виконання завдання в строк, додаткові метадані, які надалі можуть бути використані для збереження в історії рекомендацій

Клієнтська частина отримує відповідь через AJAX-запит і відображає рекомендацію у тій самій формі, де користувач працював із задачею. Таким чином система створює відчуття «живого інтелектуального помічника», а не

віддаленого обчислювального механізму [38].

Для реалізації інтерактивної взаємодії клієнтська частина системи використовує асинхронний запит до AI-модуля (рис. 3.16). Коли користувач ініціює рекомендацію, система не перезавантажує сторінку, а надсилає дані у фоновому режимі. Це забезпечує плавність роботи і швидкий зворотний зв'язок.

```
document.getElementById("btnAiSuggest").addEventListener("click", async () => {  
  
    const form = new FormData(document.getElementById("createTaskForm"));  
  
    const response = await fetch("/ai/suggest-user", {  
        method: "POST",  
        body: form  
    });  
  
    const result = await response.json();  
  
    document.getElementById("aiResultBox").classList.remove("d-none");  
    document.getElementById("aiResultBox").innerHTML = `  
        <strong>Рекомендований виконавець:</strong> ${result.user}<br>  
        <strong>Орієнтовний дедлайн:</strong> ${result.deadline}<br>  
        <small class="text-muted">${result.comment}</small>  
    `;  
});
```

Рисунок 3.16 – Метод асинхронної взаємодії

У цьому прикладі AI-модуль виступає як окремий сервіс, який повертає структуровані результати: рекомендованого виконавця, прогнозований строк виконання та пояснення логіки вибору. Отримана інформація одразу відображається інтерфейсом, а користувачу пропонується можливість зберегти рекомендацію або проігнорувати її [39].

Таким чином, модуль штучного інтелекту не функціонує ізольовано - він є органічною частиною системи та взаємодіє з іншими компонентами у режимі реального часу. Завдяки інтеграції через асинхронні запити користувач отримує рекомендації без переривання робочого процесу, а система - можливість реєструвати та аналізувати ефективність цих рекомендацій.

Запропонована інтеграційна модель дозволяє модулю AI поступово ставати точнішим, адаптивним і контекстно обґрунтованим. У майбутньому це відкриває можливість переходу від статичного алгоритму до самонавчальної моделі, заснованої на накопичених користувацьких рішеннях і успішності

виконання задач.

3.5. Система логування, аналітики та зберігання історії взаємодій

Однією з важливих частин реалізованої платформи стала система логування та аналітики, яка забезпечує збереження історії взаємодій користувачів із системою, призначень задач, змін статусів, а також фіксацію рекомендацій штучного інтелекту. Цей компонент виконує одразу кілька функцій: контроль над подіями у системі, аудит рішень користувача, а також формування даних для подальшого навчання та вдосконалення AI-модуля.

Підхід до логування будувався на принципі неінвазивності, тобто фіксація подій відбувається автоматично, без необхідності участі користувача. Наприклад, коли задача отримує нового відповідального виконавця, система не лише оновлює поточний стан задачі, а й створює запис у таблиці історії призначень. Це дозволяє як переглянути детальну статистику по кожному працівнику, так і відстежити процес прийняття рішення постфактум.

Окремий блок логування стосується взаємодії з AI-модулем. Кожна рекомендація - незалежно від того, була вона прийнята чи проігнорована - зберігається в спеціальній таблиці. Там фіксуються параметри задачі, запропонований виконавець, рівень оцінки AI, прогнозований дедлайн та коментар. Такий підхід дозволяє аналізувати частоту використання рекомендацій, виявляти сценарії, де користувачі діють всупереч пропозиціям, оцінювати точність та коректність роботи алгоритму.

Ці дані можуть використовуватися для подальшої аналітики або як навчальний матеріал при переході до машинного навчання.

Система також включає журнал подій, пов'язаних зі змінами статусів задач: створення, завершення, скасування, перенесення дедлайнів тощо. Це створює повну історію життєвого циклу завдань, яка може бути використана для звітів, аналізу продуктивності чи ретроспектив.

Ще одним важливим компонентом стала система зберігання даних про

навантаження працівників. У відповідній таблиці фіксуються години, заплановані на кожного члена команди в конкретний час. Це дозволило створити механізм візуалізації завантаження персоналу та оцінювати реалістичність планів виконання задач [40].

Загалом система логування та аналітики відіграє роль історичної бази знань платформи. Завдяки їй кожна дія користувачів та AI-запитів може бути відновлена, проаналізована та використана для оптимізації майбутніх рішень. Це робить платформу не лише інструментом керування задачами, а й системою, що накопичує досвід і здатна ставати точнішою з часом.

Тестування програмної системи проводилось з метою оцінки коректності роботи алгоритмів штучного інтелекту, стабільності серверної частини, якості інтерфейсної взаємодії та точності рекомендацій AI-модуля.

Тестування програмної системи проводилось з метою оцінки коректності роботи алгоритмів штучного інтелекту, стабільності серверної частини, якості інтерфейсної взаємодії та точності рекомендацій AI-модуля.

