НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

«На правах рукопису»
УДК 004.043

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
«___»_____2024 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Програмне забезпечення для планування робочого часу з використанням алгоритмів машинного навчання»

Виконав (-ла):
студент (-ка) ІІ курсу, групи ІП-32мп
Куржумова Марія Ігнатівна _____

Керівник:
Доцент к.е.н.
Родіонов Павло Юрійович _____

Рецензент:
Доцент і.к.с.
Апенько Наталія Вікторівна _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2024 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення інформаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Едуард ЖАРІКОВ

«___»_____2024р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Куржумова Марія Ігнатівна

1. Тема дисертації «Програмне забезпечення для планування робочого часу з використанням алгоритмів машинного навчання», науковий керівник дисертації Родіонов Павло Юрійович, доцент к.е.н., затверджені наказом по університету від «08» листопада 2024 р. № 5016-с

2. Термін подання студентом дисертації «9» грудня 2024 р.

3. Об’єкт дослідження – Процеси розробки архітектурного та програмного рішення для системи планування робочого часу

4. Предмет дослідження – Методи, технології та засоби, що підвищать ефективність виконання завдань користувачами та зменшать час на планування.

5. Перелік завдань, які потрібно розробити – Проведення аналізу існуючих рішень для планування та управління часом, включаючи їхні переваги та недоліки; опис та проєктування функціональних особливостей нового додатку, враховуючи потреби користувачів; визначення та впровадження технологій, які підвищують ефективність програмного забезпечення; розробка програмного забезпечення, спрямованого на підвищення продуктивності й мотивації у користувачів.

6. Перелік графічного (ілюстративного) матеріалу – 4 плакати.

7. Перелік публікацій – дві публікації.

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання «1» вересня 2024 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Видача завдання	1.09.2024	
2	Аналіз існуючих рішень	02.09.24-20.09.24	
3	Формування функціональних та нефункціональних вимог	21.09.24-05.10.24	
4	Розробка бази даних та додавання тестових даних	06.10.24-10.10.24	
5	Розробка методів, алгоритмів	11.10.24-24.10.24	
6	Проектування системи	25.10.24-30.10.24	
7	Реалізація програмного забезпечення	31.10.24-07.11.24	
8	Виконання експериментальних досліджень	08.11.24-11.11.24	
9	Оформлення пояснювальної записки	12.11.22-20.11.24	
10	Подання дисертації на попередній захист	22.11.2024	
11	Подання дисертації на захист	9.12.2024	

Студент Марія КУРЖУМОВА

Науковий керівник Павло РОДІОНОВ

РЕФЕРАТ

Розмір пояснювальної записки – 80 аркушів, містить 6 ілюстрацій, 22 таблиць, 6 додатків, 23 посилань на джерела.

Актуальність теми. На сьогоднішній день багато користувачів стикаються з проблемою неефективного управління часом через зростання обсягу інформації, завдань та зобов'язань як в навчанні, так і в роботі. Існуючі системи планування та організації завдань не завжди забезпечують достатній рівень інтерактивності та гнучкості для вирішення індивідуальних потреб користувачів, що веде до зниження продуктивності та якісного виконання задач.

Мета дослідження. Метою дослідження є покращення програмного забезпечення для системи організації робочого часу, оптимізація процесів управління часом, підвищення продуктивності та ефективності виконання завдань користувачами.

Об’єкт дослідження: Процеси розробки архітектурного та програмного рішення для системи планування робочого часу.

Предмет дослідження: Методи, технології та засоби, що підвищать ефективність виконання завдань користувачами та зменшать час на планування.

Для реалізації поставленої мети **сформульовані наступні завдання**:

− провести аналіз існуючих існуючих рішень для планування та управління часом, включаючи їхні переваги та недоліки;

− дослідити сучасні технології та інструменти, які можуть бути використані для розробки програмного забезпечення;

− вивчити методи та підходи, що застосовуються у проєктуванні систем управління часом із використанням алгоритмів машинного навчання;

− розробити архітектурне рішення для програмного забезпечення;

− розробити програмне забезпечення для планування робочого часу з урахуванням сучасних технологій.

Наукова новизна. Удосконалено метод розподілу завдань, що забезпечує генерацію індивідуального розкладу для користувачів, задля створення більш адаптивної та персоналізованої системи, яка підвищить продуктивність та ефективність планування.

Практичне значення. Практична цінність розробленого програмного забезпечення полягає у його широкому застосуванні як для індивідуальних користувачів, так і для організацій. Крім того, програмне забезпечення має потенціал для застосування в освітніх закладах як для індивідуальних потреб, так і командних проєктів, що сприятиме покращенню організації навчального процесу.

Зв'язок з науковими програмами, планами, темами. Робота виконувалась на кафедрі інформатики та програмної інженерії Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського".

Апробація. Наукові положення дисертації пройшли апробацію на VI Міжнародній науково-практичної конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2023). м. Київ, Україна, 21-23 травня 2024 р. С. 84–88.

Публікації. Наукові положення дисертації опубліковані в:

1) Куржумова М. І., Родіонов П.Ю. Система рекомендацій у програмному забезпеченні для організації робочого часу. Матеріали VI Міжнародної науково-практичної конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2023). м. Київ, Україна, 21-23 травня 2024 р. С. 84–88.

2) Родіонов П. Ю., Марченко О. І., Куржумова М. І. Особливості розроблення програмного забезпечення для організації робочого часу. Наука і техніка сьогодні. 2024. Т. 34, № 6. – С. 1076–1086.

3) **Ключові слова:** ПЛАНУВАННЯ, МАШИННЕ НАВЧАННЯ, НЕЙРОННА МЕРЕЖА, ПЕРСОНАЛІЗАЦІЯ, ЗАВДАННЯ РОЗПОДІЛУ.

ABSTRACT

The size of the explanatory note is 80 sheets, contains 6 illustrations, 21 tables, 6 appendices, 23 references to sources.

Topicality. Today, many users are faced with the problem of ineffective time management due to the growth of the volume of information, tasks and obligations both in education and work. Existing task planning and organization systems do not always provide a sufficient level of interactivity and flexibility to address the individual needs of users, which leads to a decrease in productivity and high-quality task performance.

The aim of the study. The purpose of the research is to improve software for the work time organization system, optimize time management processes, increase the productivity and efficiency of task performance by users.

Research object: Processes for developing an architectural and software solution for a work time planning system.

Research subject: Methods, technologies and tools that will increase the efficiency of task performance by users and reduce planning time.

To achieve this goal, **the following tasks** have been formulated:

− to conduct an analysis of existing solutions for planning and time management, including their advantages and disadvantages;

− to explore modern technologies and tools that can be used for software development;

− to study methods and approaches used in the design of time management systems using machine learning algorithms;

− to develop an architectural solution for software;

− to develop software for planning working hours taking into account modern technologies.

The scientific novelty. The method of task distribution has been improved, which ensures the generation of an individual schedule for users, in order to create a more adaptive and personalized system that will increase the productivity and efficiency of planning.

The practical value. The practical value of the software developed lies in its wide application for both individual users and organizations. In addition, the software has the potential for use in educational institutions for both individual needs and team projects, which will contribute to improving the organization of the educational process.

Relationship with working with scientific programs, plans, topics. Work was performed at the Department of Informatics and Software Engineering of the National Technical University of Ukraine «Kyiv Polytechnic Institute. Igor Sikorsky».

Approbation. The scientific provisions of the dissertation were approbated at the VI International Scientific and Practical Conference of Young Scientists and Students “Software Engineering and Advanced Information Technologies” (SoftTech-2023). Kyiv, Ukraine, May 21-23, 2024. P. 84–88.

Publications. The scientific provisions of the dissertation were published in:

1) Kurzhumova M. I., Rodionov P. Yu. System of recommendations in software for organizing working time. Materials of the VI International Scientific and Practical Conference of Young Scientists and Students “Software Engineering and Advanced Information Technologies” (SoftTech-2023). Kyiv, Ukraine, May 21-23, 2024. P. 84–88.

2) Rodionov P. Yu., Marchenko O. I., Kurzhumova M. I. Features of developing software for organizing working time. Science and Technology Today. 2024. Vol. 34, No. 6. – P. 1076–1086.

Keywords: PLANNING, MACHINE LEARNING, NEURAL NETWORK, PERSONALIZATION, DISTRIBUTION PROBLEM.

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ ТА СКОРОЧЕНЬ

Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня із суворою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом.

Нейронна мережа – це один із напрямків у створенні штучного інтелекту, який ставить своїм завданням відтворити аналітичну роботу людського мозку.

Алгоритм розподілу – це метод організації та управління розподілом даних, обчислень або градієнтів між шарами, пристроями чи вузлами для оптимізації навчання та використання ресурсів.

LSTM – це архітектура рекурентних нейронних мереж (РНМ, штучна нейронна мережа), запропонована 1997 року.

ALS – нейронна мережа, яка використовується в рекомендативних системах для матричної факторизації, коли необхідно знаходити приховані фактори в даних, наприклад, для рекомендації товарів або контенту.

Java – об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java.

JVM – Java Virtual Machine (віртуальна машина Java).

REST – Representational State Transfer (передача стану подання).

Spring Boot – це фреймворк для розробки сервісів на мові Java, що дозволяє швидко і зручно створювати самостійні, готові до виконання застосунки. Він побудований на основі основного фреймворку Spring, проте спрощує процес конфігурації та розгортання застосунків.

PostgreSql – це потужна та розширювана система керування базами даних (СКБД), яка забезпечує зберігання, керування та опрацювання структурованих даних.

SQL – Structured Query Language (мова структурованих запитів).

СКБД – система керування базами даних.

Angular – це популярний фреймворк для розробки вебсервісів з використанням мови програмування TypeScript.

TypeScript – це мова програмування, розроблена компанією Microsoft, яка є розширенням мови JavaScript.

MVC (Model-View-Controller) – це архітектурний шаблон, що використовується в розробці програмного забезпечення для розділення компонентів сервісу на три основні частини: Модель (Model), Вид (View) та Контролер (Controller).

Сервер – це комп'ютер або система, що надає послуги, ресурси або функціональні можливості іншим комп'ютерам або програмам, відомим як клієнти.

Клієнт – це комп'ютер або програма, яка отримує доступ до послуг або ресурсів, наданих сервером. У контексті мережі, клієнтська програма зазвичай звертається до сервера для отримання інформації, надсилання запитів, виконання операцій або отримання результатів обробки.

ВСТУП

На сьогоднішній день багато користувачів стикаються з проблемою неефективного управління часом через зростання обсягу інформації, завдань та зобов'язань як в навчанні, так і в роботі. Існуючі системи планування та організації завдань не завжди забезпечують достатній рівень інтерактивності та гнучкості для вирішення індивідуальних потреб користувачів, що веде до зниження продуктивності та якісного виконання задач. У таких умовах постає необхідність у створенні нових програмних рішень, які здатні адаптуватися до специфічних запитів та особливостей користувачів. Програмне забезпечення для планування робочого часу, що використовує алгоритми машинного навчання, може стати ефективним та унікальним засобом для підвищення якості планування користувачами. Застосування машинного навчання дозволяє створити систему, яка буде змозі проаналізувати поведінку користувача, зробити певні припущення, оптимізувати розподіл завдань та виявляти потенційні конфлікти у графіку. Завдяки цьому забезпечується персоналізований підхід до планування, що значно підвищує рівень задоволеності користувачів та їх продуктивність.

Тоді можемо поставити мету дослідження. Метою є виявлення основних вимог до програмного забезпечення та підвищення його якості шляхом розширення та удосконалення його функціональних можливостей, таких як декомпозиція задач, автоматичне складання розкладу на тиждень, інтеграція зі сторонніми сервісами та застосування системи рекомендацій.

Розробка програмного забезпечення для планування робочого часу із застосуванням алгоритмів машинного навчання дозволить користувачам оптимізувати управління своїм часом відповідно до їхніх індивідуальних потреб та робочих процесів.

Щоб досягти поставленої мети потрібно поставити та виконати такі задачі:

− провести аналіз існуючих існуючих рішень для планування та управління часом, включаючи їхні переваги та недоліки;

− дослідити сучасні технології та інструменти, які можуть бути використані для розробки програмного забезпечення;

− вивчити методи та підходи, що застосовуються у проєктуванні систем управління часом із використанням алгоритмів машинного навчання;

− розробити архітектурне рішення для програмного забезпечення;

− розробити програмне забезпечення для планування робочого часу з урахуванням сучасних технологій.

Отже, можна визначити об’єкт та предмет даного дослідження. Об’єктом є процеси розробки архітектурного та програмного рішення для системи планування робочого часу, а предметом є методи, засоби та технології створення архітектурного та програмного рішення для ефективного планування із застосуванням алгоритмів машинного навчання.

РОЗДІЛ 1: АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Опис предметної області

У сучасному світі питання продуктивності набуває все більшої ваги. Швидкий розвиток технологій та інформаційне перенасичення вимагають від нас удосконалення навичок ефективного управління часом та виконання завдань. Це стосується не лише професійної діяльності, але й навчання та особистого життя. Актуальність цієї роботи зумовлена важливістю планування у сучасному суспільстві, яке стає дедалі більш залежним від технологій та швидкого плину часу. Гаджети, такі як смартфони, планшети та комп'ютери, стали не лише необхідними інструментами для комунікації та роботи, але й джерелом численних відволікаючих факторів. Час, який люди проводять у соціальних мережах або в іграх, може відволікати від важливих завдань та порушувати мету планування.

Підвищення продуктивності є комплексним завданням, яке включає управління часом, планування завдань, мотивацію, а також збереження фізичного та психічного здоров'я. Часто люди стикаються з необхідністю виконати багато завдань у короткий термін, що призводить до перевантаження та зниження ефективності.

З початком війни в Україні багато людей змушені працювати та навчатися дистанційно через обмеження на пересування та небезпеку відкритого перебування в громадських місцях. Це підкреслює важливість ефективного планування, яке дозволяє забезпечити баланс між роботою та особистим часом, враховуючи постійно змінювані обставини.

Наприклад дослідження Рочестерського університету мало на меті визначити вплив щоденних та місячних планів на навчальні звички студентів [1]. Учасники були поділені на три групи. Перша група створювала щоденні досяжні цілі за допомогою робочих аркушів для планування. Друга група зосереджувалася на великих частинах діяльності, формуючи місячні плани.

Третя група не використовувала конкретних планів, отримуючи лише загальні поради, такі як робити перерви після 30-90 хвилин навчання.

Результати показали, що учасники першої групи витрачали менше часу на навчання і припинили використовувати свої планувальники через приблизно місяць. Натомість учасники другої групи були більш продуктивними, відзначаючи більше "ефективного" часу на навчання та меншу схильність до прокрастинації. Багато з них продовжували контролювати свій навчальний час навіть під час весняних канікул.

Дослідження підкреслює важливість індивідуального підходу до планування. Деяким людям підходить більш структуроване щоденне планування, іншим – гнучкіше місячне планування. Щоб уникнути вигорання і підтримувати мотивацію, додатки для планування повинні пропонувати різноманітні варіанти запису завдань. Це дозволить користувачам знаходити оптимальний спосіб організації часу та досягнення цілей.

Ефективне планування є ключовим елементом для оптимізації використання часу та ресурсів. Дослідження показують, що люди, які виділяють час на планування своїх дій, досягають вищого рівня продуктивності та успіху як у професійних, так і в особистих справах. Крім того, планування допомагає уникнути стресу та перенавантаження. Правильно розподіляючи час та енергію, люди можуть зберегти своє психічне та фізичне здоров'я, що є надзвичайно важливим у нашому швидкоплинному світі.

1.2 Аналіз існуючих рішень

1.2.1 Рішення 1: Todoist

Todoist – це популярний інструмент для управління завданнями, який допомагає користувачам організовувати свої справи та підвищувати продуктивність.

Деякі ключові особливості включають:

– Todoist дозволяє створювати проекти, в рамках яких можна додавати завдання та підзадачі. Це дозволяє структурувати роботу та зручно відслідковувати прогрес;

– користувачі можуть встановлювати пріоритети та дедлайни для завдань, що допомагає фокусуватися на найважливіших задачах;

– мітки допомагають категоризувати завдання, а фільтри дозволяють швидко знаходити потрібні задачі за різними критеріями;

– додаток підтримує спільну роботу над проектами, що дозволяє командам ефективно взаємодіяти, ділитися файлами та коментувати завдання.

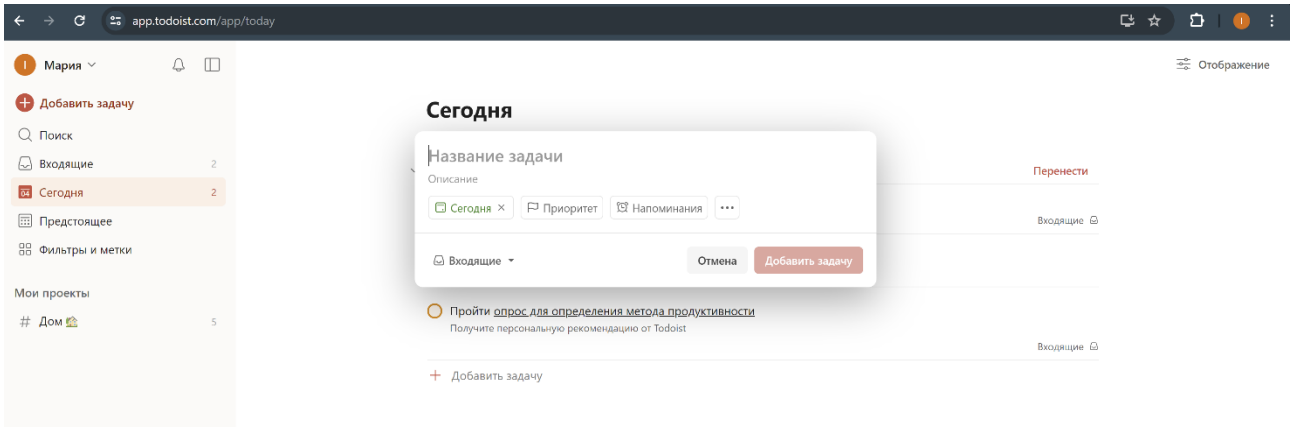


Рисунок 1. Додаток Todoist

Todoist має кілька переваг, що роблять його привабливим для широкого кола користувачів. Однією з головних переваг є простота використання та інтуїтивний інтерфейс, що дозволяє швидко адаптуватися навіть новачкам. Інтеграції з іншими популярними сервісами роблять Todoist зручним для використання в комплексі з іншими інструментами продуктивності. Крім того, можливість встановлення пріоритетів та дедлайнів допомагає користувачам ефективно керувати своїм часом, що є ключовим аспектом у підвищенні продуктивності.

Попри свої численні переваги, Todoist має і деякі недоліки. Одним з основних недоліків є обмежена кількість функціональних можливостей у безкоштовній версії у порівнянні з преміум-планом. Це може бути проблемою для користувачів, які не готові платити за додаткові можливості. У безкоштовній версії обмежена кількість проєктів, міток та фільтрів, що може

ускладнювати організацію роботи. Також, хоча додаток пропонує різні функції для управління завданнями, відсутність розширених можливостей аналітики та статистики може бути недоліком для користувачів, які бажають детально аналізувати свою продуктивність. Відсутність таких функцій, як детальні звіти про виконання завдань та моніторинг часу, зменшує потенціал Todoist як інструменту для глибокого аналізу ефективності. Іншим недоліком є те, що іноді додаток може бути надто простим для просунутих користувачів, які потребують більше налаштувань і функцій для складних проектів. Відсутність функціоналу обмежує використання Todoist для великих команд або складних проектів.

Щоб зробити Todoist кращим, можна звернути увагу на декілька аспектів. По-перше, розширення функціональності безкоштовної версії може залучити більше користувачів, які згодом можуть перейти на платний план. Це може включати додаткові мітки, фільтри або інтеграції, які зроблять додаток більш привабливим для широкої аудиторії. По-друге, додавання розширених можливостей аналітики та статистики дозволило б користувачам краще розуміти свою продуктивність та виявляти області для покращення. Це може включати детальні звіти, графіки продуктивності та інші інструменти для аналізу виконання завдань.

1.2.2 Рішення 2: Monday.com

Monday.com – це платформа для управління робочими процесами, яка спрямована на підвищення ефективності командної роботи. Вона дозволяє командам планувати, відстежувати та координувати роботу за допомогою різних інструментів і функцій. Додаток має широкий спектр можливостей для налаштування, що робить його універсальним інструментом саме для командної роботи.

Особливості Monday.com:

– Monday.com надає зручний інтерфейс у вигляді канбан-дошки, що дозволяє командам візуально організовувати завдання та відстежувати їх виконання;

– платформа пропонує велику кількість готових шаблонів для різних видів проектів, що дозволяє швидко налаштувати систему під конкретні потреби команди;

– додаток дозволяє командам працювати в режимі реального часу, писати коментарі, прикріплювати файли та оновлювати статуси завдань;

– Monday.com підтримує автоматизацію робочих процесів, що дозволяє зменшити рутину та підвищити ефективність роботи команди.

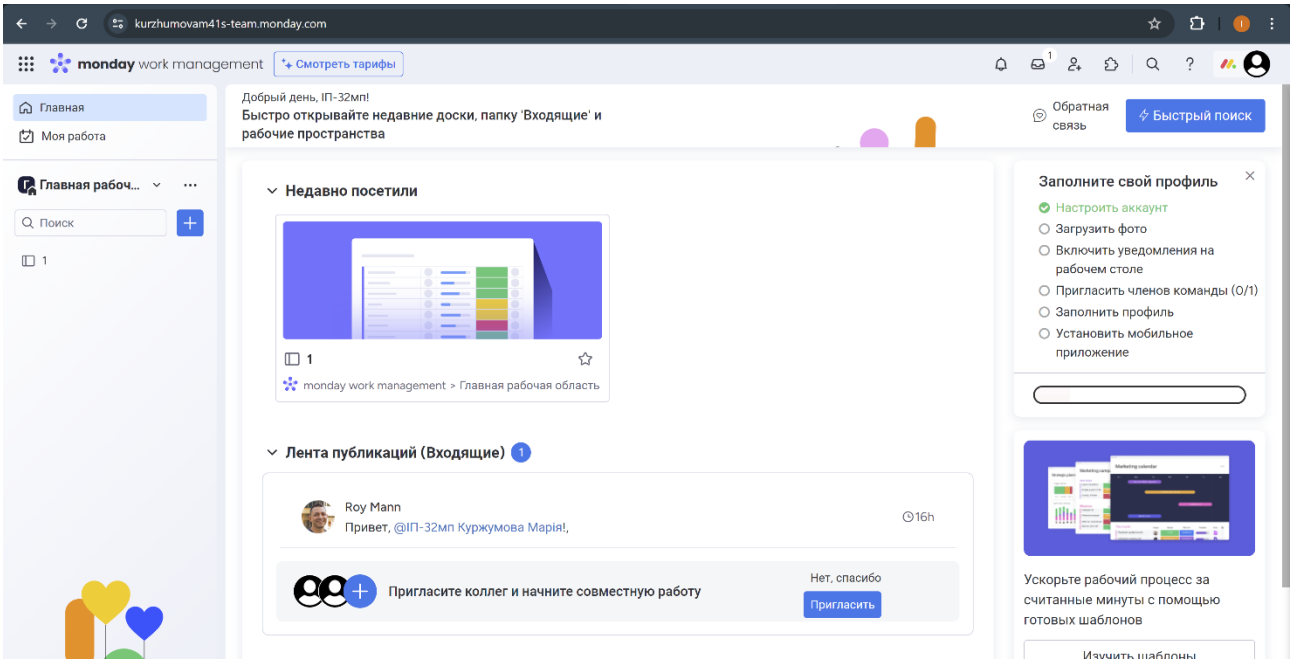


Рисунок 2. Додаток Monday.com

Однією з головних переваг Monday.com є його орієнтованість на командну роботу. Інструмент забезпечує високий рівень співпраці та координації між членами команди, що дозволяє ефективно планувати та виконувати завдання. Гнучкість у налаштуваннях та велика кількість інтеграцій робить Monday.com універсальним рішенням для різних типів бізнесів. Інтерфейс платформи є інтуїтивно зрозумілим і зручним у використанні. Завдяки можливості візуалізувати робочі процеси у вигляді

канбан-дошки або інших форматів, користувачі можуть легко відстежувати прогрес проектів та швидко вносити необхідні зміни. Автоматизація робочих процесів також є значною перевагою, яка дозволяє зменшити час на рутинні завдання та підвищити продуктивність команди.

Попри численні переваги, Monday.com має і певні недоліки. Одним з основних є відсутність багатьох функцій для індивідуального планування та управління завданнями. Платформа зосереджена на командній роботі, що може бути не зовсім зручним для користувачів, які шукають інструмент для особистого використання або для невеликих проектів. Крім того, відсутність розширених функцій для аналітики та статистики може бути обмеженням для компаній, які потребують детального аналізу продуктивності та ефективності робочих процесів. Відсутність таких можливостей, як детальні звіти та графіки продуктивності, зменшує потенціал Monday.com як інструменту для глибокого аналізу. Іншим недоліком є висока вартість преміум-планів, що може бути перешкодою для малих бізнесів та стартапів. Хоча платформа пропонує широкий спектр функцій, їх вартість може бути значною для компаній з обмеженим бюджетом. Для покращення Monday.com можна зосередитися на розширенні можливостей для індивідуального планування та управління завданнями. Додавання функцій для особистого використання, таких як календарі, нагадування та індивідуальні цілі, зробило б платформу більш універсальною. Додатково, розширення аналітичних можливостей платформи, таких як детальні звіти та графіки продуктивності, могло б значно підвищити її цінність для компаній, що потребують глибокого аналізу своїх робочих процесів. Це дозволило б користувачам краще розуміти ефективність своєї роботи та виявляти області для покращення.

1.3 Постановка задачі

Задача полягає у розробці додатку для індивідуального та командного планування з урахуванням недоліків вже існуючих рішень. Додаток має

забезпечувати зручний інтуїтивний інтерфейс для користувачів, що дозволяє ефективно організовувати свої задачі та плани.

Додаток повинен підтримувати функцію running list, яка дозволяє користувачам створювати безперервний список задач та ідей, що автоматично оновлюється та сортується за пріоритетністю чи іншими параметрами.

Колекції повинні дозволяти користувачам групувати задачі та плани за темами або проектами, забезпечуючи структуроване зберігання інформації. Це спрощує доступ до конкретних задач та покращує загальну організацію.

Інтеграція трекерів звичок дозволить користувачам відслідковувати свої щоденні, тижневі або місячні звички та їх прогрес. Це допоможе у формуванні нових корисних звичок та підтриманні мотивації.

Статистика повинна включати детальні звіти про виконані задачі, прогрес у проєктах та загальну продуктивність користувача. Це надасть цінні поради для вдосконалення процесу планування.

Рекомендації на основі аналізу даних про попередні задачі та продуктивність повинні допомагати користувачам оптимізувати свої плани та досягати кращих результатів. Додаток має пропонувати персоналізовані поради та пропозиції.

Декомпозиція планів дозволить користувачам розбивати великі проекти на дрібніші, легше керовані задачі. Це сприятиме кращому розумінню та виконанню великих цілей через покроковий підхід.

Командна робота має надавати функціональні можливості для створення загальних завдань для кількох людей, для змін в реальному часі, наприклад для коментарів або описів. Це допоможе у вирішенні командних задач та спільних проєктів.

Вдосконалення вже існуючих рішень означає впровадження нових функцій, покращення юзабіліті, підвищення стабільності та продуктивності

додатку, а також активне врахування зворотного зв’язку від користувачів для постійного розвитку продукту.

РОЗДІЛ 2: ОГЛЯД ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Мови програмування

2.1.1 Python

Python – це мова програмування, яка вважається зручною для початківців. Однією з головних причин, чому Python вважають легким для вивчення, є його простий синтаксис. Код Python легко читати та розуміти, що полегшує початківцям написання та налагодження коду. [2]

Ще однією причиною, чому ця мова вважається корисною, є її універсальність. Python можна використовувати для різноманітних програм, від веб-розробки Python до аналізу даних і машинного навчання. Це означає, що новачки можуть вибрати сферу інтересів і почати вивчати Python у будь якому контексті. Python також має багато ресурсів, таких як онлайн-підручники, відеокурси та інтерактивні платформи кодування. Ці ресурси містять покрокові інструкції та допомагають структуровано розвивати свої навички.

Python має велике та активне співтовариство розробників, яке відоме своєю гостинністю та готовністю допомогти. Знайти підтримку можна також в групах соціальних мереж і на онлайн-форумах, незалежно від того, якою мовою розмовляє розробник. Спільнота Python також дуже активна в проєктах з відкритим кодом. Python має багато відкритих бібліотек і фреймворків, які підтримуються спільнотою. Вивчаючи нову мову, важливо мати спільноту, де можна ставити запитання більш досвідченим людям і отримувати їхні відгуки.

Python є мовою високого рівня, тобто її легко читати та писати, з акцентом на абстракції деталей низького рівня та забезпеченні вищого рівня абстракції. Однак це також потужна мова, яку можна використовувати для складних проєктів. Однією з головних причин, чому Python такий гнучкий, є

його велика кількість бібліотек, модулів і пакетів. Ці бібліотеки містять готовий код, який можна легко інтегрувати в проєкт, заощаджуючи час і зусилля розробників. Крім того, Python можна використовувати для веб-розробки, аналізу даних, машинного навчання та наукових обчислень тощо. Гнучкість Python також полягає в його здатності працювати з іншими мовами. Він легко інтегрується з такими мовами, як C++ і Java, що дозволяє розробникам використовувати Python для конкретних завдань разом з іншими мовами для інших частин проєкту.

Велика бібліотека модулів і пакетів Python є однією з його найбільших переваг. Бібліотеки містять готовий код, який можна легко інтегрувати в проєкт, заощаджуючи час на розробку унікальних функцій у власному проєкті. Python має велику колекцію бібліотек, які постійно зростають, оновлюються і розвиваються відповідно до потреб розробників та користувачів. Популярні бібліотеки Python включають NumPy, яка підтримує числові та наукові обчислення, Pandas, яка використовується для аналізу й обробки даних, Matplotlib для візуалізації даних і Scikit-Learn для завдань машинного навчання. Python також має бібліотеки веб-розробки, такі як Django та Flask, які полегшують створення веб-додатків. Інші бібліотеки, такі як BeautifulSoup і Scrapy, використовуються для веб-збирання та вилучення даних. Бібліотеки Python також спрощують роботу з такими базами даних, як SQLite, MySQL і PostgreSQL.

Python можна вбудовувати, тобто його можна інтегрувати в інші мови програмування та програми. Це корисно для розробників, які хочуть додати функціональність Python до існуючого програмного забезпечення або створити власні програми, використовуючи Python як мову сценаріїв. Наприклад, Python можна вбудовувати в програми C++ за допомогою бібліотеки Boost. Це дозволяє використовувати сильні сторони Python під час використання будь якої мови програмування.

Python є масштабованим: він може обробляти великі обсяги даних і підтримувати розподілені обчислення. Бібліотеки та фреймворки Python, зокрема NumPy, Pandas і PySpark, дозволяють розробникам ефективно обробляти й аналізувати великі об’єми даних. Python підтримую паралельну обробку та розподілених обчислень через такі бібліотеки, як Dask і Apache Spark, також дозволяє масштабувати обробку даних на кількох машинах. Це робить Python популярним вибором для великих даних і програм машинного навчання.

Крім того, інтерактивна оболонка Python і блокнот Jupyter дозволяють тестувати фрагменти коду в режимі реального часу. Це допомагає швидко реагувати на зміні в коді. Ця можливість швидкого виводу інформації особливо корисна в таких сферах, як наука про дані, де проводяться експерименти та дослідження.

Зокрема, мова програмування Python легка та надійна, має значну кількість бібліотек, може підтримувати роботу з нейронними мережами та великими даними, а також має відкритий код, що значно спрощує розробку.

2.1.2 Java

Java була представлена компанією Sun Microsystems у 1995 році та стала видатною об’єктно-орієнтованою мовою програмування у сфері розробки програмного забезпечення. ЇЇ основною особливістю є її здатність працювати на кількох платформах завдяки підходу Write Once Run Anywhere (WORA). Цей підхід дозволяє розробникам написати код Java один раз і запустити його на будь-якій платформі за допомогою віртуальної машини Java (JVM).[3]

Протягом багатьох років Java зарекомендувала себе як зручна та портативна з автоматичним керуванням пам’яттю та надійним середовищем виконання. ЇЇ екосистема складається з трьох основних компонентів: Java Development Kit (JDK), Java Class Library та Java Virtual Machine (JVM). Ці компоненти відрізняють Java від інших рідних мов програмування та сприяють її широкому визнанню в індустрії програмного забезпечення.

Java має значну кількість переваг, що робить її популярним вибором для розробників, зокрема:

- легкий процес навчання;
- об'єктно-орієнтоване програмування;
- підвищена безпека;
- незалежний від платформи.

Java, мова програмування загального призначення, має простий синтаксис і є мовою програмування високого рівня, що робить її доступним вибором як для новачків, так і для досвідчених розробників. Її легкість та використання натуральної мови полегшують розуміння порівняно з іншими мовами зі складнішим кодом і мовою машинного рівня. Доступні численні онлайн-підручники, курси та документація, допомагають початківцям розпочати роботу з Java. Її широке використання в промисловості надає багато можливостей для практики та підвищення кваліфікації. Відданість і повторення роблять опанування Java досяжною метою для програмістів-початківців.

Однією з найбільших переваг Java є її підхід до об’єктно-орієнтованого програмування (ООП), який полегшує організацію, підтримку та масштабування коду. ООП дозволяє розробникам визначати типи та структури даних, а також функції, які до них застосовуються, таким чином, щоб імітувати об’єкти реального світу. Цей підхід полегшує керування великими базами коду, розбиваючи їх на менші сегменти. Навпаки, процедурне програмування вимагає від розробників слідувати послідовності інструкцій із використанням змінних і функцій, що може бути громіздким і важким для підтримки.

Java пропонує потужні функції безпеки, які роблять її популярним вибором для розробки безпечних програм. Ці функції включають:

- інкапсуляцію;
- поліморфізм;
- абстракцію.

Вбудований менеджер безпеки, дозволяє розробникам встановлювати правила доступу.

Інкапсуляція приховує деталі реалізації класу від інших об’єктів, тоді як успадкування дозволяє створювати нові класи з існуючих. З іншого боку, абстракція полягає в приховуванні деталей об’єкта від користувача. Ці правила разом із суворою політикою безпеки Java роблять її безпечною мовою для широкого кола програм.

Однією з найбільш цінних особливостей Java є її незалежність від платформи, що дає змогу розробникам написати код на своїй машині і запускати його на будь-якій платформі за допомогою віртуальної машини Java (JVM). Ця функція називається підходом Write Once Run Anywhere (WORA) і дозволяє запускати програми Java на будь-якій системі, де встановлено Java. Java також підтримує віддалені виклики методів, забезпечуючи безперебійний зв’язок між розподіленими програмами.

Віртуальна машина Java (JVM) служить абстрактним рівнем між кодом і обладнанням, дозволяючи програмам Java працювати на будь-якій платформі, яка підтримує JVM. Ця незалежність робить Java популярним вибором для:

- веб-додатків;
- мобільні додатків;
- хмарні обчислень;
- проєктів з використанням великих даних.

У підсумку, Java залишається однією з провідних мов програмування, що поєднує універсальність, простоту та потужність. Її багатий інструментарій, об'єктно-орієнтована архітектура та висока продуктивність забезпечують надійність та ефективність розробки, що робить її незамінним вибором для сучасних програмістів.

2.1.3 TypeScript

TypeScript – це надналаштування JavaScript, яке за бажанням можна перевасти та скомпілювати у звичайний JavaScript. Простіше кажучи, TypeScript технічно є JavaScript зі статичними типами, коли це потрібно. [4]

Отже, які можуть бути причини додавання статичної типізації до JavaScript? Легше зробити код більш читабельним, значно не порушуючи його. Навігація складними великими системами стає набагато простішою. Дослідження показують, що п’ятнадцять відсотків помилок, які допускають розробники на JavaScript, можна знайти та вирішити з використанням TypeScript.

Динамічне введення даних часто призводить до помилок. Це знижує як продуктивність самого програміста так і робить розробку значно повільнішою, через складність та вартість додавання нового коду. Відповідно, JavaScript стає поганим вибором для підприємств або великої кількості даних та коду.

Також варто зазначити що TypeScript доволі проста мова у порівнянні з іншими мовами, наприклад: CoffeeScript – додає синтаксичний цукор, а PureScript мало схожий на звичайний JavaScript і є складним для новачків.

В TypeScript не обов’язково задавати типи. Відповідно файли скомпільовані в JavaScript є дійсними у TypeScript, що є значною перевагою. Незважаючи на те, що компілятор може скаржитись на помилки типу у вихідних файлах, він працює з файлом JavaScript так само як і з файлом TypeScript. На якому б етапі не знаходилась розробка можна використати TypeScript, легко розширюючи його.

Також варто зазначити, що TypeScript був скомпільований в JavaScript. Це означає, що TypeScript можна використовувати скрізь, де можна було використати JavaScript, а саме: у інтерфейсах або у серверній частині.

JavaScript все ще є досить популярною мовою для реалізації інтерфейсних сценаріїв для програм і веб-сайтів. Відповідно, TypeScript використовується з тією ж метою, але він найкраще підходить для складних корпоративних проєктів через свою типізацію.

Типи – можливість відрізнити якісні програмні рішення від поганих перед їх запуском, описуючи в коді, як вони мають бути використаними. Типи можуть бути як простими, наприклад: Number і String. Та складними структурами, які ідеально моделюють складні дані.

Мови програмування можна розділити на дві категорії: статичні або динамічні. У статично типізованих мовах тип змінної має бути відомий під час компіляції. Коли оголошується змінна, компілятор має точно знати (або зробити висновок), чи є це числом, рядком чи логічним значенням. У динамічно типізованих мовах це не є обов'язковим правилом. Тип змінної стає відомий тільки після запуску програмного забезпечення. TypeScript може підтримувати статичні типи, а JavaScript – ні. Системи статичних типів дозволяють створювати власні складені типи. Це дозволяє інженерам більш детально висловити свої наміри.

Явні типи також роблять код більш документованим: вони гарантують, що змінні та функції виконують те, що вони повинні виконувати, і дозволяють комп’ютеру запам’ятовувати навколишній контекст. Тож TypeScript є хорошим вибором для програмного забезпечення, якщо розробник хоче отримати якісний типізований код.

2.2 Фреймворки

2.2.1 Spring Boot

Spring Boot – це широко використовуваний фреймворк з відкритим вихідним кодом для створення готових до запуску, автономних і продуктивних програм Java з мінімальними зусиллями. Розроблений компанією Pivotal Software (тепер це частина VMware), Spring Boot спрощує процес розробки додатків Java, надаючи набір угод, шаблонів і надійну екосистему бібліотек. [5]

Spring Boot заохочує розробників зосередитися на написанні бізнес-логіки, а не на написанні стандартного коду і конфігурації. Він забезпечує інтелектуальні параметри, які встановлені автоматично і легке налаштування, що скорочує час, витрачений на налаштування та розгортання програм.

Потужний інтерфейс командного рядка (CLI) дозволяє швидко створювати проєкти та компоненти.

Spring Boot добре підходить для створення архітектур на основі мікросервісів. Це полегшує розробку невеликих незалежних служб, які можна легко розгортати, масштабувати та підтримувати. Підтримка вбудованого веб-сервера (Tomcat, Jetty або Undertow) дозволяє легко зробити виконуваний архів JAR, що значно спрощує розгортання.

Spring Boot чудово інтегрується зі схожими проєктами Spring, такими як Spring Data, Spring Security і Spring Cloud, що полегшує створення складних програм, які потребують цих компонентів. Ця інтеграція забезпечує цілісний і послідовний процес розробки.

Spring Boot автоматично налаштовує багато аспектів програми на основі бібліотек і залежностей, які включаються у проєкт. Це мінімізує потребу в ручній конфігурації та зменшує ймовірність помилок конфігурації.

Spring Boot забезпечує підтримку вбудованих веб-серверів, таких як Tomcat, Jetty і Undertow. Відповідно, не потрібно налаштовувати зовнішній сервер, програма може працювати незалежно від вбудованого сервера, що робить її придатною для хмарного розгортання.

Spring Boot широко використовується для створення RESTful API і веб-служб. Вбудована підтримка Spring Web MVC і можливість генерувати відповіді JSON або XML спрощують розробку веб-API.

Як згадувалося раніше, Spring Boot є чудовим вибором для створення мікросервісів. Його мала вага та вбудована підтримка сервера роблять його придатним для розробки та розгортання невеликих незалежних служб.

Завдяки своїй надійності та масштабованості Spring Boot широко використовується в розробці корпоративних додатків, включаючи системи управління взаємовідносинами з клієнтами (CRM), системи управління людськими ресурсами (HRMS) і системи управління запасами.