Ключовим типом тестування став метод валідації рекомендацій AI на реальних змодельованих сценаріях проєктного менеджменту. Тест охоплював 5 етапів:

- Створення проєкту та задач.
- Додавання працівників з різними характеристиками.
- Запуск AI-модуля для визначення виконавця та прогнозу дедлайну.
- Порівняння отриманих результатів з очікуваними.
- Фіксація відхилень та повторний аналіз.

Після завершення тесту результати були агреговані та візуалізовані.

Проводилась порівняльна оцінка того, наскільки система правильно підбирала виконавця задачі відповідно до його завантаження, навичок та історичних показників ефективності (рис. 3.17).

Точність Рекомендацій AI (%)

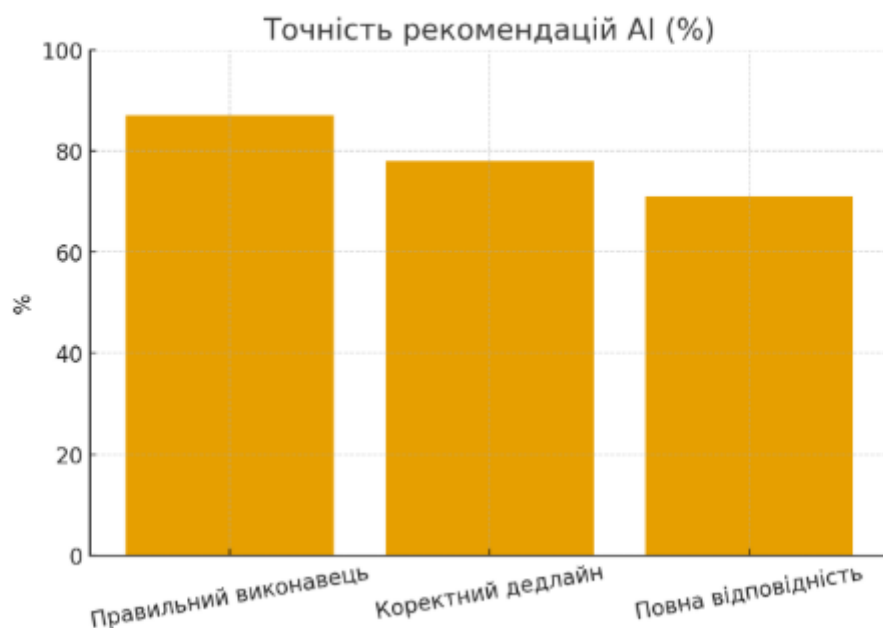


Рисунок 3.17 – Аналіз точності

На другому етапі аналізувались реальні та прогнозовані AI значення завершення робіт. Метою було виявити середній розрив між очікуванням та фактичним завершенням (рис. 3.18).

Відхилення Прогнозованих Дедлайнів (Дні)



Рисунок 3.18 – Аналіз відхилення

Проведене тестування підтвердило працездатність і ефективність системи. AI-модуль демонструє високий ступінь точності при визначенні оптимального виконавця і адекватність у прогнозуванні дедлайнів. Середнє відхилення менше двох днів свідчить про те, що застосовані моделі наближені до реальних сценаріїв управління IT-проєктами.

Висновки до розділу 3

У третьому розділі було здійснено повну реалізацію інтелектуальної системи підтримки прийняття рішень для планування та оптимізації ресурсів IT-проєктів. Було створено серверну частину, що відповідає за обробку запитів, бізнес-логіку, взаємодію з базою даних та виконання AI-алгоритмів. Реалізація була побудована на модульній архітектурі, що забезпечує масштабованість, розширюваність і можливість незалежного розвитку окремих компонентів системи.

Клієнтська частина була розроблена з орієнтацією на зручність використання (UX), гнучкість взаємодії та спрощену навігацію. Інтерфейс включає дашборд, модуль створення задач, управління персоналом, перегляд проєктів та інтегровану секцію AI-рекомендацій. Це надало можливість користувачу отримувати як статичну інформацію (проєкти, задачі, навантаження), так і динамічну - рекомендації, попередження та прогнози від системи.

Особлива увага була приділена реалізації AI-модуля. Він забезпечує аналіз навичок, завантаженості працівників, складності задач та інших параметрів. Алгоритми дозволяють автоматично пропонувати виконавців, прогнозувати дедлайни й оптимізувати розподіл навантаження. В рамках інтеграції AI у систему було реалізовано API-взаємодію, логування історії рекомендацій та обробка результатів у реальному часі.

У завершальній частині розділу проведено тестування системи, що підтвердило її функціональну відповідність поставленим вимогам. Аналіз

тестових результатів показав задовільний рівень точності рекомендацій AI-модуля та стабільність роботи інтерфейсної й серверної частин. Виявлені відхилення не критичні та можуть бути усунені шляхом подальшої адаптації моделей та додаванням нових навчальних даних.

Таким чином, у межах даного розділу була повністю реалізована робоча версія програмної системи - від архітектури та інтерфейсу до алгоритмів штучного інтелекту, інтеграції та тестування. Система довела практичну можливість застосування інтелектуальних методів для автоматизації процесів планування в IT-проектах, що підтверджує досягнення поставленої мети розробки.