Завдяки підтримці власного сервера та інтеграції з Spring Cloud Spring Boot добре підходить для створення хмарних додатків, які можна легко контейнеризувати та розгортати в клаудах, наприклад AWS або Azure.

Підводячи підсумок, Spring Boot робить розробку програм Java легшою, швидшою та ефективнішою. Переваги включають швидку розробку, підтримку мікросервісів і повну інтеграцію в екосистему Spring. Однак для початківців може знадобитися час для навчання, і налаштування може бути складним. Spring Boot найкраще підходить для веб-сервісів RESTful, мікросервісів, корпоративних програм і хмарних програм залежно від потреб вашого проєкту.

2.2.2 Angular

Angular – це популярний фреймворк для створення веб-додатків. Він використовується для створення застосунків (SPA), які забезпечують користувачам швидкий та ефективний досвід взаємодії з веб-сайтами. Angular надає розробникам можливість швидко та легко створювати високоякісні додатки, завдяки своїй потужній функціональності та вбудованим інструментам. [6]

Angular – це платформа з відкритим вихідним кодом, створена та підтримувана Google для створення високодинамічних інтерфейсів веб-додатків. Хоча цей фреймворк був запущений у вересні 2016 року, існує старіша версія, також відома як AngularJS. Звичайно, AngularJS був попередником нової версії, але конкретні технології неймовірно відрізняються.

AngularJS був випущений у жовтні 2010 року. Платформа стала дуже популярною відразу після запуску. Спеціальний фреймворк JavaScript дозволив програмістам відносно швидко з ним ознайомитися. Однак головною перевагою AngularJS було те, що він дозволяв розробникам створювати динамічні інтерфейси, які дозволяли користувачам взаємодіяти з веб-вмістом. На той час це була досить інноваційна функція, яка допомагала компаніям виділитися в онлайн-світі.

Однак AngularJS як фреймворк JavaScript був далекий від досконалості. Фреймворк був досить складним і вимагав від команди розробників великих зусиль, щоб створити інтерфейс, який добре виглядав і працював належним чином. Основною ідеєю фреймворку Angular було створення зручної для роботи платформи, яка відповідає сучасним стандартам програмування. Щоб втілити це в реальність, команда Google вирішила використовувати TypeScript, надмножину ECMAScript 6 (ES6), замість JavaScript. Оскільки зміни були настільки кардинальними, неможливо оновити веб-програму, створену за допомогою AngularJS, до Angular, тому потрібна повна міграція.

Компонентна архітектура є однією з найбільших переваг Angular перед його наступником. Простими словами, це означає, що кожна частина інтерфейсу користувача програми разом із базовою функціональністю утворює окремий компонент. Наприклад, інтерфейс Facebook включає такі компоненти, як Newsfeed, Friendliest, Chat, Stories тощо. Усі вони є самодостатніми, навіть якщо вони належать до одного веб-рішення. Їхні методи, API та структури можуть відрізнятися, але вони можуть легко «розмовляти» один з одним.

Компонентна архітектура є справжньою допомогою для програмістів. Це дозволяє оновлювати незалежні частини функціональності програми, не побоюючись, що зміни вплинуть на інші функції. Крім того, компоненти можна використовувати повторно. Команда розробників може написати компонент один раз, а потім використовувати його в багатьох подібних елементах у всій програмі. Нарешті, ця форма архітектури покращує читабельність коду та робить його набагато легшим для розуміння новим розробникам.

Спрощене кодування також є ключовою перевагою Angular перед іншими фреймворками. Як згадувалося раніше, Angular написаний на TypeScript, а не на JavaScript. Перевагою Angular є те, що він пропонує розширені можливості виявлення помилок. Вони дозволяють програмістам виявляти та виправляти типові помилки під час програмування. Це покращення особливо корисно для складних довгострокових проектів,

оскільки допомагає команді значно скоротити час, витрачений на перевірку якості та налагодження.

Однією з головних переваг використання Angular є його двосторонній синтаксис зв’язування даних. Це означає, що дані прив’язані до подання, тому будь-які зміни, внесені до даних у моделі, негайно відображаються в поданні і навпаки. Наприклад, коли користувач взаємодіє з користувальницьким інтерфейсом програми (переглядом), базова модель автоматично оновлюється. Це дозволяє програмістам працювати швидше, оскільки синхронізація не вимагає додаткового кодування. Двонаправлене зв’язування даних не тільки скорочує час веб-розробки, але також призводить до чистішого коду та покращення продуктивності програми.

Асинхронне програмування має вирішальне значення для сучасних веб-додатків, оскільки воно покращує швидкість роботи та запобігає збоям системи. Ця модель дозволяє запускати кілька команд одночасно. Отже, коли користувач викликає функцію, програма продовжує працювати та може виконувати інші дії. Для порівняння, модель синхронного програмування дозволяє програмам виконувати команди лише рядок за рядком. Якщо система використовує цей підхід, вона може розпочати наступну дію лише після завершення попередньої. Для реалізації асинхронного програмування Angular використовує бібліотеку RxJS.

Таким чином Angular є чудовим вибором фреймворку для написання відображення так як використовує компоненти, двосторонній синтаксис та підтримує асинхронне програмування.

2.3 Бази даних

2.3.1 PostgreSql

PostgreSQL – це надійна об’єктно-реляційна база даних з відкритим кодом. Це одна з найпоширеніших систем баз даних. За даними Statista, вона займає четверте місце за попитом. PostgreSQL розширює мову SQL і додає функціональність для ефективної обробки складних операцій із даними та їх

зберігання. Жодна компанія чи організація не контролює її, оскільки вона є відкритим кодом. Вона має понад 52 000 комітів, що підвищило її стабільність і можливості з моменту запуску. З цієї кількості комітів можна здогадатися, наскільки активно спільнота PostgreSQL сприяє її розвитку. З усіма фантастичними можливостями, використання PostgreSQL має багато переваг.[7]

Враховуючи переваги PostgreSQL і його список функцій, підтримку спільноти та виняткову продуктивність, не дивно, що більшість організацій у всьому світі вважають PostgreSQL найкращим вибором. Серед десятків переваг, які може запропонувати база даних PostgreSQL, нижче наведено основні переваги, про які варто згадати.

Якщо не використовувати транзакції, доведеться написати багато обробок помилок, що знижує час розробки. Однак не потрібно турбуватися про це, якщо у існує база даних, яка пропонує транзакції. PostgreSQL пропонує транзакційний DDL. Йдеться не лише про INSERT, UPDATE або DELETE для транзакції, а й про такі речі, як створення таблиці, автоматична таблиця, видалення таблиці тощо. Наприклад, якщо змінити таблицю, цю транзакцію буде зафіксовано негайно. Це важливо, якщо у реляційна програма значно складна. Крім того, дуже часто вносять зміни до програми одночасно зі зміною основної схеми бази даних. Крім того, якщо потрібно застосувати це для змін програми, усі зміни програми буде внесено в одній транзакції. Це одна з переваг PostgreSQL перед іншими базами даних.

Серед плюсів і мінусів PostgreSQL є одна з його головних переваг. PostgreSQL випущено за ліцензією PostgreSQL. Це дозвільна ліцензія з відкритим кодом, подібна до ліцензій BSD або MIT. Це означає, що вихідний код доступний і дозволяє користувачам використовувати, ділитися, змінювати та впроваджувати його за потреби для своїх проектів з обробки даних.

Понад 25 років PostgreSQL вдосконалювався, розширювався та впроваджувався інновації за підтримки спільноти організацій і окремих осіб.

Спеціальна спільнота PostgreSQL регулярно виправляє помилки та покращує загальну продуктивність, постійно покращуючи ефективність системи баз даних. Тому, зважуючи плюси і мінуси бази даних PostgreSQL, плюси завжди переважують мінуси.

Однією з переваг бази даних PostgresSQL є те, що вона здатна розширюватися. Отже, якщо потрібна додаткова функція в Postgres, можна додати її самостійно, що важко з іншими базами даних. Розширюваність – це ціла купа різних елементів Postgres. Можна використовувати Postgres для створення типів даних, функцій, мов і всіляких модифікацій.

Сам Postgres має властивості, пов’язані з безпекою. Крім того, він має розширення, які дозволяють підвищити безпеку. PostgreSQL отримав міжнародне визнання за свою здатність забезпечувати вашу безпеку. Він забезпечує безпеку налаштування та безпеку програм. Що стосується параметрів безпеки, якщо вам потрібно заблокувати вашу систему бази даних, вона надає конфігурації рівня операційної системи, які ви можете налаштувати для блокування вашої бази даних. З точки зору безпеки програми, він надає користувачам привілеї, розділяючи облікові записи на лише для читання, читання/запису або інші дії залежно від категорії.

2.3.2 Hibernate

Hibernate – це платформа ORM (Object-Relational Mapping) з відкритим кодом для програм Java. Він спрощує взаємодію з базами даних, відображаючи класи Java у таблицях баз даних, і дозволяє розробникам маніпулювати даними, використовуючи концепції об’єктно-орієнтованого програмування високого рівня замість складних запитів SQL. Hibernate був розроблений Гевіном Кінгом у 2001 році та став одним із найпопулярніших ORM-рішень на Java. Він пропонує численні функції для ефективного доступу та керування базами даних. [8]

Java Database Connectivity (JDBC) — це стандартизований API, наданий Java для підключення та запитів до баз даних. Він визначає, те як програмне

забезпечення може отримати доступ до певної бази даних, і надає відповідні методи для запитів (SELECT) і оновлення даних (UPDATE, DELETE). Драйвери JDBC дозволяють додаткам Java спілкуватися з різними базами даних і, отже, мають вирішальне значення для взаємодії з базами даних у додатках Java.

Хоча JDBC служить основою для доступу до бази даних у Java, він створює кілька перешкод для розробників. Давайте заглибимося в ці проблеми та подивимося, як Hibernate їх вирішує:

Написання складних SQL-запитів може бути виснажливим, особливо для складних операцій, пов’язаних із об’єднаннями, агрегаціями або фільтруванням. Навіть незначні друкарські або синтаксичні помилки в інструкціях SQL можуть призвести до виняткових ситуацій під час виконання, які вимагають налагодження та потенційно зупинять прогрес програми.

Код JDBC тісно пов’язаний зі специфічною структурою та синтаксисом цільової бази даних, що робить його негнучким під час переходу на іншу систему баз даних. Зміна таблиць, стовпців або типів даних часто вимагає значних змін у коді всієї програми.

JDBC вимагає від розробників вручну обробляти помилки та винятки, пов’язані з базою даних. Це включає виявлення потенційних проблем, таких як помилки підключення, невідповідності типів даних або порушення обмежень. Написання коду обробки помилок вручну повторюється і додає стандартну логіку до програми.

Переклад між об’єктно-орієнтованими концепціями в Java та моделлю реляційної бази даних може зайняти час і зусилля. JDBC не надає вбудованих механізмів для відображення об’єктів Java у таблиці бази даних і навпаки. Розробникам доводиться писати власну логіку для перетворення даних між цими двома парадигмами, що ускладнює розробку.

Hibernate автоматично генерує необхідні SQL-запити на основі таких операцій об’єктів, як створення, отримання, оновлення або видалення даних. Це усуває необхідність ручного кодування SQL, скорочує час розробки та мінімізує можливі помилки.

Hibernate відображає класи Java у таблиці бази даних, а об’єкти – у рядки. Це дозволяє розробникам працювати з об’єктами, які вони розуміють, замість того, щоб писати чистий SQL, спрощуючи логіку зберігання даних.

Програми Hibernate менш залежать від конкретних деталей основної схеми бази даних. Вони взаємодіють з різними базами даних через стандартні інтерфейси та конфігурації, що покращує переносимість коду.

Hibernate надає механізми обробки винятків для належної обробки помилок бази даних. Він створює певні винятки, які розробники можуть уловлювати та відповідним чином обробляти, покращуючи зручність обслуговування та надійність коду.

Вирішуючи ці проблеми, Hibernate забезпечує більш ефективний і зручний для розробників підхід до взаємодії з базами даних у програмах Java.

Обмеження JDBC вимагають більш ефективного способу взаємодії з базами даних. Hibernate вирішує ці проблеми, надаючи високорівневу структуру, яка абстрагує основні взаємодії з базою даних. Зіставляючи об’єкти Java у таблиці бази даних, Hibernate усуває потребу у великій кількості шаблонного коду, зменшує кількість помилок і спрощує операції з базою даних. Він також пропонує портативність бази даних і підтримує складне керування запитами через HQL (Hibernate Query Language).

2.4 Мікросервіси та зв’язок між ними

Мікросервіси – це певна архітектура розробки програмного забезпечення, яка передбачає побудову додатків як набір незалежних, дрібнозернистих сервісів, що виконують окремі функції і можуть взаємодіяти один з одним через чітко визначені інтерфейси, зазвичай за допомогою

HTTP/REST API або інших протоколів. Кожен мікросервіс відповідає за конкретну бізнес-функцію і розгортається, тестується і масштабується окремо. [13]

Мікросервіси дозволяють розгортати та масштабувати кожен сервіс окремо, що покращує використання ресурсів і зменшує ризики збоїв. Це особливо корисно для великих систем, де різні частини додатка мають різні вимоги до продуктивності.

Оскільки кожен мікросервіс розробляється окремо, команди можуть працювати незалежно одна від одної, використовуючи різні мови програмування, що прискорює процес розробки і дозволяє експериментувати з новими технологіями.

Збої в одному мікросервісі не обов’язково впливають на всю систему. Це зменшує ризик повного відмовлення додатку та полегшує відновлення після збоїв.

Мікросервіси зазвичай мають менший обсяг коду, що спрощує їх тестування, налагодження та підтримку. Крім того, модульність мікросервісів сприяє повторному використанню коду.

Нові функції можуть бути додані або змінені без впливу на інші частини системи, що робить процес оновлення більш безпечним та передбачуваним.

REST (Representational State Transfer) є популярним стилем архітектури для створення API, який зазвичай розробники використовують для забезпечення взаємодії між мікросервісами. [12]

REST має кілька основних принципів для роботи: сервер надає ресурси, а клієнт отримує доступ до цих ресурсів. Це дозволяє розділити обов’язки між фронтендом і бекендом. Всі запити від клієнта до сервера повинні містити всю інформацію, необхідну для виконання запиту. Сервер не повинен зберігати стан між конкретними запитами. Відповіді можуть бути позначені як кешовані, що дозволяє клієнтам зберігати копії відповідей і повторно використовувати їх

без повторного звернення до сервера. Використання стандартних методів HTTP (CRUD) для обробки даних, робить API більш простими і зрозумілими. Всі об’єкти, які надаються сервісами, розглядаються як ресурси, які ідентифікуються унікальними URL.

2.5 Бібліотеки

2.5.1 Apache Spark

Apache Spark – це розподілена обчислювальна система, яка часто згадується як фреймворк або платформа, але її також можна розглядати як бібліотеку для обробки великих даних. Spark забезпечує API для роботи з великими даними та може використовуватись різними мовами програмування, включаючи Java, Scala, Python. Це дозволяє Spark інтегруватися в програми як бібліотека, що надає можливості для обробки даних, але з акцентом на розподілені обчислення. [19]

Spark може виконувати обчислення в пам'яті, що значно прискорює обробку даних, якщо порівнювати з дисковими обчисленнями, наприклад, в Hadoop MapReduce. Spark надає багатий API для обробки даних, який підтримує кілька мов програмування, що робить Spark корисним для розробників. Spark підтримує різноманітні види обробки даних, включаючи пакетну обробку, обробку потоків (Streaming), інтерактивні запити (Spark SQL) і машинне навчання (MLlib). Spark може працювати на кластерах, що складаються з тисячі вузлів, обробляючи петабайти даних, що дозволяє масштабувати обчислення залежно від потреб. Spark забезпечує розподілені обчислення за допомогою RDD (Resilient Distributed Datasets), які автоматично розподіляють дані по кластеру і забезпечують стійкість до збоїв.

2.5.2 TensorFlow

TensorFlow – це потужний фреймворк для нейронних мереж та машинного навчання, але його також часто називають бібліотекою. TensorFlow надає набір інструментів і API для створення, тренування та розгортання моделей машинного навчання. Як бібліотека, TensorFlow може бути

інтегрована в різні додатки для вирішення задач машинного навчання, але його функціональні можливості і екосистема роблять його чимось більшим, ніж просто бібліотека.[9]

TensorFlow підтримує великий спектр шаблонів моделей і алгоритмів машинного навчання, наприклад: лінійні моделі, рекурентні мережі (RNN) або згорткові мережі (CNN). Фреймворк використовує апаратне прискорення з використанням GPU і TPU, що значно прискорює обчислення, особливо для задач глибокого навчання. TensorFlow дозволяє масштабувати обчислення від одного пристрою до кластерів з тисячами машин, що робить його придатним для обробки великих обсягів даних. TensorFlow має велику екосистему інструментів, бібліотек і ресурсів, що підтримуються Google і спільнотою, включаючи TensorFlow Hub, TensorFlow Lite, TensorFlow.js і інші. TensorFlow забезпечує інструменти для розгортання моделей у виробництві, такі як TensorFlow Serving для розгортання моделей як веб-сервісів.

РОЗДІЛ 3: ОГЛЯД ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Загальний огляд програмного забезпечення та запропоновані покращення

У рамках дисертаційного дослідження було проведено аналіз існуючих систем для управління завданнями та удосконалено програмне забезпечення, що покликане усунути виявлені недоліки та запровадити нові функціональні можливості для поліпшення користувацького досвіду завдяки використанню алгоритмів машинного навчання.

Аутентифікація та реєстрація користувачів Користувач взаємодіє з системою через авторизацію, для якої використовується електронна пошта та пароль. Реєстраційний процес включає введення логіна, пароля та імені користувача. Програмне забезпечення автоматично перевіряє валідність даних для забезпечення безпеки користувачів.

Після успішної авторизації користувач отримує доступ до сторінки «Plans», де відображаються задачі на поточний тиждень. Інтерфейс дозволяє створювати нові завдання, редагувати та видаляти існуючі, а також позначати їх відповідно до статусу: виконано, у процесі, неактуально або перенесено на інший день.

При удосконаленні програмного забезпечення було реалізовано можливість додавати до кожного плану дедлайн, точний час виконання (задля подальшого використання в алгоритмі розподілення завдань та складання розкладу) та пріоритетність завдання. Це забезпечує краще управління ресурсами та дає можливість чіткіше структурувати робочий день, а також дозволяє автоматично створювати розклад завдяки алгоритму розподілу. Система аналізує введені задачі та автоматично генерує оптимізований розклад, зважаючи на пріоритетність та тривалість завдань.

Додатково, користувач може декомпозувати завдання на підзавдання, що дозволяє детальніше планувати процес виконання великих проєктів. Система також дозволяє експортувати декомпозицію у файл для подальшого аналізу.

Користувачам доступний трекер звичок, що відображається у вигляді календаря. Це дозволяє відзначати дні, коли було виконано певну звичку, з можливістю аналізу прогресу.

Нотатки Розділ «Notes» дозволяє користувачам створювати, редагувати та видаляти нотатки, які можуть бути відсортовані за категоріями. Це зручний спосіб для збереження важливих заміток та ідей.

Система підтримує функціонал «Collections», що дозволяє користувачам створювати та управляти колекціями (списками) різних типів: книг, фільмів, місць для відвідування тощо. Користувачі можуть сортувати та шукати колекції за назвою, а також редагувати їх або видаляти.

З урахуванням потреб користувачів було додано базові колекції для фільмів та книг, що значно спрощує створення нових списків. Крім того, інтегровано систему рекомендацій, яка пропонує нові фільми та книги на основі вподобань користувача. Це не лише економить час, але й покращує загальний досвід взаємодії з програмним забезпеченням.

Важливим елементом є сторінка статистики, де користувач може переглядати графіки виконаних та невиконаних задач, успіхи в трекері звичок та створені колекції. Дані можна аналізувати за місяць або рік, що дає можливість виявити тренди та оптимізувати продуктивність.

Було додано розширений аналітичний інтерфейс, що дозволяє отримувати прогнози на основі поточних тенденцій у завданнях та звичках. Це сприяє кращому плануванню та підвищенню продуктивності користувачів.

Нова функціональна можливість – можливість спільної роботи над проєктами. Користувачі можуть створювати групи, де учасники можуть додавати, редагувати або переглядати спільні плани та завдання.

Розроблене програмне забезпечення є потужним інструментом для управління завданнями, колекціями, звичками та командною роботою. Запропоновані покращення дозволяють підвищити ефективність користувача, оптимізувати робочі процеси та забезпечити персоналізоване середовище для досягнення цілей.

3.2 Архітектура програмного забезпечення

3.2.1 Загальний опис багаторівневої архітектури

Для написання програмного забезпечення, а саме веб-застосунку для планування робочого часу, було вирішено використовувати багаторівневу архітектуру.

Багаторівнева архітектура (multi-tier architecture) – це модель побудови програмного забезпечення, в якій система розділяється на кілька рівнів (шарів), кожен з яких виконує конкретні завдання і відповідає за певну функціональність. Найпоширеніший приклад – тришарова архітектура, яка складається з клієнтського рівня (фронтенду), сервісного рівня (бекенд) та рівня баз даних.

Є багато переваг застосовувати клієнт-серверну архітектуру. Кожен рівень системи виконує свої унікальні функції, що полегшує розробку, підтримку та тестування. Наприклад, зміни на рівні інтерфейсу не впливають на рівень логіки або бази даних. Така архітектура дозволяє чітко розподілити обов’язки між різними частинами системи, що спрощує масштабування та обслуговування. Кожен рівень може масштабуватися незалежно. Наприклад, якщо навантаження на серверний рівень збільшується, можна додати більше серверів без необхідності змінювати клієнтську частину чи базу даних. Це дозволяє краще розподілити ресурси залежно від поточного навантаження та потреб системи. Завдяки розподілу функціональних ролей між різними рівнями система може паралельно обробляти запити на різних рівнях, що підвищує її продуктивність. Оскільки рівень даних відокремлений від клієнтської частини, можна забезпечити кращий контроль доступу до даних

користувачів, а також зменшити ризик несанкціонованого доступу. Легко додавати або змінювати функціонал на окремих рівнях, не зачіпаючи інші частини системи. Це дозволяє швидко реагувати на зміну вимог або виправляти помилки, не впливаючи на загальну стабільність.

3.2.2 Опис архітектури програмного забезпечення для планування

Архітектура програмного забезпечення, включає фронтенд реалізований на Angular і TypeScript, серверну частину реалізовану на Java з використанням реляційної бази даних PostgreSQL та окремий модуль бекенду реалізований на Python для роботи з нейронними мережами, та являє собою багаторівневу систему з чітким розподілом функціональних ролей. Така архітектура забезпечує модульність, масштабованість і розширюваність системи, що особливо важливо для сучасних програм з великим обсягом даних та високими вимогами до продуктивності.

Кожен компонент системи працює на окремих серверах, що дозволяє досягти гнучкості. Завдяки окремим серверам для різних компонентів система може розширюватися без значних змін в архітектурі. Це дозволяє легко додавати нові сервери за потреби розширення функціональних можливостей програмного забезпечення. Фронтенд, бекенд і база даних працюють незалежно, що підвищує продуктивність і забезпечує чітку організацію робочих процесів. Використання різних серверів для роботи з Java і Python дозволяє обробляти складні операції паралельно, оптимізуючи час виконання та підвищуючи продуктивність нейронних мереж.

Серверна частина, побудована на Java, є основою бізнес-логіки системи та забезпечує взаємодію з реляційною базою даних PostgreSQL. Java має високу надійністю, стабільність і можливість обробляти великий обсяг запитів. Java є однією із найпопулярніших мов програмування для корпоративних застосунків через свою надійність і високий рівень безпеки. Система легко масштабована завдяки можливостям Java і PostgreSQL, що дозволяє обробляти великий обсяг транзакцій і запитів у міру зростання бази

даних і користувацького навантаження. PostgreSQL є ACID-сумісною базою даних, що забезпечує надійне управління транзакціями, а також підтримує складні запити і індексацію, що підвищує швидкість доступу до даних.

Окремий сервер реалізований на Python, виконує специфічні завдання, пов'язані з обробкою даних і машинним навчанням, зокрема нейронними мережами. Python є однією з провідних мов програмування для роботи зі штучним інтелектом (ШІ) завдяки великій кількості бібліотек, таких як TensorFlow, PyTorch та інші. Python надає потужні інструменти для швидкої розробки та тренування нейронних мереж, що дозволяє системі адаптуватися під потреби користувачів і вдосконалюватися на основі аналізу даних. Бекенд на Python інтегрується з сервісами високопродуктивних обчислень, що дозволяє швидко обробляти великі обсяги даних. Python дозволяє ефективно обробляти і передавати дані між різними модулями системи, завдяки чому машинне навчання та інші алгоритми можуть взаємодіяти з бізнес-логікою, реалізованою на Java.

Фронтенд, побудований на Angular, реалізований на TypeScript, слугує головною точкою взаємодії користувача з системою. Angular є сучасним фреймворком для створення застосунків, що забезпечує високу продуктивність і користувацький досвід. Angular надає можливість створення окремих модулів і компонентів, що полегшує підтримку та розвиток системи. Використання TypeScript забезпечує жорстку типізацію, що дозволяє виявляти помилки на етапі компіляції, підвищуючи стабільність коду. Angular має вбудовану підтримку двостороннього зв'язування даних, що дозволяє синхронізувати зміни між моделями та представленням у реальному часі. Angular забезпечує ефективну роботу з REST API, що полегшує взаємодію з бекендом.

Комунікація між фронтендом і бекендом, а також між різними бекенд-сервісами (Java та Python) відбувається через API. Використання REST API дозволяє легко передавати дані між різними компонентами системи, незалежно від мови програмування чи платформи. REST API забезпечує стандартизацію

взаємодії, що спрощує розширення функціональності системи. Нові функції можуть бути додані без суттєвого переписування існуючих модулів завдяки стандартизованим API. Фронтенд залишається відокремленим від деталей реалізації бекенду, що дозволяє вносити зміни в бізнес-логіку без впливу на інтерфейс.

Запропонована багаторівнева архітектура системи з використанням Angular, Java, Python і PostgreSQL забезпечує високу продуктивність, масштабованість та гнучкість (рис. . Розподіл обчислювальних ресурсів між серверами та інтеграція з нейронними мережами через окремий Python-бекенд роблять систему гнучкою для завдань, які вимагають аналізу даних та автоматизації процесів. Така архітектура дозволяє легко адаптуватися до нових вимог і забезпечує стабільне функціонування навіть під значними навантаженнями.

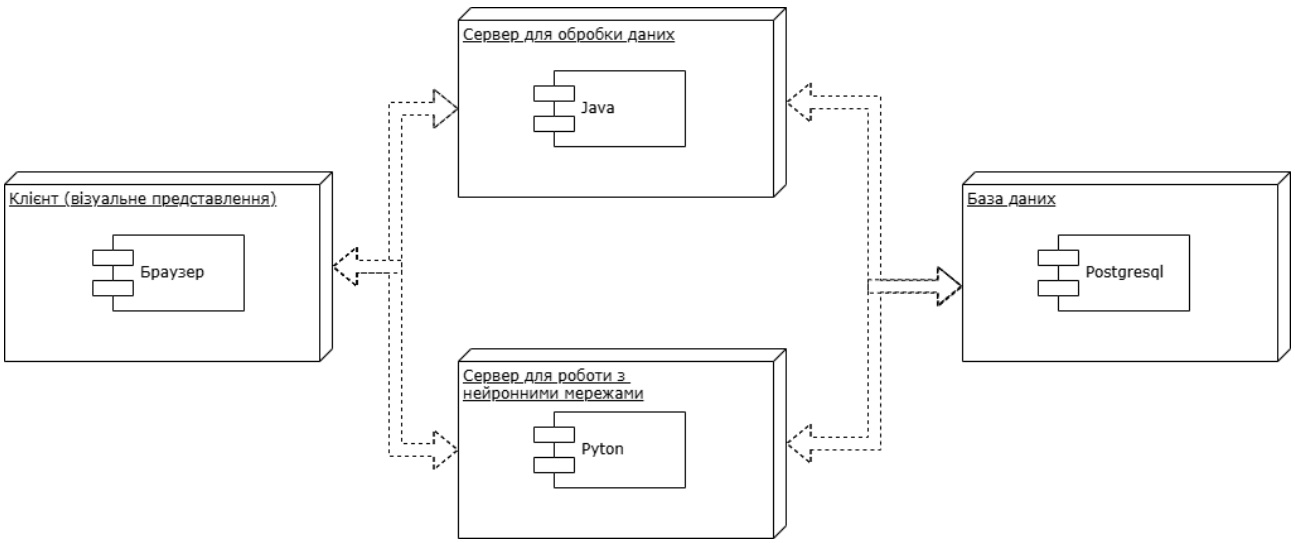


Рисунок 3. Архітектура ПЗ

3.3 Опис бази даних

База даних, представлена на рисунку 4, структурована таким чином, щоб підтримувати функціональність програмного забезпечення, зосередженого на плануванні, створенні колекцій та командній роботі. Особливу увагу варто звернути на таблиці, що стосуються планів, колекцій та команд, оскільки вони є ключовими елементами системи.

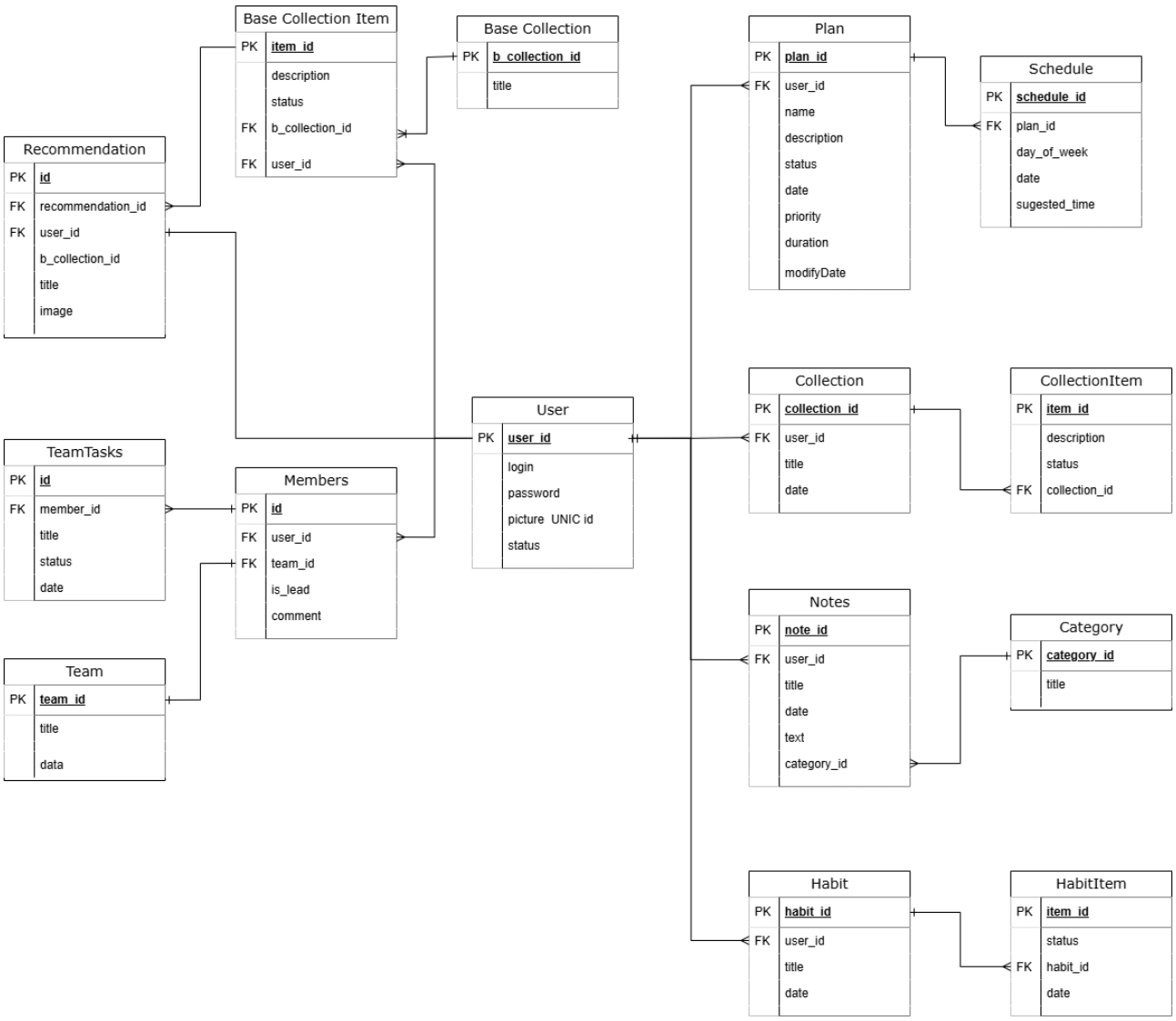


Рисунок 4. Схема бази даних

1. Таблиця Plan

Таблиця зберігає інформацію про плани користувача. Кожен план містить опис завдання, його статус, пріоритетність, тривалість та дату останньої модифікації.

Основні поля:

- plan_id (Primary Key) – унікальний ідентифікатор плану;

- user_id (Foreign Key) – посилання на користувача, якому належить план;
- name – назва плану;
- description – детальний опис завдання;
- status – стан виконання плану (виконано, в процесі тощо);
- date – дата дедлайну;
- priority – рівень пріоритету завдання;
- duration – тривалість завдання;
- creatingDate – дата створення плану;
- modifyDate – дата останньої модифікації плану.

Зв'язки:

Кожен план належить певному користувачеві через зв'язок із таблицею User. Таким чином, користувач може мати декілька планів.

План також пов'язаний з командою через таблицю TeamUser, що дозволяє користувачам співпрацювати над виконанням спільних завдань.

2. Таблиця Collection

Ця таблиця зберігає інформацію про колекції користувача. Колекції можуть містити різноманітні елементи, наприклад, книги, фільми чи інші об'єкти, які користувач хоче відслідковувати.

Основні поля:

- collection_id (Primary Key) – унікальний ідентифікатор колекції;
- user_id (Foreign Key) – посилання на користувача, який створив колекцію;
- title – назва колекції;
- date – дата створення колекції.

Зв'язки:

Кожна колекція належить певному користувачеві, через зв'язок із таблицею User.

Колекція містить елементи, представлені в таблиці CollectionItem, де зберігаються окремі об'єкти колекції (наприклад, книги або фільми).

3. Таблиця Base Collection

Базова колекція являє собою набір типових колекцій, які система пропонує користувачам за замовчуванням. Це може бути колекція книг, фільмів або інших категорій, яка вже має попередньо визначений набір елементів.

Основні поля:

- b_collection_id (Primary Key) – унікальний ідентифікатор базової колекції;
- title – назва базової колекції («Фільми», «Книги»).

Зв'язки:

Базові колекції мають зв'язок із таблицею Base Collection Item, де зберігаються елементи, що входять до цих колекцій. Користувач може використовувати базові колекції як шаблон для власних колекцій.

4. Таблиця Team

Таблиця зберігає інформацію про команди користувачів. Користувачі можуть об'єднуватися в команди для спільної роботи над планами або завданнями.

Основні поля:

- team_id (Primary Key) – унікальний ідентифікатор команди;
- title – назва команди;
- date – дата створення команди.

Зв'язки:

Кожна команда може мати кілька користувачів, що відображено в таблиці Member. Це дозволяє зберігати інформацію про склад команди та роль кожного користувача у спільній роботі над завданнями.

5. Таблиця Member

Таблиця реалізує зв'язок між користувачами та командами, а також зберігає додаткову інформацію про участь користувачів у командах.

Основні поля:

- id (Primary Key) – унікальний ідентифікатор запису;
- user_id (Foreign Key) – посилання на користувача, який є членом команди;
- team_id (Foreign Key) – посилання на команду, до якої належить користувач;
- is_lid – позначає, чи є користувач лідером команди;
- plan_id (Foreign Key) – посилання на план, над яким працює команда;
- comment – коментар щодо участі користувача у виконанні плану.

3.4 Опис алгоритмів

3.4.1 Alternating Least Squares

Алгоритм Alternating Least Squares (ALS) – це метод для розв'язання задачі факторизації матриці, часто використовуваний у системах рекомендацій. Його мета полягає в заповнені пропусків в матриці користувачів і предметів, шляхом розкладу цієї матриці на дві матриці меншого розміру, які потім використовуються для прогнозування відсутніх значень.

У контексті систем рекомендацій, маємо матрицю взаємодії R, де рядки – це користувачі, стовпці – це предмети (фільми, товари і т.д.), а елементи

матриці r_{ij} – це оцінка користувачем і предмета j. Завдання полягає в передбаченні оцінки, які користувачі ще не надали.

ALS розв'язує це завдання через факторизацію матриці. Матрицю оцінок R намагаються розкласти на дві матриці:

Матриця користувачів U – кожен рядок цієї матриці відповідає вектору характеристик користувача.

Матриця предметів V – кожен рядок цієї матриці відповідає вектору характеристик предмета.

Мета алгоритму – знайти такі U і V, що їхній добуток $U \times V^T$якнайкраще наближав оригінальну матрицю R, тобто мінімізував похибку між фактичними й передбаченими оцінками.

Основна ідея ALS полягає в тому, що ми чергуємо оновлення U і V за методом найменших квадратів (Least Squares). Спочатку фіксуємо V і розв'язуємо рівняння для U, що мінімізує функцію втрат. Оскільки задача для U є квадратичною, її можна ефективно розв'язати за допомогою регресії. Потім фіксуємо U і оновлюємо V, мінімізуючи функцію втрат відносно V. Це чергування триває, доки модель не збіжиться до прийнятного результату, коли прогнозовані оцінки максимально наближатимуться до фактичних.

ALS добре підходить для великих розріджених матриць, де велика кількість елементів відсутня, що є типовим для реальних систем рекомендацій. Оскільки оновлення кожного користувача і предмета є незалежним, алгоритм легко паралелізується, що дозволяє застосовувати його до великих наборів даних із використанням розподілених обчислювальних платформ, наприклад Apache Spark. ALS може легко обробляти різні типи даних взаємодії, завдяки чому він може використовуватись у різних сценаріях – від системи рекомендацій до електронної комерції.

Стандартна версія ALS призначена для роботи з експліцитними оцінками (тобто, коли користувачі явно ставлять оцінки). Однак у багатьох реальних застосуваннях, таких як електронна комерція або потокові сервіси, явних оцінок може бути мало. У цьому випадку корисно працювати з імпліцитними відгуками (наприклад, переглядами, кліками, покупками). Існує модифікація алгоритму – Implicit ALS, яка розглядає сам факт взаємодії (чи купував користувач товар, чи переглядав контент) як сигнал, і зважує цей сигнал залежно від частоти взаємодій. Стандартна регуляризація L2 у ALS може бути доповнена адаптивною регуляризацією, де ваги регуляризації змінюються залежно від індивідуальних користувачів або предметів. Це дозволяє краще адаптувати модель до гетерогенних даних, де деякі користувачі можуть бути більш активними або надавати більш релевантні дані. Введення додаткових факторів, таких як часовий контекст або інформація про взаємодії користувачів з предметами, може значно покращити точність рекомендацій. Це дозволяє алгоритму враховувати не лише історію взаємодій, але й зовнішні фактори такі як: час доби, день тижня, сезонність тощо. У деяких сценаріях різні оцінки або взаємодії можуть мати різну важливість. Можна застосувати різні ваги для різних типів взаємодій або різних груп користувачів, щоб покращити якість рекомендацій. Наприклад, взаємодії активних користувачів можуть бути важливішими для тренування моделі. Щоб уникнути перенавчання, можна використовувати раннє зупинення (early stopping), коли тренування моделі припиняється, якщо після певної кількості ітерацій не відбувається значного покращення точності. Алгоритм ALS є одним із найефективніших методів для факторизації матриць і часто використовується в системах рекомендацій. Завдяки покращенням, таким як робота з імпліцитними відгуками, адаптивна регуляризація, контекстуальне зважування та паралелізація, ALS залишається одним із провідних інструментів для побудови рекомендаційних систем у великих і розріджених наборах даних.

3.4.2 Long Short-Term Memory

Алгоритм Long Short-Term Memory (LSTM) є варіацією рекурентної нейронної мережі (RNN), яка дає змогу ефективно працювати з послідовними даними, зберігаючи важливу інформацію про попередні події та забезпечуючи гнучкий підхід до оптимізації управління завданнями. У контексті автоматизації розподілу завдань із врахуванням дедлайнів, часу виконання, пріоритетів та минулих дій користувача, LSTM може бути використаний для прогнозування оптимального часу для виконання завдань, а також для адаптації розкладу залежно від змін у статусі виконання завдань.

LSTM працює із серіями послідовних даних. У нашому випадку, вхідними даними для кожного завдання будуть:

- Дедлайн та точний час виконання;
- статус виконання (завершено, у процесі, перенесено або неактуальне);
- пріоритетність завдання (високий, середній, низький);
- попередні плани користувача (коли і як вони виконувались);
- частково виконані завдання (завдання, що виконані на певний відсоток);
- індивідуальні звички користувача щодо виконання завдань (історичні дані про зміну статусу завдання).