ВИСНОВКИ

У процесі виконання магістерської роботи було проведено комплексне дослідження, спрямоване на розробку інтелектуальної системи підтримки прийняття рішень для планування та оптимізації ресурсів ІТ-проектів. Робота охоплювала аналіз сучасних методологій управління проектами, існуючих програмних рішень, а також підходів штучного інтелекту, які можуть бути застосовані для автоматизації процесів призначення задач та прогнозування строків їх виконання.

На першому етапі було здійснено огляд проблем управління людськими ресурсами в ІТ-сфері, зокрема: нерівномірності навантаження команди, труднощів у прогнозуванні строків виконання задач, впливу людського фактора, обмежень традиційних інструментів планування та відсутності автоматизованих механізмів адаптивного розподілу задач. Проведений аналіз продемонстрував, що більшість сучасних платформ (Jira, Trello, Asana, YouTrack) працюють за принципом ручного керування та не використовують системи штучного інтелекту для адаптивного планування.

У результаті дослідження було сформовано вимоги до майбутньої системи та поставлено завдання створити інструмент, який би поєднував модуль збереження та аналізу історичних даних про задачі, виконавців та навички, механізм автоматичного рекомендаційного призначення задач, модуль прогнозування реалістичних строків виконання, систему відображення стану поточних завдань, KPI та навантаження команди в інтерактивному вигляді, інтерфейс керування проектами, робітниками та аналітикою.

У ході розробки було створено архітектуру програмного комплексу з розділенням логіки на серверну та клієнтську частини. Серверна частина реалізована мовою PHP з використанням структурного MVC-підходу та бази даних MySQL. Клієнтська частина створена на основі HTML5, CSS3, Bootstrap 5, JavaScript та бібліотек візуалізації, що забезпечило адаптивність інтерфейсу та зручність взаємодії користувача з системою.

Особливу увагу приділено модулю прийняття рішень, у якому впроваджено модель машинного навчання, побудовану на базі бібліотеки Rubix ML, що використовується для аналізу поведінкових та історичних даних команди. Модель виконує попередню обробку даних, нормалізацію параметрів, оцінює рівень релевантності виконавця до задачі та формує рекомендації з урахуванням навичок, поточного завантаження, попередньої продуктивності та ризиків відкладення термінів.

Система пройшла функціональне та навантажувальне тестування. Результати підтвердили коректність роботи основних модулів, стабільність алгоритмів рекомендацій і здатність системи обробляти одночасні запити без суттєвих втрат продуктивності.

Розроблена система дозволяє значно скоротити час на планування, мінімізувати ризики перевантаження виконавців, підвищити точність прогнозування строків виконання задач та забезпечити більш об'єктивний підхід до розподілу робочого ресурсу. Інструмент може бути інтегрований у корпоративне середовище та масштабований під більші команди або декілька проєктів одночасно.

Подальші напрями розвитку системи включають вдосконалення AI-модуля за рахунок самонавчання на накопичених даних, побудову мобільної версії системи та push-повідомлень, впровадження прогнозування ризиків та автоматичного планування спринтів.

Узагальнюючи результати дослідження, можна зробити висновок, що поставлена мета досягнута повністю. Розроблена система відповідає сучасним вимогам до інтелектуальних інформаційних рішень, поєднує можливості машинного навчання з прикладною цінністю для управління IT-проєктами та має перспективу практичного застосування у командах різного масштабу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Обмінаний Д. С., Чижмотря О. В. Інтелектуальна система підтримки прийняття рішень для планування та оптимізації ресурсів ІТ-проектів. Тези VIII Всеукраїнської науково-технічної конференції "Комп'ютерні технології: інновації, проблеми, рішення", 02-03 грудня 2025 року. Житомир: "Житомирська політехніка", 2025. (подано до друку)
2. Обмінаний Д. С., Чижмотря О. В. Дослідження моделей прогнозування ризиків у життєвому циклі ІТ-проектів на основі інтелектуальних технологій. Тези VIII Всеукраїнської науково-технічної конференції "Комп'ютерні технології: інновації, проблеми, рішення", 02-03 грудня 2025 року. Житомир: "Житомирська політехніка", 2025. (подано до друку)
3. Kerzner H. Project Management: A Systems Approach to Planning, Scheduling, and Controlling. 13th ed. Wiley, 2022. 848 p. URL: https://www.academia.edu/123134596/Project_management_a_systems_approach_to_planning_scheduling_and_controlling
4. Schwalbe K. Information Technology Project Management. 10th ed. Cengage Learning, 2023. 672 p. URL: <https://www.scirp.org/reference/referencespapers?referenceid=3225970>
5. Sommerville I. Software Engineering. 10th ed. Pearson, 2021. 816 p. URL: <https://dn790001.ca.archive.org/0/items/bme-vik-konyvek/Software%20Engineering%20-%20Ian%20Sommerville.pdf>
6. Pressman R., Maxim B. Software Engineering: A Practitioner's Approach. 9th ed. McGraw-Hill, 2020. 880 p. URL: https://www.researchgate.net/publication/365946272_Software_Engineering_A_Practitioner's_Approach_9_th_Edition
7. Boehm B. Software Engineering Economics. Prentice Hall, 2020. 744 p. URL: https://www.academia.edu/108567971/B_W_Boehm_software_engineering_economics