LSTM має комірку пам'яті, яка дозволяє моделі запам'ятовувати важливі характеристики попередніх станів. Це ключова особливість, яка вирізняє LSTM від звичайної RNN. На кожному етапі послідовності, LSTM вирішує, які частини інформації передати до наступних етапів (користуючись вхідними даними та історією виконання завдань). Інформація, яка не є актуальною для поточного стану (наприклад, неактуальні або завершені завдання), може бути «забута».

У LSTM використовуються три ключові елементи, які допомагають контролювати потік інформації:

- Input gate – контролюють, яка нова інформація додається до комірки пам'яті;
- forget gate – вирішують, яка частина інформації з попереднього стану може бути «забута»;
- output gate – контролюють, яка частина збереженої інформації використовується для прогнозування наступного стану.

У контексті планування та розподілу завдань, Forget gate вирішують, чи має бути врахована історія виконання попередніх планів. Наприклад, якщо завдання вже виконане або неактуальне, ця інформація може бути відфільтрована. Input gate дозволяють моделі оновлювати свою пам'ять на основі нових завдань та даних. Наприклад, при додаванні нових планів, які мають високий пріоритет або короткі терміни виконання, модель оновлює свої уявлення про поточний стан завдань, що дозволяє краще організувати розклад. Output gate відповідають за те, щоб забезпечити правильний прогноз. Виходячи з інформації в комірці пам'яті, LSTM може запропонувати оптимальний час для виконання завдань, з урахуванням всіх раніше зібраних даних про плани користувача.

Приклад роботи LSTM у контексті розподілу завдань:

- Користувач вводить новий план із зазначенням дедлайну, тривалості виконання та пріоритетності завдання;
- модель LSTM аналізує цей план разом із історичними даними про виконання попередніх завдань;
- на основі поточних та минулих даних, система генерує пропозиції щодо оптимального часу для виконання завдання, враховуючи такі фактори: завдання з високим пріоритетом можуть бути заплановані на ранні години дня або на періоди підвищеної продуктивності користувача (на основі його історичних даних); завдання, які частково виконані, можуть бути заплановані на найближчий час для завершення; завдання, які користувач

часто переносить, можуть бути запропоновані для відкладення або зниження пріоритету.

LSTM може враховувати персональні особливості користувача, такі як періоди найбільшої активності або продуктивності. Це дозволяє адаптувати розклад під конкретного користувача, враховуючи його звички та історію виконання завдань. Модель може враховувати статус завдань, які вже частково виконані, і надавати їм пріоритет у розкладі для завершення. Це дозволяє уникнути перенесення таких завдань на пізніший час і забезпечити їх своєчасне завершення. Завдяки LSTM, система може ефективно розподіляти завдання по часу, оптимізуючи використання часу та ресурсів користувача. Завдання з високим пріоритетом і коротким дедлайном отримують вищий пріоритет у розкладі, що дозволяє уникнути перевантаження або недосягнення дедлайнів.

3.4.3 Визначення якісних моделей нейронних мереж в програмному забезпеченні

Для отримання якісного програмного забезпечення, що надає точність більше 90% потрібно провести аналіз моделей нейронних мереж та визначити якісну. Порівняємо дві архітектури нейронних мереж на основі LSTM, що використовуються для класифікаційних завдань. Метою цього дослідження є оцінка ефективності та стабільності більш простих моделей порівняно з більш складними моделями, включаючи додаткові механізми для покращення навчання.

Перша модель (див. рис. 5), яка називається build_simple_lstm_model, складається з одного шару LSTM з 64 нейронами та активацією на основі вхідних даних. Після шару LSTM було додано 7 вихідних даних, що відповідають кількості класів, і 1 щільний шар з функцією активації softmax для обчислення ймовірностей класів. Модель скомпільована за допомогою оптимізатора adam та функції втрат categorical_crossentropy.

Softmax – це функція активації, яка зазвичай використовується у вихідному шарі нейронної мережі для задач багатокласової класифікації. Її основне завдання – перетворити вихідні значення нейронної мережі на ймовірності, які сумуються до одиниці.

Adam (Adaptive Moment Estimation) – це оптимізатор, який найчастіше використовується для тренування нейронних мереж завдяки своїй швидкості та ефективності. Adam комбінує властивості двох інших методів оптимізації: AdaGrad (адаптивна швидкість навчання) та RMSProp (корекція градієнтів).

Categorical Crossentropy (категорійна крос-ентропія) – це функція втрат для багатокласової класифікації. Її мета – оцінити, наскільки прогнозовані ймовірності класів (згенеровані функцією softmax) відрізняються від справжніх міток.

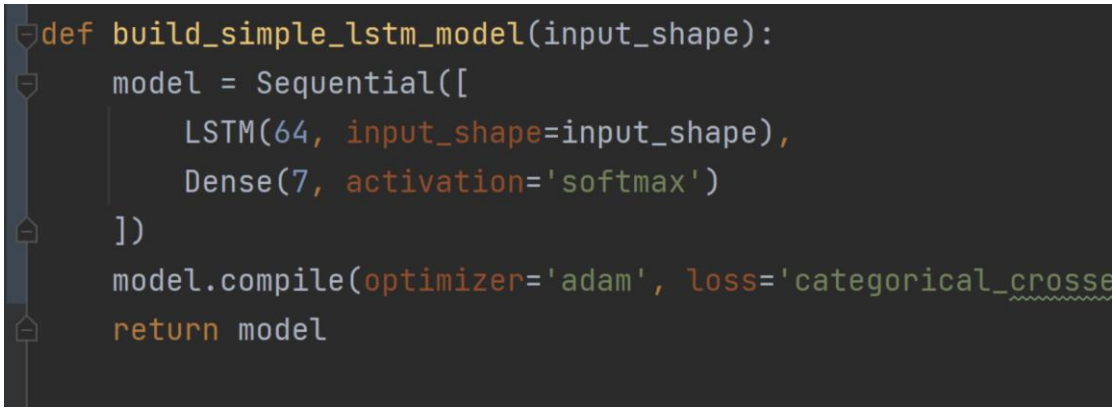


Рисунок 5. Проста модель LSTM

Ця архітектура мінімальна і підходить для високошвидкісного тестування з низькими обчислювальними витратами. Однак результати навчання показують низьку кінцеву точність (близько 58,33%) і варіабельність значення функції втрат, що вказує на нестабільність процесу навчання і обмежену можливість узагальнення моделі.

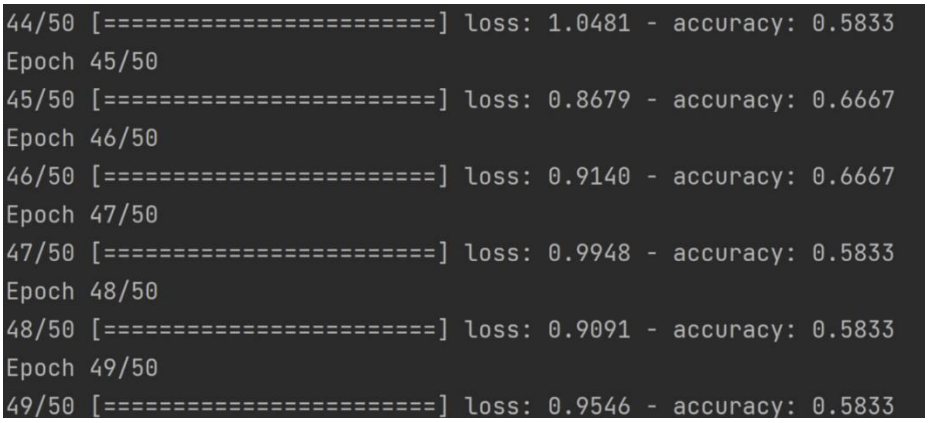


Рисунок 6. Точність та втрати на простій моделі

Друга модель, build_lstm_model, демонструє більш повну архітектуру. Вона містить 2 шари LSTM. Перший шар містить 64 нейрони, а параметр return_sequences=True дозволяє перенести послідовність на наступний шар, тоді як другий шар може бути переданий з 32 нейронами. Після кожного шару LSTM додається шар відсіву на 0,2%, щоб запобігти перенавчання. Крім того, архітектура забезпечує повнозв’язний шар з 64 нейронами і активацією relu, а також шар пакетної нормалізації, який допомагає стабілізувати навчання. У класифікації задіяний останній повнозв’язний шар з 7 виходами і функціями активації softmax. Також перед навчанням моделі була застосована аугментація даних а при навчанні функція швидкої зупинки задля запобігання перенавчання.

Аугментація даних (Data Augmentation) – це метод штучного збільшення кількості тренувальних даних шляхом застосування різноманітних перетворень до існуючих зразків. Цей підхід дозволяє підвищити загальну ефективність моделі, особливо коли початковий набір даних обмежений або не збалансований.

Функція швидкої зупинки (Early Stopping) – це метод, який використовується для запобігання перенавчанню моделі. Ідея полягає в тому, щоб припинити навчання, коли модель перестає покращувати свою продуктивність на валідаційних даних.

```python
def build_lstm_model(input_shape):
    model = Sequential([
        LSTM(64, return_sequences=True, input_shape=input_shape),
        Dropout(0.2),
        LSTM(32),
        Dropout(0.2),
        Dense(64, activation='relu'),
        BatchNormalization(),
        Dense(7, activation='softmax')
    ])
    model.compile(optimizer='adam', loss='categorical_crossentrop
    return model
```

Рисунок 7. Якісна модель LSTM

Додаткові механізми значно підвищили ефективність навчання. Це підтверджується досягненням точності 91% і стійким зниженням втрат.

```
43/50 [==========================] loss: 0.3485 - accuracy: 0.8000
Epoch 44/50
44/50 [==========================] loss: 0.3604 - accuracy: 0.8000
Epoch 45/50
45/50 [==========================] loss: 0.4639 - accuracy: 0.8000
Epoch 46/50
46/50 [==========================] loss: 0.3237 - accuracy: 0.8833
Epoch 47/50
47/50 [==========================] loss: 0.3508 - accuracy: 0.9667
Epoch 48/50
48/50 [==========================] loss: 0.3986 - accuracy: 0.8000
Epoch 49/50
49/50 [==========================] loss: 0.4091 - accuracy: 0.9167
```

Рисунок 8. Точність та втрати на якісній моделі

Порівняння показало, що друга модель значно краща за першу з точки зору точності та стабільності. Використання механізмів вибуття та пакетної нормалізації в поєднанні з багаторівневою архітектурою LSTM дозволяє краще узагальнювати та ефективно враховувати послідовні залежності в даних. Однак, чим складніша архітектура другої моделі, тим довша швидкість

її навчання, тому при розробці моделей реальних додатків, вибір цих двох підходів залежить від обмеженості ресурсів і вимог до якості моделі.

Якість моделі потрібно також визначити і для системи рекомендацій (алгоритм ALS). Перша модель заснована на багатошаровій архітектурі високої щільності (high-density layers) і орієнтована на обробку текстових або табличних даних. Вона складається з декількох рівнів з активацією ReLU, що забезпечує нелінійність моделі і дозволяє виявляти зв'язки в даних. Для стабілізації процесу навчання використовується рівень нормалізації (BatchNormalization), який сприяє більш швидкій і точній оптимізації. Оскільки вихідний рівень містить один нейрон з лінійною активацією, модель підходить для регресійних завдань.

```
LSModel:

 __init__(self):
 self.model = tf.keras.Sequential([
     tf.keras.layers.Dense(64, activation='relu', input_shap
     tf.keras.layers.BatchNormalization(),
     tf.keras.layers.Dense(32, activation='relu'),
     tf.keras.layers.Dense(16, activation='relu'),
     tf.keras.layers.Dense(1, activation='linear')
 ])
 self.model.compile(optimizer='adam', loss='mse', metrics=['
```

Рисунок 9. Проста модель ALS

Однак модель має відносно низьку точність (48%), що вказує на її обмежену здатність пояснювати складні взаємозв'язки між різними типами даних. Зокрема, неможливо ефективно обробляти зображення, які важливі для виконання завдань рекомендаційної системи. Це значно звужує сферу застосування, коли необхідно працювати з даними мультимедіа.

```
Epoch 1/5
1/5 [==========================] loss: 1694.7101 - accuracy: 0.2385
Epoch 2/5
2/5 [==========================] loss: 1508.8824 - accuracy: 0.4840
Epoch 3/5
3/5 [==========================] loss: 1109.1758 - accuracy: 0.4989
Epoch 4/5
4/5 [==========================] loss: 764.1783 - accuracy: 0.4972
Epoch 5/5
5/5 [==========================] loss: 584.6813 - accuracy: 0.4804
```

Рисунок 10. Точність та втрати на простій моделі

Друга модель має більш складну архітектуру, яка включає шар згортки (Conv2D), який може аналізувати візуальні дані, такі як обкладинки книг та плакати фільмів. Шари згортки відображають ключові характеристики зображення, такі як текстури, кольори та форми. Шар Maxpooling2d зменшує просторові розміри даних, виділяє найважливіші характеристики та зменшує обчислювальну складність моделі. Потім дані згладжуються за допомогою шару Flatten, щоб підготувати їх до подальшої обробки в повнозв'язному шарі.

```
class ALSModel:

    def __init__(self):
        self.model = tf.keras.Sequential([
            tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
            tf.keras.layers.MaxPooling2D((2, 2)),
            tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
            tf.keras.layers.MaxPooling2D((2, 2)),
            tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
            tf.keras.layers.Flatten(),

            tf.keras.layers.Dense(256, activation='relu'),
            tf.keras.layers.BatchNormalization(),
            tf.keras.layers.Dropout(0.3),
            tf.keras.layers.Dense(128, activation='relu'),
            tf.keras.layers.BatchNormalization(),
            tf.keras.layers.Dense(64, activation='relu'),
            tf.keras.layers.Dense(32, activation='relu')
        ])
        self.model.compile(optimizer='adam', loss='mse', metrics=['accuracy'])
```

Рисунок 11. Якісна модель ALS

Додатково, у щільних шарах обробляються текстові дані, які інтегруються з візуальними особливостями. Використання шару Dropout із параметром 0.3 дозволяє уникнути перенавчання, що є важливим для роботи з великими наборами даних. Базовий рівень забезпечує адаптацію до різних

завдань, таких як класифікація уподобань та прогнозування рейтингів. Завдяки такій архітектурі модель досягла значно більшої точності – 95%, що свідчить про можливість інтеграції текстової та візуальної інформації.

```
Epoch 1/5
1/5 [==========================] loss: 168.5602 - accuracy: 0.4995
Epoch 2/5
2/5 [==========================] loss: 144.8074 - accuracy: 0.9709
Epoch 3/5
3/5 [==========================] loss: 100.8745 - accuracy: 0.9984
Epoch 4/5
4/5 [==========================] loss: 69.4222 - accuracy: 0.9903
Epoch 5/5
5/5 [==========================] loss: 55.4968 - accuracy: 0.9589
```

Рисунок 12. Точність та втрати на якісній моделі

3.4.4 Алгоритм розподілу для генерації розкладу

Алгоритм автоматичного розподілу розкладу заснований на сучасних методах аналізу даних і використанні штучного інтелекту, що дуже актуально для практичного застосування. Основною перевагою цього підходу є значне скорочення часу, витраченого на ручне планування, яке часто є трудомістким і не завжди ефективним. Алгоритм враховує широкий спектр факторів, таких як пріоритет завдання, тривалість, термін виконання і найбільш продуктивна тривалість, що дозволяє оптимізувати робочий процес. Важливість цього алгоритму також полягає в його здатності враховувати індивідуальні особливості виконання завдань і надавати персоналізований план, що підвищує загальну продуктивність. Використання моделі глибокого навчання, такої як LSTM, забезпечує точність прогнозів, що є ключовим фактором у складних та динамічних сценаріях. З цієї причини алгоритм стає ефективним інструментом управління проектами, навчання та організації особистого чи колективного робочого часу.

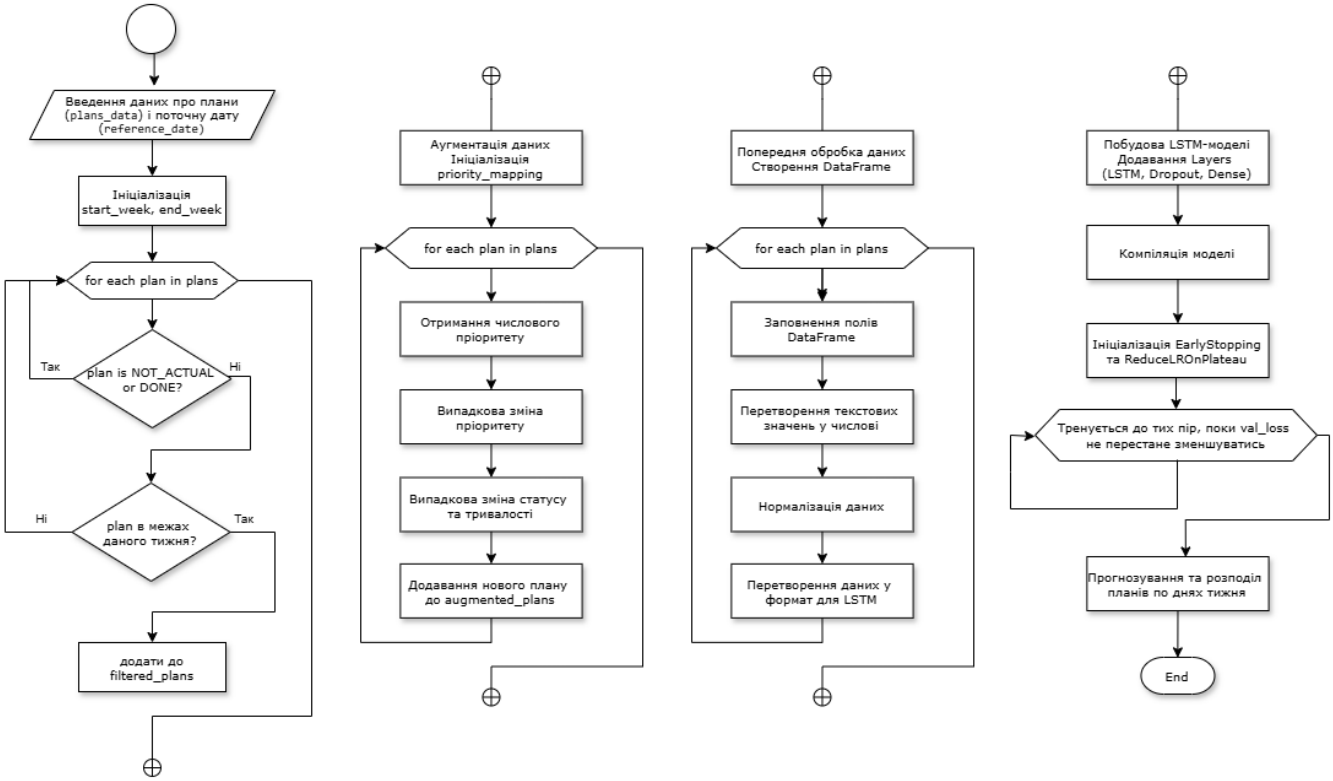


Рисунок 13. Алгоритм розподілу

Першим етапом роботи алгоритму є вибір відповідних завдань. Усі плани, які виконуються або застаріли, виключаються зі списку, що дозволяє зосередитися на завданнях, які дійсно важливі. Крім того, алгоритм враховує тимчасові рамки і вибирає тільки ті завдання, термін виконання яких припадає на поточний тиждень. Це дозволить уникнути перенасичення графіка і вкластися в реальні терміни.

Для підвищення стабільності та адаптивності моделі на другому етапі виконується аугментація даних. Алгоритм додає випадкові зміни до пріоритету, статусу та тривалості завдання для моделювання реальних ситуацій. Це створює більш широкий набір сценаріїв для навчання моделі, щоб система могла належним чином адаптуватися до несподіваних ситуацій.

На етапі обробки даних текстові значення, такі як статус і пріоритет, перетворюються в числовий формат, необхідний для роботи нейронної мережі. Дані нормалізовані до єдиного масштабу, що підвищує стабільність навчання моделі.

Важливим етапом є додавання, компіляйія та навчання моделі на основі LSTM. Ця архітектура дозволяє аналізувати залежності даних. Це важливо для завдань, які розподіляють завдання за часом. Модель вивчає взаємозв'язок між пріоритетами, періодами дня, термінами та іншими факторами для оптимального розподілу завдань по днях тижня.

Виходячи з прогнозів, модель визначає найкращий день тижня для кожного завдання. Для цього враховуються такі параметри, як робоче навантаження на день, пріоритет завдання і період найбільшої продуктивності користувача. Алгоритм розподіляє завдання таким чином, щоб уникнути перевантаження в певний день і забезпечити рівномірний розподіл роботи. Якщо кілька днів залишаються невиконаними, на ці дні будуть перепризначені завдання з низьким пріоритетом. Це дозволить ефективно використовувати всі дні тижня, щоб основне завдання не залишалася незавершеною.

Цей підхід заснований на синтезі методів глибокого навчання та обчислювальної оптимізації для досягнення найвищої точності прогнозування. Він адаптується до різних сценаріїв, підтримує баланс між автоматизацією і гнучкістю і враховує індивідуальні особливості користувача. У порівнянні з традиційними методами планування вручну, алгоритм забезпечує більш високу ефективність і продуктивність, зводячи до мінімуму людські помилки і тимчасові витрати.

3.5 Висновки до розділу

У цьому розділі було проведено детальний аналіз програмного забезпечення для планування робочого часу. Особлива увага приділяється забезпеченню ефективності та персоналізації роботи користувача в системі. Одним з ключових досягнень є впровадження розширених можливостей управління завданнями. Система дозволяє додавати дедлайни, визначати точний час виконання і розставляти пріоритети завдань. Це дозволяє користувачам більш чітко структурувати свої робочі дні та оптимізувати розподіл ресурсів. Також було реалізовано автоматичне планування на основі

алгоритмів машинного навчання з урахуванням таких факторів, як пріоритет, тривалість завдання та індивідуальні звички користувача. Ця функція сприяє більш ефективному використанню часу і знижує навантаження на користувачів.

Інтеграція рекомендаційної системи стала ще одним важливим поліпшенням. Використання алгоритму ALS дозволяє автоматизувати процес рекомендації книг, фільмів чи інших об'єктів на основі уподобань користувача. Ця інтеграція не тільки покращує роботу користувача, але й економить час на пошук потрібної інформації.

Розроблене програмне забезпечення підтримує багаторівневу архітектуру, що включає інтерфейс на Angular і TypeScript, серверну частину на Java з базою даних PostgreSQL і окремий сервер на Python для роботи з нейронними мережами. Така архітектура забезпечує модульність, масштабованість і стабільність системи. Чіткий розподіл функціональних ролей між компонентами дозволяє легко оновлювати і розширювати можливості без ризику дестабілізації програми.

При розробці велика увага приділялася інтеграції алгоритмів машинного навчання. Для оптимізації розкладу завдань використовується модель LSTM, яка аналізує пріоритети, терміни виконання та інші фактори для створення персоналізованих планів. Розроблений алгоритм автоматичного призначення розкладу дозволяє ефективно використовувати робочий час і знизити ймовірність пропуску дедлайнів.

РОЗДІЛ 4: МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЄКТУ

4.1 Опис ідеї проєкту

Програмне забезпечення, що розроблюється – це веб-застосунок, основною метою якого є оптимізація робочого часу користувачів шляхом автоматизації процесів планування, управління завданнями та впровадження сучасних технологій персоналізації. Ідея проєкту спрямована на створення системи, яка забезпечує не лише високу ефективність індивідуальної роботи, але й дозволяє командами працювати злагоджено, підвищуючи їхню продуктивність. Ключовою особливістю розробки є автоматизація планування завдань за допомогою алгоритмів машинного навчання. Подамо зміст ідеї стартап-проєкту у вигляді таблиці.

Таблиця 4.1 – Опис стартап-проєкту

Зміст ідеї	**Напрямки застосування**	**Вигоди для користувача**
Розробка веб-додатку для управління завданнями та планування робочого часу	Індивідуальне планування, командна робота	Оптимізація робочого часу, автоматизація планування, підвищення особистої продуктивності
Автоматичне планування завдань за допомогою алгоритмів машинного навчання	Індивідуальне та персоналізоване використання	Зменшення навантаження на прийняття рішень, врахування індивідуальних звичок і пріоритетів

Продовження таблиці 4.1

Інтеграція рекомендаційної системи для особистих вподобань (наприклад, книги, фільми тощо)	Персоналізовані рекомендації, підбір релевантного контенту	Економія часу, зручність у пошуку контенту, підвищення якості вибору
Підтримка командної роботи з можливістю розподілу завдань між учасниками	Корпоративне середовище, управління проєктами	Поліпшення координації в команді, ефективний розподіл задач, централізоване управління
Впровадження аналізу статистики використання часу та продуктивності	Особисте використання	Контроль продуктивності, виявлення зон для покращення, підвищення ефективності
Використання багаторівневої архітектури для забезпечення модульності та масштабованості	Розгортання у клаудах	Стабільність роботи, можливість обслуговувати велику кількість користувачів без зниження продуктивності

Після запуску веб-застосунку у відкритий доступ і залучення перших користувачів платформа може стати незамінним інструментом для людей, які намагаються ефективно організувати свій робочий час і завдання. Вона

допомагає окремим користувачам складати структурований розклад, не пропускати важливі дедлайни і оптимізувати робочі дні, а також забезпечувати баланс між завданнями і вільним часом. Це значно знижує рівень стресу і підвищує загальну продуктивність. Для команд платформа є важливим інструментом спільної роботи. Учасники можуть легко розподіляти завдання між собою, відстежувати виконання та ділитися оновленнями в режимі реального часу. Така інтеграція може допомогти покращити взаємодію в команді, зменшити непорозуміння та ефективніше досягти спільних цілей.

Надалі буде проведено детальний аналіз технічних і економічних переваг цього рішення, а також його власних характеристик в порівнянні з конкурентами. Особлива увага приділяється виявленню важливих переваг і можливостей для подальшого розвитку. Результати дослідження відображені у вигляді таблиці, що показує переваги і недоліки ідеї в контексті конкурентного середовища.

Таблиця 4.2 – Визначення характеристик стартап-проєкту

Критерії	Аналоги			**Розроблене ПЗ**
	Todoist	**Monday.com**	**TickTick**	
Статуси, дедлайни, пріоритет завдань	+	+	+	+
Розклад для планування (7 днів)	+	-	+	+
Різні типи завдань	+	+	+	+
Командна робота	-	+	-	+
Складання індивідуального розкладу	-	-	-	+
Рекомендації для користувачів	-	-	-	+

Продовження таблиці 4.2

Декомпозиція завдань	-	-	-	+
Аналіз статистики користувача	-	-	-	+

Розроблене програмне забезпечення виділяється серед сучасних систем управління завданнями завдяки своїм інноваційним функціям, які враховують потреби як окремих користувачів, так і команд. Його основна функція – автоматичне створення персонального розкладу на основі алгоритмів машинного навчання. Ця функція дозволяє враховувати терміни, пріоритети і звички користувачів, щоб оптимізувати робочий процес і забезпечити розумне використання часу. Інші унікальні можливості програмного забезпечення включають систему рекомендацій для користувачів, засновану на аналізі попередніх дій та вподобань. Також реалізована декомпозиція завдань, що дозволяє розділити їх на підзадачі для детального планування. Крім того, в систему включений модуль аналізу статистики, який забезпечує графічну візуалізацію прогресу користувачів.

4.2 Технологічний аудит ідеї проєкту

Для успішної реалізації будь якого проєкту важливим етапом буде проведення технологічного аудиту. Він включає аналіз технологій, необхідних для створення продукту, та їх доступність для використання. Цей аудит дозволить оцінити рівень технологічної готовності та ефективність запропонованого рішення.

Першим кроком є аналіз ключових технологій, які планується використовувати в реалізації ідеї проєкту. Зокрема, це програмні засоби, фреймворки, бібліотеки, бази даних та мікросервісна архітектура. Визначається, чи є ці технології доступними, чи їх потрібно розробляти або адаптувати. Здійснюється оцінка готовності рішень до інтеграції у розроблюваний продукт. Оцінюється також можливість доступу до зазначених

технологій: чи є вони відкритими для використання, ліцензованими, або потребують додаткових витрат на їх придбання чи підтримку.

Таблиця 4.3 – Технологічний аудит ідеї проєкту

№	Ідея проєкту	Технології для написання	Наявність технологій	Доступ до дани технологій
1	Створення бекенду	Python, Spring Boot	Наявна	Доступна
2	Реалізація фронтенду	Angular, TypeScript	Наявна	Доступна, використання безкоштовне
3	Збереження та обробка даних	PostgreSQL, Hibernate	Наявна	Доступна
4	Аналіз даних	Apache Spark, TensorFlow	Наявна	Доступна, відкритим кодом
5	Реалізація нейронних мереж	Apache Spark, TensorFlow, Python	Розроблена у межах роботи	Доступна

Для реалізації ідеї проєкту було обрано такі основні технології. Python використовується для розробки високопродуктивних серверних компонентів, а саме для використання нейронних мереж, Java – для високопродуктивних мікросервісів, а саме для авторизації та обробки даних, а TypeScript – для front-end розробки. Spring Boot забезпечує швидку розробку серверної логіки, Angular дозволяє створювати інтерактивний користувальницький інтерфейс. PostgreSQL – найкраще рішення для зберігання структурованих даних, а Hibernate полегшує взаємодію з базою даних. Apache Spark використовується для обробки великих обсягів даних, а TensorFlow – для побудови моделей машинного навчання.

На підставі аналізу технології можна зробити висновок, що всі необхідні рішення для реалізації проекту є. Вибрані інструменти ефективні, протестовані та підтримуються широкою спільнотою. Використання цих технологій забезпечує високий рівень продуктивності, масштабованості та надійності продукту.

4.3 Аналіз ринкових можливостей запуску стартап-проєкту

Оцінити ринкові можливості – є важливим етапом для визначення можливості реалізації стартап-проєкту. Цей аналіз може допомогти визначити перспективи, проблеми та ризики, які можуть вплинути на успіх проєкту. Завдяки детальному вивченню ринку, потреб клієнтів і конкурентного середовища можна ефективно розробляти стратегії розробки продукту і позиціонування.

Сфера програмного забезпечення для планування демонструє стійке зростання в зв'язку з потребою компаній і окремих користувачів в цифрових рішеннях для оптимізації процесів. Попит на такі інструменти активно зростає, оскільки багато компаній переходять на гнучкі моделі роботи.

Основні вимоги клієнтів пов'язані з різноманітністю інструментів (для індивідуального і командного використання), простою інтеграція з іншими сервісами, аналіз та рекомендації щодо підвищення продуктивності.

На ринку вже є такі гравці, як Todoist,Monday.com і TickTick, які надають широкий спектр функцій. Але їх рішення має певні обмеження, наприклад: недостатня персоналізація розкладу, відсутність аналізу та рекомендацій для користувачів.

Основними ризиками проекту є: висока конкуренція серед великих гравців, складність привернення уваги до нового продукту без значних витрат на маркетинг, технічні бар'єри для інтеграції з іншими популярними платформами.

Таблиця 4.4 – Характеристики потенційного ринку

№	Стан ринку стартапу для планування	Характеристики
1	Кількість головних гравців	3–5 компаній
2	Загальний обсяг ринку	Близько 150000 грн/рік
3	Динаміка ринку	Стабільний розвиток із приростом 8–12% на рік
4	Обмеження для входу на ринок	Конкуренція та необхідність інноваційного підходу
5	Потреби користувачів	Персоналізація, рекомендації, аналіз продуктивності
6	Середня рентабельність в галузі, %	Від 20% до 30%

Аналіз показує, що існує значний потенціал для реалізації стартап-проєктів, пов'язаних з плануванням. Попит на цифрові інструменти управління завданнями продовжує зростати, а потреба в більш інноваційних рішеннях створює можливості для нових гравців на ринку.

Ризики, пов'язані з конкуренцією, можуть бути зведені до мінімуму завдяки унікальним функціям, таким як аналітика користувачів, автоматизовані рекомендації та підтримка як команд, так і окремих користувачів. Це відкриває привабливі перспективи для виходу на ринок і довгострокового розвитку проєкту.

У наступній таблиці представлені основні категорії потенційних замовників розроблюваного програмного забезпечення, їх потреби, поведінкові характеристики і вимоги до продукту. Цей аналіз враховує особливості різних груп користувачів, тому важливо зосередити стратегію розробки та маркетингу на найпоширеніших функціях.

Таблиця 4.5 – Характеристики потенційних клієнтів

№	Потреби ринку	Цільова аудиторія	Різниця поведінки цільових груп	Потреби споживачів
1	Ефективне управління особистими завданнями	Індивідуальні користувачі	Шукають простоту використання, швидкий доступ до функцій	Інтуїтивний інтерфейс, персоналізація планування, зручність додавання завдань
2	Оптимізація командної робот	Малі та середні команди	Шукають можливості спільного редагування та управління	Інструменти для командної роботи, прозорість у виконанні завдань, доступ до статистики
3	Поліпшення продуктивності	Користувачі, які хочуть вдосконалити свій робочий процес	Орієнтовані на аналіз прогресу, потребують детального планування	Аналітика продуктивності, автоматизоване складання розкладу, можливість декомпозиції задач
4	Виконання великих проектів	Індивідуальні користувачі	Шукають інструменти для управління завданнями	Декомпозиція задач, інтеграція з іншими системами

В аналітичній таблиці вказані основні ризики, з якими може зіткнутися програмне забезпечення, і запропоновані можливі стратегії їх подолання. Цей аналіз дозволяє виявити слабкі сторони проекту, щоб заздалегідь сформувати ефективний механізм реагування.

Таблиця 4.6 – Фактори загроз

№	Впливаючи фактори	Загрози	Що в даній ситуації може зробити компанія
1	Конкуренти	Поява схожих систем з аналогічним функціоналом	Розробка унікальних функцій, таких як рекомендації, інтеграція з ШІ, розширення аналітики
2	Обмежене фінансування	Відсутність достатніх коштів для масштабування	Пошук інвестицій, подання на гранти
3	Низька популярність серед аудиторії	Недостатнє просування продукту, низька впізнаваність на ринку	Активізація маркетингової кампанії, співпраця з інфлюенсерами, пропозиція безкоштовних базових функцій
4	Технічні недоліки	Затримки в оновленні або нестабільна робота	Постійний моніторинг системи, швидке реагування на збої, регулярні релізи оновлень

Насутпна таблиця розкриває ключові можливості для розвитку програмного забезпечення. Розробка унікальних функцій, таких як декомпозиція завдань та індивідуальні рекомендації, може стати основою для виходу на ринок. Зворотний зв’язок від користувачів дозволяє оперативно адаптувати продукт під реальні потреби. Монетизація через рекламу відкриває додаткові джерела доходу, зберігаючи доступність базових функцій для широкої аудиторії.

Таблиця 4.7 – Фактори можливостей

№	Впливаючи фактори	Можливості	Що в даній ситуації може зробити компанія
1	Новий продукт	Можливість представити унікальне рішення, яке закриває специфічні потреби ринку	Активна робота з цільовою аудиторією, просування унікальних функцій, таких як автоматичне складання розкладу та рекомендації
2	Вихід аналогу	Вихід схожого продукту може стимулювати вдосконалення існуючих рішень	Аналіз функціоналу конкурента, вдосконалення власного продукту, інтеграція нових функцій
3	Зворотний зв’язок від користувачів	Можливість отримати корисні інсайти для поліпшення продукту	Регулярна комунікація із користувачами, швидке оновлення на основі їх запитів, проведення опитувань
4	Монетизація через рекламу	Можливість отримати дохід через використання рекламних інтеграцій	Розробка механізмів ненав’язливої реклами, партнерство з компаніями, що відповідають інтересам аудиторії

Аналіз конкурентного середовища показує, що ринок програмного забезпечення для управління завданнями характеризується високою конкуренцією, як на регіональному, так і на глобальному рівнях. Для успішної розробки продуктів необхідно забезпечувати баланс між ціною і якістю, впроваджувати інноваційні рішення, що відрізняють продукти від їх аналогів.

Гнучка стратегія створення впізнаваного бренду і роботи на різних ринках допоможе вам успішно конкурувати і залучати нових користувачів.

Таблиця 4.8 – Аналіз конкурентного середовища на ринку

№	Конкурентне середовище	Прояв характеристик конкурентного середовища	Фактори що можуть вплинути на підприємство
1	Тип конкуренції: монополістична	Декілька великих компаній формують ринок	Акцент на унікальних функціях (рекомендації, автоматизація), фокус на задоволенні специфічних потреб
2	Рівень конкурентної боротьби: світовий	Конкуренти працюють на глобальному ринку	Розробка міжнародної стратегії, локалізація продукту для різних ринків, створення гнучкої системи підтримки
3	Галузева ознака: внутрішньогалуз ва	Конкуренти переважно працюють у сфері управління завданнями	Налагодження партнерств, наприклад, інтеграція з іншими продуктами для розширення функціональності
4	Функціональна	Продукти з різним рівнем функціоналу конкурують між собою	Акцент на автоматизацію та персоналізацію, впровадження аналітики та гейміфікації

Продовження таблиці 4.8

5	Характер конкурентних переваг: цінова та нецінова	Орієнтація на якість, дизайн, інтуїтивність	Збалансування вартості та функціоналу, створення привабливого UI/UX, покращення підтримки
6	За інтенсивністю: марочна	Бренди змагаються за впізнаваність	Створення впізнаваного бренду через позитивні відгуки, активне просування на ринку

У наступній таблиці проаналізовано конкурентне середовище проєкту за допомогою моделі Майкла Портера «П'ять сил». Прямі конкуренти представлені такими популярними сервісами, як Todoist і Monday.com, вони вже міцно зарекомендували себе на ринку. Головне завдання – перевершити їх за допомогою унікальних функцій, таких як автоматизація розкладу, рекомендації або декомпозиція завдань. Потенційні конкуренти, включаючи нові стартапи, завжди представляють ризики, тому потрібно постійно вдосконалювати свій продукт. Постачальники хмарних послуг впливають на витрати, тому важливо вибрати вигідне рішення. Замовники потребують персоналізації та зручності для визначення напрямку розвитку. Альтернативні продукти, такі як рішення з відкритим кодом, можуть бути корисними альтернативами, але неефективними при комплексному управлінні.

Таблиця 4.9 – Аналіз конкурентоспроможності за М. Портером

Складові аналізу	Прямі конкуренти	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Приклади	Todoist, Monday.com, TickTick	Нові стартапи з управління завданнями	Провайдери хмарних послуг	Індивідуальні користувачі та команди	Open-source рішення, таблиці Excel

Продовження таблиці 4.9

Висновки	Потрібно перевершити конкурентів за якістю сервісу та функціоналу	Постійна загроза нових гравців на ринку, потребує вдосконалення функціоналу	Оцінювати політику цін на хмарні сервіси, використовувати оптимальні рішення	Необхідно підвищувати цінність для клієнтів через функціонал і зручність	Регулярні інновації допоможуть мінімізувати вплив товарів-замінників

У наступній таблиці наведено основні фактори, що визначають конкурентоспроможність проєкту. Позиціонування на ринку дозволяє виділити проєкт унікальними можливостями, які не представлені у конкурентів. Технологічність забезпечує сучасний підхід до управління завданнями, заснований на передових алгоритмах. Унікальність є важливим аспектом, що дозволяє значно поліпшити якість обслуговування користувачів. Індивідуальне налаштування забезпечує адаптацію системи до індивідуальних вимог замовника, створюючи додаткову цінність. Цінова політика, заснована на оптимізації процесів, забезпечує доступність продукту для широкого кола користувачів.

Таблиця 4.10 – Фактори конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування
1	Позиціонування на ринку	Проект займає унікальну нішу завдяки поєднанню автоматизації, рекомендацій і аналітики.

Продовження таблиці 4.10

2	Технологічність	Використання сучасних алгоритмів, таких як LSTM для планування та ALS для рекомендацій.
3	Унікальність	Система забезпечує автоматизоване створення розкладу, що економить час користувачів.
4	Індивідуальність	Можливість адаптувати задачі під індивідуальні потреби користувачів, включаючи декомпозицію.
5	Цінова політика	Завдяки оптимізації процесів програмне забезпечення може запропонувати конкурентні ціни.

Наступна таблиця порівнює основні конкурентні фактори проєкту з аналогами. Зручний інтерфейс є сильною стороною продукту, який оцінюється високо серед конкурентів. А надійність та масштабованість отримали максимальний бал, що свідчить про готовність продукту до зростання навантаження і роботи в різних сценаріях.