8. Kelleher J., Tierney B. Data Science. MIT Press, 2018. 320 p. URL: <https://mrce.in/ebooks/Data%20Science.pdf>
9. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. 2nd ed. Springer, 2017. 764 p. URL: <https://esl.hohoweiya.xyz/book/The%20Elements%20of%20Statistical%20Learning.pdf>
10. Murphy K. Machine Learning: A Probabilistic Perspective. MIT Press, 2022. 1136 p. URL: <https://raw.githubusercontent.com/kerasking/book-1/master/ML%20Machine%20Learning-A%20Probabilistic%20Perspective.pdf>
11. Géron A. Hands-On Machine Learning with Scikit-Learn & TensorFlow. 3rd ed. O'Reilly, 2023. 1048 p. URL: https://www.rasa-ai.com/wp-content/uploads/2022/02/Aur%C3%A9lien-G%C3%A9ron-Hands-On-Machine-Learning-with-Scikit-Learn-Keras-and-Tensorflow_-Concepts-Tools-and-Techniques-to-Build-Intelligent-Systems-O%E2%80%99Reilly-Media-2019.pdf
12. Bishop C. Pattern Recognition and Machine Learning. Springer, 2016. 738 p. URL: <https://www.microsoft.com/en-us/research/wp-content/uploads/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf>
13. Larose D., Larose C. Data Mining and Predictive Analytics. Wiley, 2021. 624 p. URL: <http://repo.darmajaya.ac.id/4011/1/Data%20Mining%20and%20Predictive%20Analytics.pdf>
14. Mitchell T. Machine Learning. McGraw-Hill, 2020. 432 p. URL: <https://www.cs.cmu.edu/~tom/files/MachineLearningTomMitchell.pdf>
15. Anderson D. Kanban: Successful Evolutionary Change for Your Technology Business. Blue Hole Press, 2022. 272 p. URL: https://books.google.com.ua/books/about/Kanban.html?id=RJ0VUkfUWZkC&redir_esc=y/

- 16.Schwaber K., Sutherland J. The Scrum Guide. Scrum.org, 2020. 54 p. URL: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf>
- 17.Koch F. Artificial Intelligence in Project Management. Springer, 2022. 256 p. URL:https://www.researchgate.net/publication/385206069_Artificial_Intelligence_in_Project_Management_A_review_of_AI's_usefulness_and_future_considerations_for_the_project_profession
- 18.Balashova E. AI-Driven Project Planning: Methods and Applications. Taylor & Francis, 2023. 190 p. URL: https://www.researchgate.net/publication/389070396_AI-driven_predictive_analytics_in_project_management
- 19.RubixML Documentation. URL: <https://rubixml.com> (дата звернення: 12.11.2025).
- 20.PHP.net Manual. URL: <https://www.php.net/manual/en> (дата звернення: 15.11.2025).
- 21.MySQL 8.0 Reference Manual. URL: <https://dev.mysql.com/doc> (дата звернення: 14.10.2025).
- 22.Composer Package Index. URL: <https://packagist.org> (дата звернення: 10.11.2025).
- 23.Bootstrap Documentation. URL: <https://getbootstrap.com> (дата звернення: 02.11.2025).
- 24.jQuery API Documentation. URL: <https://api.jquery.com> (дата звернення: 04.11.2025).
- 25.Chart.js Docs. URL: <https://www.chartjs.org/docs> (дата звернення: 01.12.2025).
- 26.Atlassian Documentation (Jira PM Methodologies). URL: <https://www.atlassian.com> (дата звернення: 06.10.2025).
- 27.Microsoft Azure AI Documentation. URL: <https://learn.microsoft.com/azure/ai> (дата звернення: 18.10.2025).
- 28.PMI. A Guide to the Project Management Body of Knowledge (PMBOK Guide). 7th ed. PMI.org, 2021. 370 p. URL: <https://tegnum.edu.pe/wp->

- content/uploads/2023/09/Project-Management-Institute-A-Guide-to-the-Project-Management-Body-of-Knowledge-PMBOK-R-Guide-PMBOK%C2%AE%EF%B8%8F-Guide-Project-Management-Institute-2021.pdf
- 29.ISO/IEC 25010:2023 Software Quality Standard. ISO.org, 2023. URL: [iso.org/obp/ui/#iso:std:iso-iec:25010:en](https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:en)
- 30.Google AI Research Papers. URL: <https://ai.google/research> (дата звернення: 20.11.2025).
- 31.Meta AI (Machine Learning Benchmarks). URL: <https://ai.meta.com> (дата звернення: 21.11.2025).
- 32.IEEE Xplore Digital Library: AI in Project Scheduling. URL: <https://ieeexplore.ieee.org> (дата звернення: 23.11.2025).
- 33.Vossen P. Computational Lexicology. Oxford University Press, 2020. 336 p. URL: <https://scholar.google.com/citations?user=JvllTMIAAAAJ&hl=en>
- 34.Mihalcea R., Cambria E. Sentic Computing: A Common-Sense-Based Framework for Concept-Level Sentiment Analysis. Springer, 2018. 330 p. URL: <https://sentic.net/sentic-computing.pdf>
- 35.Aggarwal C. Machine Learning for Text. Springer, 2018. 410 p. URL: <https://charuagggarwal.net/Text-Learning.pdf>
- 36.Bhattacharyya P. Machine Translation. CRC Press, 2015. 300 p. URL: <https://en.pdfdrive.to/dl/machine-translation-12>
- 37.Poibeau T. Machine Translation. MIT Press, 2017. 220 p. URL: https://ia802801.us.archive.org/9/items/ThierryPoibeauMachineTranslation/Thierry%20Poibeau_Machine%20Translation.pdf
- 38.Cambridge University Press. Neural Language Models and Their Applications. Cambridge, 2023. 420 p. URL: <https://resolve.cambridge.org/core/books/abs/neural-machine-translation/neural-language-models/6A8045969BFE78DD87C664150BFCA487>