Таблиця 4.11 – Аналіз сильних та слабких сторін з конкурентами

№	Фактор конкурентоспроможності	Бали (1-20)	Рейтинг конкурентів у порівнянні з розроблюваним продуктом						
			-3	-2	-1	0	1	2	3
1	Зручний інтерфейс	13				V			
2	Надійність та масштабованість	18						V	
3	Використання унікальних функцій	15		V					

SWOT-аналіз дозволяє оцінити внутрішні та зовнішні фактори, що впливають на проєкт. Серед сильних сторін слід відзначити унікальність продукту та використання сучасних технологій, які створюють конкурентну перевагу. Слабкі сторони зосереджені на початковій недосконалості, зокрема відсутності мобільної підтримки. Серед можливостей – зростання попиту на автоматизацію завдань і вихід на міжнародні ринки. Загрози включають конкуренцію та ризики втрати унікальності через стандартизацію.

Таблиця 4.12 – SWOT аналіз

Сильні сторони (S):	Слабкі сторони (W):
Автоматизоване складання персонального розкладу, рекомендації та аналітика, які не представлені у конкурентів. Використання інноваційних алгоритмів: LSTM для планування та ALS для рекомендацій. Зручний і адаптивний дизайн, який підходить для різних типів користувачів. Створення підзадач, гнучке налаштування пріоритетів і типів завдань. Система здатна працювати з великими обсягами даних, що дозволяє використовувати її як для індивідуальних користувачів, так і для команд.	Обмеження доступу для користувачів, які працюють із мобільних пристроїв. Потреба в значних інвестиціях: потрібно багато ресурсів для розробки і підтримки системи. Відсутність довіри на ринку через початковий етап розвитку проекту. Залежність від хмарних послуг: наприклад можливі зміни у ціновій політиці постачальника.

Продовження таблиці 4.12

Можливості (O):	Загрози (T):
Більшість користувачів зацікавлені в ефективному управлінні завданнями з допомогою сучасних технологій. Вихід на міжнародний ринок: можливість адаптувати продукт для іноземних користувачів через локалізацію та підтримку різних мов. Впровадження мобільної підтримки, інтеграція календарів і сторонніх сервісів. Активне опитування користувачів дозволяє швидко адаптувати продукт їхніх потреб.	Нові стартапи або вдосконалення існуючих продуктів можуть зменшити конкурентні переваги. Більші компанії можуть впровадити стандартизовані рішення, які звузят унікальність продукту. Потенційна втрата інтересу користувачів через перенасичення ринку схожими рішеннями. Можливі затримки в оновленнях, нестабільна робота на початкових етапах. Зміни у вартості хмарних сервісів або необхідність зниження ціни продукту через тиск з боку конкурентів.

Альтернатива впровадженню на ринок визначає можливий спосіб запуску продукту. Запуск власного веб-сайту – найбільш перспективний варіант, який гарантує незалежність і контроль над продуктом. Співпраця з платформою допоможе розширити охоплення вашої аудиторії за допомогою партнерських проектів. Участь у зустрічах підвищує обізнаність і дозволяє представити продукт цільовій аудиторії. Для досягнення максимального ефекту ці заходи можна проводити одночасно.

Таблиця 4.13 – Альтернативи поведінки на ринку

№	Альтернатива поведінки на ринку	Шанс на отримання певних ресурсів	Час отримання
1	Запуск власного вебзастосунку	Висока, основний ресурс – час і команда	2-5 місяців
2	Співпраця з існуючими платформами	Середня, потребує залучення партнерів	3-6 місяців
3	Участь у тематичних конференціях	Висока, потрібні ресурси на організацію, кошти, час	1-3 місяці

4.4 Розроблення ринкової стратегії проєкту

Ринкова стратегія проєкту з планування завдань базується на аналізі цільових груп, специфіки ринку та конкурентних переваг. Вона спрямована на ефективне впровадження продукту, забезпечення його унікальності та задоволення потреб різних сегментів користувачів.

Таблиця 4.14 – Цільові групи потенційних споживачів

№	Опис потенційних клієнтів	Чи готові споживачі сприйняти продукт?	Приблизний попит у цільових групах	Сила конкуренції	Легкість входу до ринку
1	Індивідуальні користувачі, що прагнуть покращити особисту продуктивність	Так	Висока	Середня	Легка

Продовження таблиці 4.14

2	Команди малого, яким потрібно спільне управління завданнями	Так	Середня	Висока	Середня

Основні стратегії розвитку включають запуск онлайн-платформи, яка надає доступ до інноваційних інструментів планування завдань. Основними елементами стратегії є наприклад стратегія охоплення ринку. Маркетингові кампанії, спрямовані на активне залучення цільової аудиторії через соціальні мережі, SEO-оптимізація для поліпшення видимості в пошукових системах і партнерство з популярними платформами. А також конкурентна перевага. Диференціація продукту з функціями, яких немає у більшості конкурентів. Це автоматизація процесів, персоналізовані рекомендації, підтримка командної роботи і статистичний аналіз користувачів.

Таблиця 4.15 – Визначення подальшої стратегії просування стартапу

№	Альтернатива розвитку стартапу	Просування на ринку	Ключові позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Розробка та запуск платформи для управління завданнями	Просування через соціальні мережі, SEO-оптимізація, просування через партнерів	Унікальні функції: автоматизація складання розкладу, персоналізація, аналіз продуктивності; зручний інтерфейс	Диференціація

Хоча проєкт не є абсолютним "першопрохідцем" на ринку, він пропонує інноваційний підхід, який фокусується на автоматизації планування та персоналізації. Стратегія конкурентної поведінки заснована на залученні нових користувачів, в тому числі тих, хто використовує аналоги, завдяки інноваційним функціям, яких немає у конкурентів. Диференціація-це основний напрямок, що дозволяє виділити продукт серед аналогічних рішень.

Таблиця 4.16 – Базової стратегія конкурентної поведінки

№	Стартап «першопрохідець» на ринку?	Компанія шукатиме нових користувачів?	Компанія копіюватиме функціонал конкурентів?	Стратегія конкурентної поведінки
1	Ні, оскільки існують непрямі аналоги, але він пропонує новаторський підхід до планування завдань	Компанія залучатиме нових споживачів, зокрема тих, хто вже користується іншими сервісами для планування	Ні. Основний акцент на унікальних функціях: автоматизація планування, персоналізація, аналітика продуктивності	Диференціація

Стратегія позиціонування проєкту повинна підкреслювати ключові переваги, такі як швидкість і простота використання, унікальність функцій (рекомендації, автоматизація, індивідуальний розклад), гнучкість в задоволенні потреб користувачів, а продукт повинен позиціонуватися як інноваційний інструмент для забезпечення ефективності та поліпшення користувацького досвіду.

Таблиця 4.17 – Визначення стратегії позиціонування

№	Фактори важливі для цільової аудиторії	Базова стратегі розвитку	В яких позиціях проєкт буде конкурентоспроможним?	Асоціації, які мають сформувати комплексну позицію проект (три ключових)
1	Швидкість, якість, надійність процесу планування, простота у використанні	Стратегія диференціації	Унікальність функцій, гнучкість обслуговування орієнтація на потреби користувачів	Інновації, ефективність, зручність для користувачів

4.5 Розроблення маркетингової програми стартап-проєкту

Маркетингова програма стартапу, спрямована на автоматизацію планування робочого часу з використанням алгоритмів машинного навчання, заснована на формуванні чіткої концепції продукту, орієнтованої на потреби і конкурентні переваги цільової аудиторії.

Таблиця 4.18 – Переваг концепції потенційного товару

№	Потреби користувачів	Вигоди, які може запропонувати стартап	Ключові конкурентоспроможні переваги
1	Потреба в автоматизації та структуризації завдань	Надає користувачам можливість автоматично складати персональний розклад	Швидкість планування, точність рекомендацій, аналітика продуктивності

Продовження таблиці 4.18

2	Потреба у зручному інтерфейсі	Забезпечує простий і зручний у використанні додаток з гнучкими налаштуваннями	Інтуїтивний дизайн, адаптивність до потреб користувачів
3	Потреба у підтримці командної роботи	Дає можливість командам спільно працювати над завданнями, відслідковувати прогрес	Функціонал для спільного редагування, прозорість процесів, автоматизація розподілу задач
4	Потреба у доступності та персоналізації	Доступність як для індивідуальних користувачів, так і для команд через гнучкі підписки	Гнучка система цін, персоналізація функцій відповідно до профілю користувача

Концепція товару зосереджена на вирішенні ключових потреб користувачів: автоматизація процесів планування, доступність і гнучкість. Основні переваги включають унікальні функції, такі як аналітика продуктивності та автоматичне складання розкладу.

Таблиця 4.19 – Визначення переваг концепції потенційного товару

Рівні товару	Сутність та складові
I. Товар за задумом	Онлайн платформа для автоматизації планування завдань із використанням алгоритмів машинного навчання
II. Товар у реальному виконанні	
Властивості:	Автоматичне планування, аналітика продуктивності, персоналізовані рекомендації

Продовження таблиці 4.19

Якість:	Тестування програми, моніторинг роботи, надійність і стійкість до високих навантажень
Пакування:	SaaS-рішення з кросплатформеним доступом через браузер і мобільний додаток
Марка:	Назва компанії та продукту, що асоціюються з інноваціями і продуктивністю
III. Товар із підкріпленням	
До продажу:	Консультації для користувачів щодо вибору функцій і підписок, демо-версії з повним доступом до базових функцій
Після продажу:	Підтримка користувачів, регулярні оновлення, консультації щодо використання, рекомендації на основі аналітики
Захист:	Реєстрація авторських прав на унікальні елементи (дизайн, логотип, алгоритми)

Концепція продукту представлена на трьох рівнях. Перший – це онлайн-платформа, яка відрізняється простотою використання та універсальністю. У реальному виконанні продукт забезпечує інноваційну функціональність і надійність. Удосконалення продукту включають підтримку клієнтів та захист інтелектуальної власності.

Концепції цінової політики, системи продажів і маркетингових комунікацій є ключовими елементами стратегії виходу стартапу на ринок. Вони засновані на аналізі деталей цільової аудиторії, конкурентів, каналів комунікації та поведінки клієнтів.

Таблиця 4.20 – Визначення меж встановлення ціни

Ціни у товарів конкурентів	Середній рівень цін у аналогів	Доходи у цільовій групі користувачів	Верхня та нижня межі встановлення ціни на товар/послугу
Середній	$15–50 (середня ціна на аналоги)	Доходи середній та вище середнього рівня	Мінімальна ціна: $5, максимальна ціна: $100

Система збуту базується на онлайн-каналах, що враховують звички цільової аудиторії, яка орієнтується на швидкість і якість обслуговування. Основний акцент робиться на власному вебсайті, який пропонує прямий продаж послуг, а також на партнерстві з онлайн-маркетплейсами для збільшення охоплення.

Таблиця 4.21 – Система збуту стартапу

Що шукають користувачі та за що готові платити?	Функції, які має надати постачальник товару	Канали збуту	Що використовуватимесь для системи збуту?
Клієнти шукають інноваційні рішення, готові протестувати нові продукти	Інформування про продукт, забезпечення швидкого доступу до тестових версій, збір відгуків користувачів	Широкий: вебсайти, соціальні мережі, маркетингові платформи	Власна платформа для продажу, партнерство з платформами для оглядів (G2, Capterra)

Продовження таблиці 4.21

Перевага надається легкодоступним рішенням із простим процесом придбання	Автоматизація процесу оформлення замовлення, прозорість умов підписки, інтеграція з платіжними системами	Неглибокий: прямий канал через вебсайт	Прямі продажі через сайт, інтеграція з App Store/Google Play
Команди середнього бізнесу, які потребують демонстрації функціоналу	Проведення демонстрацій продукту, консультації, персоналізовані презентації	Глибокий: відділи продажів, спеціалізовані майданчики для SaaS-рішень	Побудова команди продажів, співпраця з консалтинговими компаніями

Маркетингова комунікація заснована на підкресленні унікальних переваг продукту: інноваційності, високої якості, зручності. Для надання інформації глядачам використовуються популярні онлайн-канали, такі як соціальні мережі, електронна пошта та платформи для ведення блогів. Основний упор робиться на інтерактивний формат (відео, демонстрація), в якому ви можете детально розповісти про особливості продукту і його переваги.

Таблиця 4.22 – Як проходитимуть маркетингові комунікації

№	Поведінка майбутніх користувачів	Канали комунікацій притаманні клієнтам	Ключові позиції стартап продукту	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Цінують простоту використання, зручність і швидкість	Соціальні мережі (Facebook, LinkedIn, Instagram), вебсайти, email-розсилки	Зручність, швидкість, інтуїтивність	Показати, як легко та швидко продукт вирішує завдання	Презентація основних функцій через короткі відео, приклади застосування у реальних ситуаціях
2	Цікавляться аналітичними інструментами для покращення продуктивності	Спеціалізовані платформи для оглядів SaaS (Capterra, G2), блоги	Аналітика, оптимізація роботи	Розкрити переваги аналітики для індивідуальних користувачів і команд	Написання статей, оглядів, створення візуалізацій результатів, які надає продукт

4.6 Висновки до розділу

У даному розділі проведено маркетинговий аналіз стартап-проєкту, спрямованого на розробку інноваційного веб-застосунку для планування робочого часу. Основною перевагою продукту є автоматизація планування, персоналізація роботи з користувачами та використання нейронних мереж для підвищення ефективності роботи як окремих користувачів, так і команд.

Проведений технологічний аудит показав, що всі необхідні інструменти для реалізації проєкту є доступними. Використання Python, TensorFlow, Angular та PostgreSQL дозволяє забезпечити високу продуктивність, масштабованість та стабільність роботи системи. Це створює надійне підґрунтя для впровадження інноваційних рішень, таких як нейронні мережі для автоматичного створення персоналізованих розкладів.

Аналіз ринкових можливостей показав, що потреба у цифрових рішеннях для планування зростає. Проте стартап стикається з такими викликами, як висока конкуренція на ринку та потреба у значних маркетингових інвестиціях. Для зниження цих ризиків було розроблено диференційовану маркетингову стратегію, спрямовану на акцентування унікальних переваг продукту: аналізу статистики, автоматизації та рекомендацій для користувачів.

На основі аналізу конкурентного середовища виявлено, що унікальний функціонал продукту, зокрема можливість декомпозиції завдань та підтримка командної роботи, може суттєво виділити його серед існуючих аналогів. Акцент на потребах індивідуальних користувачів і малих команд дозволяє задовольнити попит у різних сегментах ринку.

Таким чином, запропонований стартап-проєкт має високий потенціал для успішного впровадження на ринок. Подальші кроки включають масштабування проєкту, впровадження мобільної версії продукту та активну роботу з користувачами для вдосконалення функціоналу і підвищення конкурентоспроможності.

ВИСНОВКИ

За результатами дослідження було отримано значні результати, що підтверджують ефективність обраного підходу до розробки програмного забезпечення для планування робочого часу.

Перше що потрібно було зробити – це провести аналіз. Детальний та глибокий. Було розглянуто популярні платформи, такі як Todoist і Monday.com. У ході аналізу було визначено як сильні сторони, так і недоліки. Наприклад, відсутність персоналізації чи недостатня аналітика. А також невелика кількість базових функціональних можливостей. Тож були поставлені задачі на покращення вже існуючих рішень.

Важливо було не просто використати функціонал аналогів, а додати можливості, які змінять підхід: створення індивідуального розкладу за потребами користувача, декомпозиція задач, для більш простого виконання завдань, система трекінгу звичок, інтерактивні аналітичні інструменти.

Задля створення нового якісного програмного забезпечення було проведено аналіз існуючих програмних засобів. Відповідно, інтерфейс сучасний і легкий у використанні. Angular на фронтенді гарантує швидкість і динамічність. На бекенді Java з Spring Boot забезпечують надійність, а PostgreSQL відповідає за використання нейромереж у проєкті. Було розроблено багаторівневу архітектуру, базу даних та відображення у вигляді веб-застосунку.

Також варто зазначити використані алгоритми. Було створено новаторський механізм генерації розкладу, завдяки LSTM-рекурентним нейронним мережам. Це дозволило автоматизувати весь процес. Користувач задає дедлайни, визначає пріоритети – алгоритм усе зводить у чіткий і зручний розклад. Система аналізує звички, пропонує певні покращення та працює в режимі реального часу як особистий помічник.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1) Daily planning is important. For everyone? [Електронний ресурс]. — 2023 — Режим доступу: https://uk.elvtr.com/blog/daily-planning-is-important-for-everyone

2) Pros and Cons of Python Programming Language [Електронний ресурс]. — 2024 — https://serokell.io/blog/python-pros-and-cons

3) Pros and Cons of Java Development in 2023 [Електронний ресурс]. — 2023 — Режим доступу: https://www.netguru.com/blog/java-pros-and-cons

4) Why You Should Choose TypeScript Over JavaScript [Електронний ресурс]. — 2023 — Режим доступу: https://serokell.io/blog/why-typescript

5) Getting Started with Spring Boot: Advantages, Disadvantages, and Use Cases [Електронний ресурс]. — 2023 — Режим доступу: https://medium.com/@jayeshwarke011/getting-started-with-spring-boot-advantages-disadvantages-and-use-cases-497b0f04fb86

6) Why Use Angular? Pros and Cons over Other Frameworks [Електронний ресурс]. — 2022 — Режим доступу: https://exoft.net/angular-pros-and-cons/

7) Advantages of PostgreSQL [Електронний ресурс]. — 2024 — Режим доступу: https://www.danytechcloud.com/advantages-of-postgresql/

8) Elevating Development Practices: Unveiling The Top 7 Advantages Of Hibernate Framework [Електронний ресурс]. — 2023 — Режим доступу: https://medium.com/@techisbeautiful/elevating-development-practices-unveiling-the-top-7-advantages-of-hibernate-framework-ca1bbca79ed0

9) Hibernate Framework in Java: Detailed Overview [Електронний ресурс]. — 2024 — Режим доступу: https://www.simplilearn.com/hibernate-framework-in-java-article

10) Загальний опис Tensorflow [Електронний ресурс]. — 2022 — Режим доступу: https://www.tensorflow.org/about.

11) Загальний опис PyTorch [Електронний ресурс]. — 2023 — Режим доступу: https://pytorch.org/features/.

12) REST API: what it is, how it works, advantages and disadvantages [Електронний ресурс]. — 2023 — Режим доступу: https://www.thepowermba.com/en/blog/rest-api-what-it-is

13) What are the benefits of a microservices architecture? [Електронний ресурс]. — 2022 — Режим доступу: https://about.gitlab.com/blog/2022/09/29/what-are-the-benefits-of-a-microservices-architecture/

14) Загальний опис Python [Електронний ресурс]. — Режим доступу: https://www.python.org/about/.

15) Загальний опис Pandas [Електронний ресурс]. — Режим доступу: https://pandas.pydata.org/pandas-docs/stable/index.html.

16) Загальний опис Dask [Електронний ресурс]. — Режим доступу: https://docs.dask.org/en/latest/.

17) Рекомендовані практики використання Dask [Електронний ресурс]. — Режим доступу: https://docs.dask.org/en/latest/best-practices.html.

18) Join-операції для Dask DataFrame [Електронний ресурс]. — Режим доступу: https://docs.dask.org/en/latest/best-practices.html.

19) Статистика щодо популярності Apache Airflow [Електронний ресурс]. — Режим доступу: https://www.astronomer.io/blog/why-airflow.

20) Про ефективність Python [Електронний ресурс]. — Режим доступу: https://www.monterail.com/blog/is-python-slow.