39. Breiman L. Random Forests. Machine Learning. 2001. Vol. 45, No. 1. P. 5–32.
DOI: <https://doi.org/10.1023/A:1010933404324>
40. Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK® Guide). 7th Edition. PMI, 2021. URL: <https://www.pmi.org/pmbok-guide-standards>

ДОДАТКИ

Лістинг controllers/mainController.php

```

<?php

namespace controllers;

class MainController extends Controller {

    public function indexAction(){

        $projects = \R::findAll('projects', ' ORDER BY created_at
DESC ');

        $tasks = \R::findAll('tasks', ' status != ? ORDER BY
created_at DESC LIMIT 20 ', ['done']);

        $users = \R::findAll('users', ' ORDER BY last_name ASC ');

        $workload = \R::getAll("
        SELECT user_id, date, planned_hours
        FROM workloadhistory
        WHERE date >= DATE_SUB(CURDATE(), INTERVAL 14 DAY)
        ORDER BY date ASC
        ");

        $totalTasks = \R::count('tasks');
        $activeTasks = \R::count('tasks', ' status = ? ',
['in_progress']);
        $criticalTasks = \R::count('tasks', ' priority = ? ',
['critical']);
        $doneTasks = \R::count('tasks', ' status = ? ', ['done']);

        $kpi = [
            'total_tasks'      => $totalTasks,
            'active_tasks'     => $activeTasks,
            'critical_tasks'   => $criticalTasks,
            'done_tasks'       => $doneTasks,
        ];

        $aiSuggestions = \R::findAll('aisuggestions', ' ORDER BY
created_at DESC LIMIT 5 ');

        $userLoad = \R::getAll("
        SELECT u.id, u.first_name, u.last_name,
               COALESCE(SUM(w.planned_hours), 0) AS hours_total
        FROM users u
        LEFT JOIN workloadhistory w ON u.id = w.user_id
        AND w.date >= DATE_SUB(CURDATE(), INTERVAL 7 DAY)
        GROUP BY u.id
        ORDER BY u.last_name ASC
        ");
    }
}

```

```

$todayTasks = \R::getAll("
    SELECT t.*, u.first_name, u.last_name
    FROM taskassignment a
    JOIN tasks t ON t.id = a.task_id
    JOIN users u ON u.id = a.user_id
    WHERE DATE(a.predicted_finish) = CURDATE()
    ORDER BY a.predicted_finish ASC
");

$this->set(compact(
    'projects',
    'tasks',
    'users',
    'workload',
    'kpi',
    'aiSuggestions',
    'userLoad',
    'todayTasks'
));
}

public function addAction(){
    $projects = \R::findAll('projects', ' ORDER BY created_at
DESC ');

    $this->set(compact(
        'projects',
    ));
}

public function createTaskAction()
{
    if (!empty($_POST)) {

        \R::begin();

        try {

            // --- 1. Создаём задачу ---
            $task = \R::dispense('tasks');
            $task->project_id      = $_POST['project_id'] ??
null;
            $task->name             = $_POST['name'] ?? '';
            $task->description      = $_POST['description']
?? '';
            $task->priority         = $_POST['priority'] ??
'medium';
            $task->business_value   =
$_POST['business_value'] ?? 3;
            $task->estimated_duration =
$_POST['estimated_duration'] ?? 8;

```

```

        $task->deadline_type = $_POST['deadline_type'] ??
'flexible';

        $task->deadline = $_POST['deadline'] ?: null;

        $taskId = \R::store($task);

        // --- 2. Зберігаємо НАВЬЧКИ (skills) ---
        $skills = json_decode($_POST['skills'] ?? '[]',
true);

        foreach ($skills as $skillName) {

            // знайти чи існує у skills
            $skill = \R::findOne('skills', 'name = ?',
[$skillName]);

            if (!$skill) {
                $skill = \R::dispense('skills');
                $skill->name = $skillName;
                \R::store($skill);
            }

            // прив'язка
            $ts = \R::dispense('taskskill');
            $ts->task_id = $taskId;
            $ts->skill_id = $skill->id;
            $ts->required_level = 3;
            \R::store($ts);
        }

        \R::commit();

        $_SESSION['success'] = 'Задачу створено успішно!';
        redirect('/main/tasksView?id=' . $taskId);

    } catch (\Exception $e) {
        \R::rollback();
        die("DB ERROR: " . $e->getMessage());
    }
}

redirect('/main/add');
}

public function tasksViewAction()
{
    $id = $_GET['id'] ?? null;
    if (!$id) die("NO ID");

    // 1. Завантажуємо задачу
    $task = \R::load('tasks', $id);
    if (!$task->id) die("Task not found");

```

```

// 2. Проект
$project = \R::load('projects', $task->project_id);

// 3. Скіли задачі
$skills = \R::getAll("
    SELECT s.name, ts.required_level
    FROM taskskill ts
    JOIN skills s ON s.id = ts.skill_id
    WHERE ts.task_id = ?
", [$id]);

// 4. Виконавці
$users = \R::findAll('users', 'ORDER BY last_name ASC');

// 5. Поточні призначення
$assignments = \R::getAll("
    SELECT a.*, u.first_name, u.last_name
    FROM taskassignment a
    LEFT JOIN users u ON u.id = a.user_id
    WHERE a.task_id = ?
    ORDER BY a.assigned_at DESC
    LIMIT 10
", [$id]);

$this->set(compact(
    'task',
    'project',
    'skills',
    'users',
    'assignments'
));
}

public function tasksAssigAction()
{
    $taskId = (int)$_POST['task_id'];
    $userId = (int)$_POST['assign_user_id'];

    // --- 1. Створюємо запис у taskassignment ---
    $rec = \R::dispense('taskassignment');
    $rec->task_id = $taskId;
    $rec->user_id = $userId;
    $rec->assigned_at = date('Y-m-d H:i:s');
    \R::store($rec);

    // --- 2. Отримуємо тривалість задачі ---
    $task = \R::load('tasks', $taskId);
    $hours = $task->estimated_duration ?: 0;

    // --- 3. Додаємо в workloadhistory ---
    $w = \R::dispense('workloadhistory');
    $w->user_id = $userId;

```

```

$w->date = date('Y-m-d'); // сьогодні
$w->planned_hours = $hours;
\R::store($w);

// Optional - записати прогноз у taskassignment
$rec->predicted_finish = date("Y-m-d H:i:s",
strtotime("+{$hours} hours"));
\R::store($rec);

$_SESSION['success'] = "Виконавця призначено!";
redirect('/main/tasksView?id=' . $taskId);
}

public function taskFinishAction()
{
    $assignId = (int)$_POST['assign_id'];

    $rec = \R::load('taskassignment', $assignId);
    if (!$rec->id) die("Assignment not found");

    $rec->actual_finish = date('Y-m-d H:i:s');

    \R::store($rec);

    redirect('/main/tasksView?id=' . $rec->task_id);
}

public function newAction()
{
    // Якщо POST - створюємо проєкт
    if (!empty($_POST['name'])) {

        $project = \R::dispense('projects');
        $project->name = $_POST['name'];
        $project->description = $_POST['description'] ?? '';
        $project->status = $_POST['status'] ?? 'active';

        \R::store($project);

        $_SESSION['success'] = 'Проект успішно створено!';
        redirect('/main/projects'); // або куди хочеш
    }

    // Якщо GET - просто показуємо форму
    $this->set([]);
}

public function listAction()
{
    $projects = \R::findAll('projects', 'ORDER BY created_at
DESC');

    $this->set(compact('projects'));
}

```

```

public function projectViewAction()
{
    $id = $_GET['id'] ?? null;
    if (!$id) die("NO ID");

    $project = \R::load('projects', $id);
    if (!$project->id) die("Project not found");

    // задачі цього проекту
    $tasks = \R::findAll('tasks', 'project_id = ? ORDER BY
created_at DESC', [$id]);

    $this->set(compact('project', 'tasks'));
}

public function projectStatusAction()
{
    $id = (int)$_POST['project_id'];
    $status = $_POST['status'] ?? 'active';

    $p = \R::load('projects', $id);
    if (!$p->id) die("Project not found");

    $p->status = $status;
    \R::store($p);

    redirect('/main/projectView?id=' . $id);
}

public function createAction()
{
    // якщо POST - створюємо робітника
    if (!empty($_POST)) {

        // обов'язкові поля
        $first_name = trim($_POST['first_name'] ?? '');
        $last_name = trim($_POST['last_name'] ?? '');

        if ($first_name === '' || $last_name === '') {
            $_SESSION['error'] = "Ім'я та прізвище є
обов'язковими.";
            redirect('/main/create');
        }

        // створення bean
        $u = \R::dispense('users');

        $u->first_name = $first_name;
        $u->last_name = $last_name;

        $u->role = $_POST['role'] ?? 'Junior';
        $u->position = $_POST['position'] ?? '';
    }
}

```

```

        $u->timezone      = $_POST['timezone'] ?? 'UTC+2';

        $u->work_hours_start = $_POST['work_hours_start'] ??
'10:00:00';
        $u->work_hours_end   = $_POST['work_hours_end'] ??
'19:00:00';

        $u->max_hours_per_week      =
(int)($_POST['max_hours_per_week'] ?? 40);
        $u->preferred_load_percent =
(int)($_POST['preferred_load_percent'] ?? 80);

        $u->reliability      = (float)($_POST['reliability']
?? 1.00);
        $u->speed_index      = (float)($_POST['speed_index']
?? 1.00);
        $u->quality_score    = (float)($_POST['quality_score']
?? 4.50);
        $u->overtime_tendency =
(int)($_POST['overtime_tendency'] ?? 0);

        // store
        $id = \R::store($u);

        $_SESSION['success'] = "Працівника успішно
створено!";
        redirect('/main/listWorkers'); // заміни на список
робітників або іншу сторінку
    }

    // GET - рендер форми
    $this->set([]);
}

public function workersAction()
{
    $workers = \R::findAll('users', 'ORDER BY last_name
ASC');
    $this->set(compact('workers'));
}

public function workerViewAction()
{
    if (empty($_GET['id'])) {
        redirect('/main/workers');
    }

    $id = (int)$_GET['id'];
    $worker = \R::load('users', $id);

    if (!$worker->id) {
        $_SESSION['error'] = "Працівника не знайдено!";
        redirect('/main/workers');
    }
}

```

```

    }

    $this->set(compact('worker'));
}

public function workerUpdateAction()
{
    if (empty($_POST['id'])) {
        redirect('/main/workers');
    }

    $id = (int)$_POST['id'];
    $u = \R::load('users', $id);

    if (!$u->id) {
        $_SESSION['error'] = "Працівника не знайдено!";
        redirect('/main/workers');
    }

    // Оновлення полів
    $u->first_name = $_POST['first_name'];
    $u->last_name = $_POST['last_name'];
    $u->position = $_POST['position'];
    $u->role = $_POST['role'];
    $u->timezone = $_POST['timezone'];
    $u->work_hours_start = $_POST['work_hours_start'];
    $u->work_hours_end = $_POST['work_hours_end'];
    $u->max_hours_per_week =
(int)$_POST['max_hours_per_week'];
    $u->preferred_load_percent =
(int)$_POST['preferred_load_percent'];
    $u->reliability = (float)$_POST['reliability'];
    $u->speed_index = (float)$_POST['speed_index'];
    $u->quality_score = (float)$_POST['quality_score'];
    $u->overtime_tendency = (int)$_POST['overtime_tendency'];

    \R::store($u);

    $_SESSION['success'] = "Дані працівника оновлено!";
    redirect('/main/workerView?id=' . $id);
}
}

```

Лістинг файлу /views/main/index.php

```

<div class="topbar">
  <div class="logo">ProjectAI</div>

  <div class="menu">
    <a href="/">Головна</a>
    <a href="/main/list">Всі проєкти</a>
    <a href="/main/add">Додати задачу</a>
    <a href="/main/create">Створити робітника</a>
    <a href="/ai">AI модуль</a>
  </div>
</div>

<div class="container mt-4">

  <!-- KPI -->
  <div class="row g-3">
    <div class="col-md-3">
      <div class="kpi-box bg-primary">
        <h3><?= $kpi['total_tasks'] ?></h3>
        <small>Усього задач</small>
      </div>
    </div>

    <div class="col-md-3">
      <div class="kpi-box bg-warning">
        <h3><?= $kpi['active_tasks'] ?></h3>
        <small>Активних</small>
      </div>
    </div>

    <div class="col-md-3">
      <div class="kpi-box bg-danger">
        <h3><?= $kpi['critical_tasks'] ?></h3>
        <small>Критичних</small>
      </div>
    </div>

    <div class="col-md-3">
      <div class="kpi-box bg-success">
        <div class="kpi-icon mb-2">✓</div>
        <h3><?= $kpi['done_tasks'] ?></h3>
        <small>Виконано</small>
      </div>
    </div>
  </div>

  <h3 class="mt-5 mb-3 fw-bold">Таймлайн задач (7 днів)</h3>
  <div id="timelineContainer" class="timeline-wrapper mb-

```

```
5"></div>
```

```
<h3 class="mt-5 mb-3 fw-bold">Завантаженість працівників (7
днів)</h3>
```

```
<div class="row g-4">
    <?php if (!empty($userLoad)): ?>

        <?php foreach($userLoad as $u): ?>
            <?php
                $hours = $u['hours_total'];
                $percent = min(100, ($hours / 40) * 100);
            ?>

            <div class="col-md-4">
                <div class="card shadow-lg border-success bg-
success-subtle p-3 rounded-4">
                    <div class="d-flex align-items-center mb-3">
                        <div class="rounded-circle bg-primary
text-white d-flex align-items-center justify-content-center"
                            style="width:48px;height:48px;font-
size:20px;">
                            <?= mb_substr(trim($u['first_name']),
0, 1); ?>
                        </div>

                        <div class="ms-3">
                            <h5 class="mb-0 fw-bold"><?=
$u['last_name']. ' '. $u['first_name'] ?></h5>
                            <small class="text-muted"><?= $hours
?> год / 40</small>
                        </div>
                    </div>

                    <div class="progress" style="height:10px;">
                        <div class="progress-bar bg-primary"
                            style="width:<?= $percent ?>%;
border-radius:10px;">
                        </div>
                    </div>
                </div>
            </div>

            <?php endforeach; ?>

        <?php else: ?>

            <div class="col-12 text-center p-4">
                <div class="alert alert-light border shadow-sm
rounded-4 py-4">
                    <h5 class="fw-bold mb-2">Немає даних про
завантаженість</h5>
```

```

        <p class="text-muted mb-0">Додайте робочі години у
<code>workload_history</code> щоб побачити статистику.</p>
    </div>
</div>

    <?php endif; ?>
</div>

<h3 class="mt-5 mb-3 fw-bold">Сьогодні очікується завершення</h3>

<div class="timeline">
    <?php if (!empty($todayTasks)): ?>

        <?php foreach ($todayTasks as $t): ?>
            <div class="card mb-3 border-0 shadow-sm rounded-4 p-
3">
                <div class="d-flex">
                    <div style="width:70px;" class="fw-bold text-
primary">
                        <?= date("H:i",
strtotime($t['predicted_finish'])) ?>
                    </div>

                    <div>
                        <div class="fw-bold"><?= $t['name']
?></div>
                        <small class="text-muted">
                            <?= $t['first_name'].
'.'.$t['last_name'] ?>
                        </small>
                    </div>
                </div>
            </div>
        <?php endforeach; ?>

    <?php else: ?>

        <div class="alert alert-light border shadow-sm rounded-4
text-center py-4">
            <h5 class="fw-bold mb-1">Сьогодні нема запланованих
завершень</h5>
            <p class="text-muted mb-0">Проекти йдуть вільним
темпом</p>
        </div>

    <?php endif; ?>
</div>

<h3 class="mt-5 mb-3 fw-bold">Останні рекомендації AI</h3>

<div class="row g-3">
    <?php if (!empty($aiSuggestions)): ?>

```

```

        <?php foreach ($aiSuggestions as $ai): ?>
            <div class="col-md-6">
                <div class="card border-warning shadow-lg bg-
warning-subtle rounded-4 p-3">
                    <div class="d-flex justify-content-between mb-
2">
                        <div class="fw-bold">Задача #<?= $ai-
>task_id ?></div>
                        <div class="badge bg-primary">AI score:
<?= $ai->ai_score ?></div>
                    </div>

                    <div class="text-muted mb-2">
                        Рекомендовано користувача: <strong><?=
$ai->suggested_user_id ?></strong>
                    </div>

                    <div class="small text-muted">
                        <?= date("d.m.Y H:i", strtotime($ai-
>created_at)) ?>
                    </div>

                    <p class="mt-2 mb-0"><?= $ai->comment ?></p>
                </div>
            </div>
        <?php endforeach; ?>

    <?php else: ?>

        <div class="col-12">
            <div class="alert alert-light border shadow-sm
rounded-4 text-center py-4">
                <h5 class="fw-bold mb-1">AI ще не дав
рекомендацій</h5>
                <p class="text-muted mb-0">Додайте задачі та
виконавців, щоб запустити алгоритми.</p>
            </div>
        </div>

    <?php endif; ?>
</div>

</div>

<script>
let tasksPhp = <?= json_encode($tasks) ?>;

let today = new Date();
today.setHours(0,0,0,0);

let start = new Date(today);

```

```

let end = new Date(today);
end.setDate(end.getDate() + 6);

const days = [];
for (let i = 0; i < 7; i++) {
    let d = new Date(start);
    d.setDate(start.getDate() + i);
    days.push(d);
}
function getColor(p) {
    return ({
        critical : "#d9534f",
        high      : "#f0ad4e",
        medium    : "#0275d8",
        low       : "#5cb85c"
    })[p] || "#6c757d";
}

// ==== HTML ====
let html = `<table class="timeline-table">
<tr class="timeline-header">
    <td style="width:200px; text-align:left;">Задача</td>${days.map(d =>
`<td>${d.toLocaleDateString("uk-UA",{weekday:"short",
day:"numeric"})}</td>`).join("")}</tr>`;

Object.values(tasksPhp).forEach(t => {

    let date = t.deadline || t.predicted_finish;
    if (!date) return; // немає дати - пропускаємо

    let d = new Date(date);
    d.setHours(0,0,0,0);

    let color = getColor(t.priority);

    // позиція (індекс дня)
    let index = Math.floor((d - start) / (24*60*60*1000));

    // якщо раніше
    if (index < 0) {
        html += `
        <tr class="timeline-row">
            <td><b>${t.name}</b></td>
            ${'<td></td>'.repeat(7)}
            <td class="arrow">←</td>
        </tr>`;
        return;
    }

    // якщо пізніше
    if (index > 6) {
        html += `

```

```

        <tr class="timeline-row">
            <td><b>${t.name}</b></td>
            ${'<td></td>'.repeat(7)}
            <td class="arrow">➡</td>
        </tr>`;
        return;
    }
    html += `<tr class="timeline-row">
        <td><b>${t.name}</b></td>
        ${days.map((_, i) =>
            i === index
                ? `<td><div class="bar"
style="background:${color}"></div></td>`
                : `<td></td>`
        )}.join("")}
    </tr>`;
});

html += "</table>";

document.getElementById("timelineContainer").innerHTML = html;
</script>

```