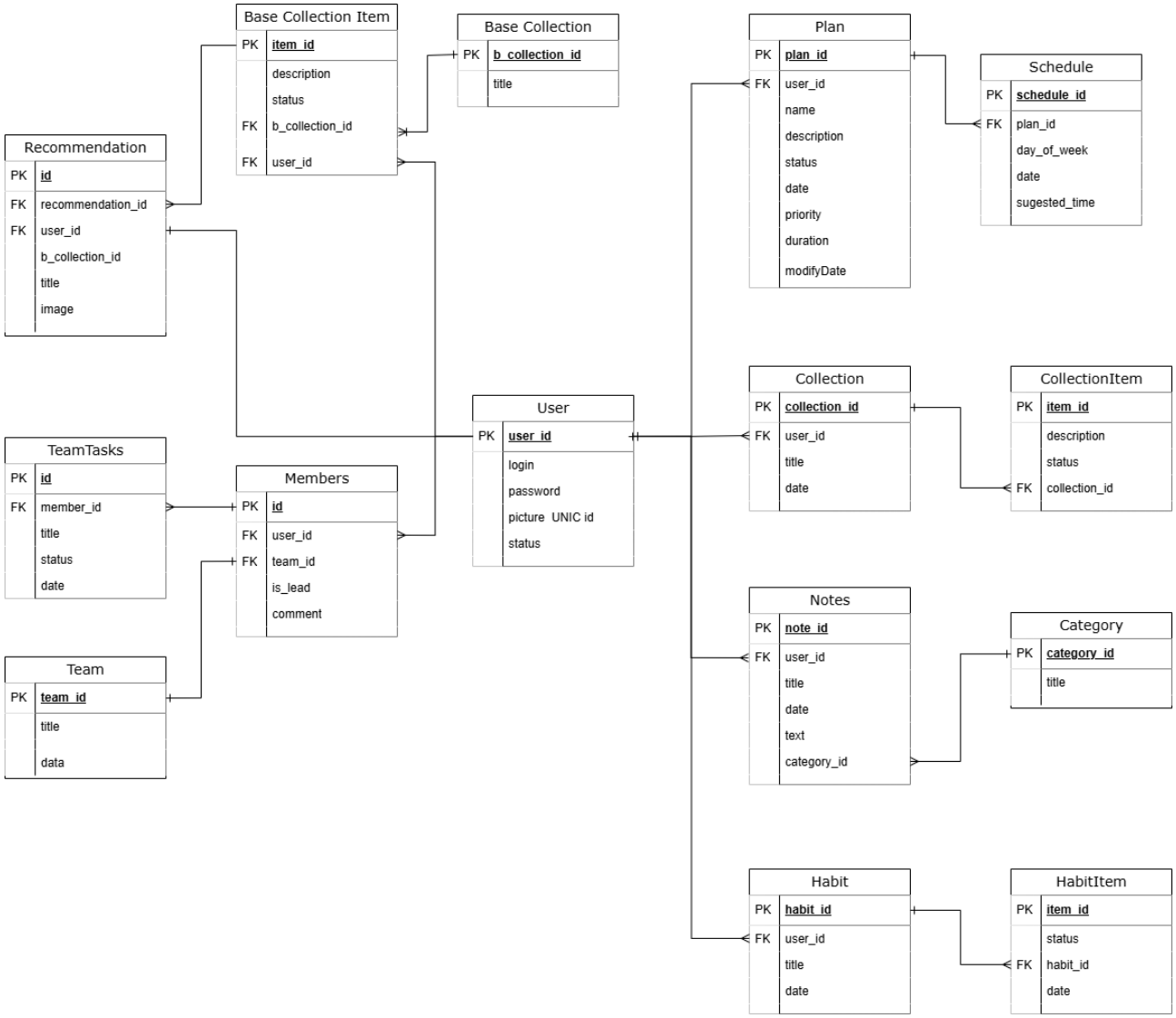
21) TypeScript: офіційний вебсайт TypeScript. [Електронний ресурс]. — Режим доступу: https://www.typescriptlang.org/

22) PostgreSQL: офіційний вебсайт PostgreSQL. [Електронний ресурс]. — Режим доступу: https://www.postgresql.org/

23) Spring Boot: офіційна документація Spring Boot. [Електронний ресурс]. — Режим доступу: https://spring.io/projects/spring-boot

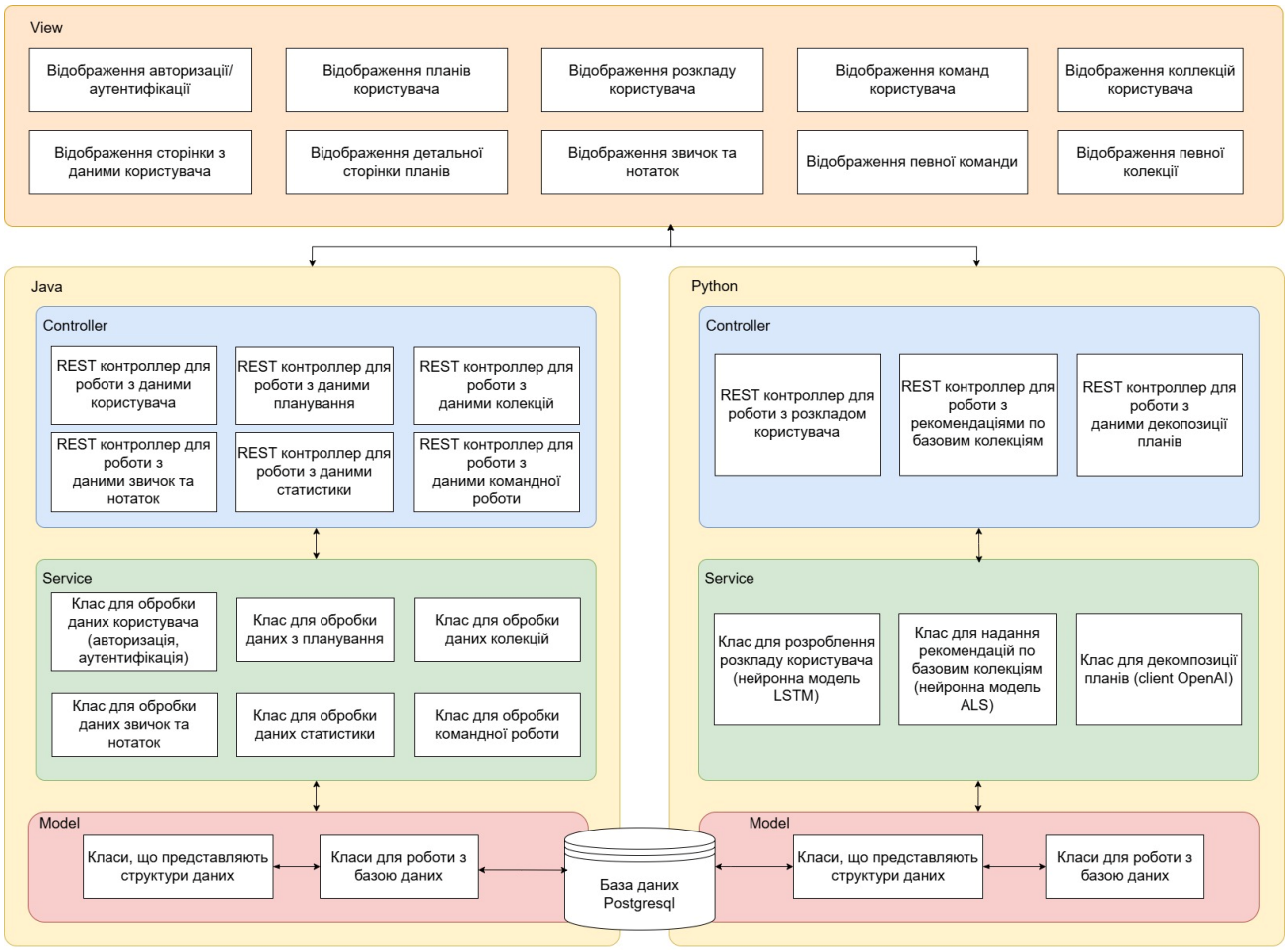
ДОДАТОК А

Схема бази даних



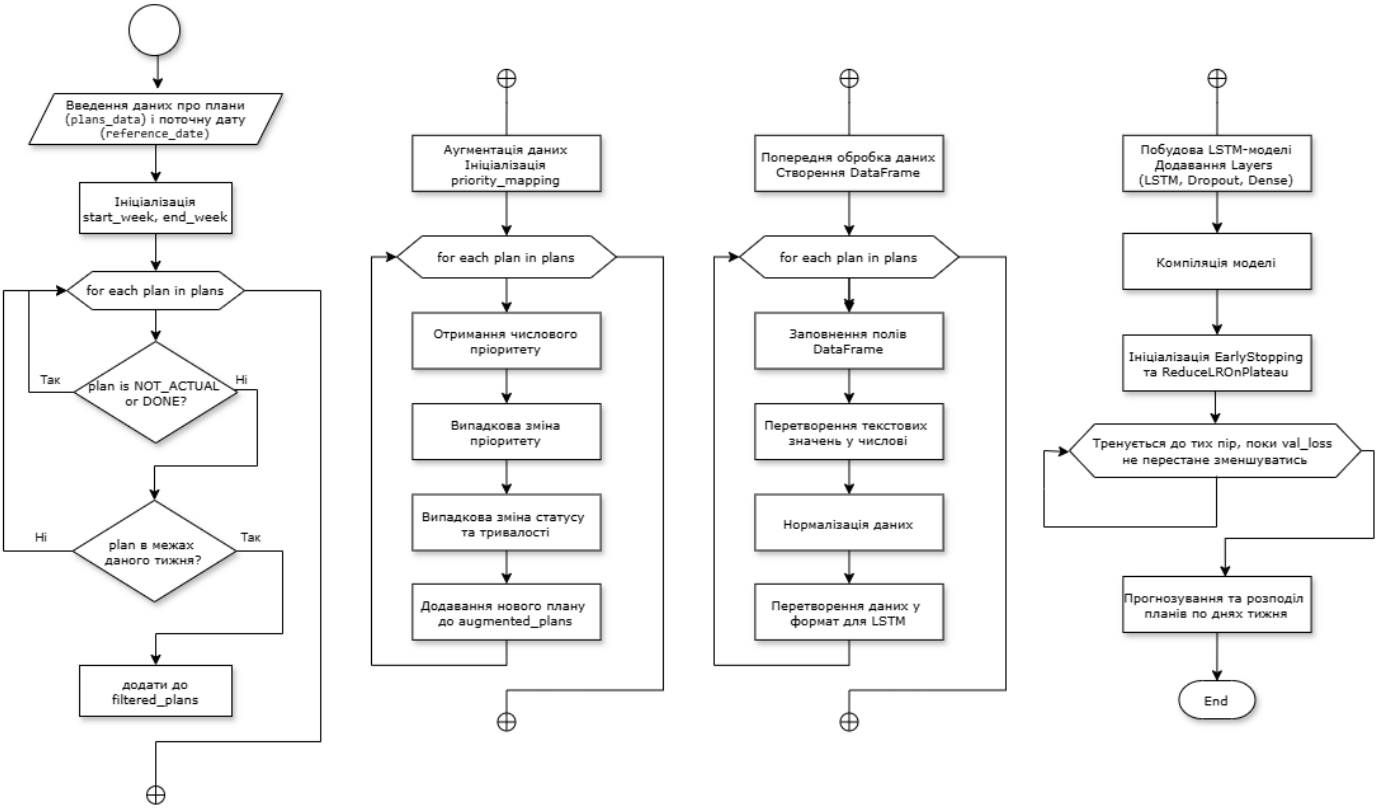
ДОДАТОК Б

UML діаграма класів



ДОДАТОК В

Алгоритм розподілу



ДОДАТОК Г

Лістинг програми

```
from flask import Blueprint, request, jsonify
from services.schedule_service import ScheduleService
from extensions import db

schedule_bp = Blueprint('schedule', __name__)
schedule_service = ScheduleService(db)

@schedule_bp.route('/generate-schedule', methods=['POST'])
def generate_schedule():
    data = request.json
    user_id = data.get('userId')
    date = data.get('date')

    if not user_id or not date:
        return jsonify({'error': 'Invalid input'}), 400

    result, error = schedule_service.generate_week_schedule(user_id, date)

    if error:
        return jsonify({'error': error}), 404

    return jsonify(result), 200

@schedule_bp.route('/get-schedule', methods=['POST'])
def get_schedule():
    data = request.json
    user_id = data.get('userId')
    date = data.get('date')

    if not user_id or not date:
        return jsonify({'error': 'Invalid input'}), 400

    result = schedule_service.get_schedule_by_date(user_id, date)

    return jsonify(result), 200


from datetime import timedelta, datetime
```

```
from repositories.plan_repository import PlanRepository
from repositories.schedule_repository import ScheduleRepository
from utils.lstm_model import predict_schedule


class ScheduleService:
    def __init__(self, db):
        self.plan_repo = PlanRepository(db)
        self.schedule_repo = ScheduleRepository(db)

    def get_schedule_by_date(self, user_id, date_str):
        week_start = self.get_week_start(date_str)
        week_end = week_start + timedelta(days=6)
        return self.schedule_repo.get_plans_with_schedule(week_end, user_id)

    def get_week_start(self, date_str):
        date = datetime.strptime(date_str, "%d.%m.%Y").date()
        return date - timedelta(days=date.weekday())

    def generate_week_schedule(self, user_id, date_str):
        week_start = self.get_week_start(date_str)
        week_end = week_start + timedelta(days=6)

        plans = self.plan_repo.get_plans_by_user_before_date(user_id, week_end)

        if not plans:
            return None, "No plans found for the user."

        schedule_predictions = predict_schedule(plans, week_end)
        schedule_predictions = {str(k): v for k, v in schedule_predictions.items()}

        schedule = self.schedule_repo.get_schedule_by_date(week_end, user_id)
        if schedule:
            self.schedule_repo.delete_schedule_by_date(week_end, user_id)

        for day, schedules in schedule_predictions.items():
            for schedule_data in schedules:
                plan_id = schedule_data['plan_id']
                date = week_end
                day_of_week = day
                self.schedule_repo.add_schedule(plan_id, date, day_of_week,
schedule_data['suggested_time'])

        return self.schedule_repo.get_plans_with_schedule(week_end, user_id), None
```

```
import random

import pandas as pd
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.layers import LSTM, Dense, Dropout, BatchNormalization
from tensorflow.keras.models import Sequential

from loger import Logger
from models.plan import Plan


def build_lstm_model(input_shape):
    model = Sequential([
        LSTM(64, return_sequences=True, input_shape=input_shape),
        Dropout(0.2),
        LSTM(32),
        Dropout(0.2),
        Dense(64, activation='relu'),
        BatchNormalization(),
        Dense(7, activation='softmax')
    ])
    model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
    return model


def augment_data(plans):
    priority_mapping = {'CRITICAL': 3, 'HIGH': 2, 'MEDIUM': 1, 'LOW': 0}
    augmented_plans = []
    for plan in plans:
        numeric_priority = priority_mapping[plan.priority]
        augmented_priority = max(0, min(3, numeric_priority + random.choice([-1, 0,
1])))
        augmented_priority_str = [k for k, v in priority_mapping.items() if v ==
augmented_priority][0]
        augmented_plan = Plan(
            plan_id=plan.plan_id,
            user_id=plan.user_id,
            name=plan.name,
            description=plan.description,
            status=random.choice(['IN_PROGRESS', 'DELAY', 'DONE']),
            creation_date=plan.creation_date,
            day_of_week=plan.day_of_week,
```

```
            priority=augmented_priority_str,
            modify_date=plan.modify_date,
            duration=plan.duration + random.randint(-1, 2),
            deadline=plan.deadline
        )
        augmented_plans.append(augmented_plan)
    return augmented_plans


def preprocess_lstm_data(plans, reference_date):
    data = pd.DataFrame([{
        'status': plan.status,
        'priority': plan.priority,
        'duration': plan.duration,
        'deadline_days_left': (plan.deadline - reference_date).days,
        'modify_hour': plan.modify_date.hour if plan.modify_date else 0
    } for plan in plans])
    data['status'] = data['status'].map({'IN_PROGRESS': 1, 'DELAY': 1,
'NOT_ACTUAL': 0, 'EMPTY': 0, 'DONE': 0})
    data['priority'] = data['priority'].map({'CRITICAL': 3, 'HIGH': 2, 'MEDIUM': 1,
'LOW': 0})
    data['duration'] /= 12
    data['modify_hour'] /= 24
    data['deadline_days_left'] /= 7
    return data.values.reshape((data.shape[0], 1, data.shape[1]))


def suggest_best_time(plans):
    time_of_day_count = {'morning': 0, 'afternoon': 0, 'evening': 0}
    for plan in plans:
        if plan.modify_date:
            hour = plan.modify_date.hour
            if 8 <= hour < 12:
                time_of_day_count['morning'] += 1
            elif 12 <= hour < 17:
                time_of_day_count['afternoon'] += 1
            elif 17 <= hour < 22:
                time_of_day_count['evening'] += 1
    best_time_of_day = max(time_of_day_count, key=time_of_day_count.get)

    return best_time_of_day


def suggest_least_productive_time(best_time):
    if best_time == 'morning':
```

```
        return 'evening'
    elif best_time == 'afternoon':
        return 'morning'
    elif best_time == 'evening':
        return 'afternoon'


from collections import defaultdict
import numpy as np


def distribute_schedule(plans, day_predictions, max_hours_per_day=12):
    week_schedule = defaultdict(list)
    day_load = np.zeros(7, dtype=int)
    best_time = suggest_best_time(plans)
    least_productive_time = suggest_least_productive_time(best_time)

    for plan, day in zip(plans, day_predictions):
        duration = plan.duration
        if plan.priority == 'CRITICAL':
            plan_time_suggestion = best_time
        elif plan.priority == 'MEDIUM':
            plan_time_suggestion = 'afternoon' if best_time == 'morning' else 'morning'
        else:
            plan_time_suggestion = least_productive_time

        for offset in range(7):
            current_day = (day + offset) % 7

            if day_load[current_day] + duration <= max_hours_per_day or offset == 6:
                week_schedule[current_day + 1].append({
                    'plan_id': plan.plan_id,
                    'name': plan.name,
                    'description': plan.description,
                    'status': plan.status,
                    'priority': plan.priority,
                    'duration': duration,
                    'deadline': plan.deadline.strftime('%Y-%m-%d'),
                    'suggested_time': plan_time_suggestion
                })
                day_load[current_day] += duration
                break

    empty_days = [i + 1 for i, load in enumerate(day_load) if load == 0]
    if empty_days:
```

```
        redistribute_tasks_to_empty_days(week_schedule, empty_days,
max_hours_per_day)

    return dict(week_schedule)


def redistribute_tasks_to_empty_days(schedule, empty_days,
max_hours_per_day):
    for day in empty_days:
        for existing_day in sorted(schedule.keys()):
            tasks = schedule[existing_day]
            for task in tasks:
                if task['priority'] == 'LOW' and task['duration'] <= max_hours_per_day:
                    schedule[day].append(task)
                    tasks.remove(task)
                    break
            if schedule[day]:
                break


def filter_plans(plans, reference_date):
    start_of_week = reference_date - pd.Timedelta(days=reference_date.weekday())
    end_of_week = start_of_week + pd.Timedelta(days=6)

    filtered_plans = [plan for plan in plans if
                      plan.status not in {'NOT_ACTUAL', 'DONE'} and start_of_week <=
plan.deadline <= end_of_week]

    filtered_plans.sort(key=lambda x: x.deadline)

    return filtered_plans


def predict_schedule(plans_data, reference_date, days_in_week=7,
max_hours_per_day=12):
    filtered_plans = filter_plans(plans_data, reference_date)
    augmented_plans = augment_data(filtered_plans)
    X = preprocess_lstm_data(augmented_plans, reference_date)
    model = build_lstm_model((X.shape[1], X.shape[2]))

    early_stopping = EarlyStopping(monitor='val_loss', patience=5)
    reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=3,
min_lr=0.001)

    y = np.eye(days_in_week)[np.random.choice(days_in_week, len(X))]
```

```
    model.fit(X, y, epochs=50, batch_size=16, validation_split=0.2,
              callbacks=[early_stopping, reduce_lr, Logger], verbose=0)

    y_pred = model.predict(X)
    day_blocks = np.argmax(y_pred, axis=1)

    week_schedule = distribute_schedule(filtered_plans, day_blocks,
max_hours_per_day)
    return week_schedule


        from extensions import db
        from models.plan import Plan
        from models.schedule import Schedule


        class ScheduleRepository:
            def __init__(self, db):
                self.db = db

            @staticmethod
            def add_schedule(plan_id, date, day_of_week, suggested_time):
                schedule = Schedule(plan_id=plan_id, date=date,
        day_of_week=day_of_week, suggested_time=suggested_time)
                db.session.add(schedule)
                db.session.commit()
                return schedule

            @staticmethod
            def get_schedule_by_date(date, user_id):
                return Schedule.query.join(Plan, Schedule.plan_id == Plan.plan_id) \
                    .filter(Schedule.date == date, Plan.user_id == user_id) \
                    .all()

            @staticmethod
            def delete_schedule_by_date(date_value, user_id):
                schedules = Schedule.query.join(Plan, Schedule.plan_id == Plan.plan_id)
        \
                    .filter(Schedule.date == date_value, Plan.user_id == user_id) \
                    .all()
                if schedules:
                    for schedule in schedules:
                        schedule = db.session.merge(schedule)
                        db.session.delete(schedule)
                    db.session.commit()
```

```
            return True
        return False

    @staticmethod
    def get_plans_with_schedule(date, user_id, days_of_week=None):
        query = db.session.query(
            Plan.plan_id,
            Plan.user_id,
            Plan.status,
            Plan.name,
            Plan.priority,
            Plan.deadline,
            Plan.duration,
            Schedule.day_of_week,
            Schedule.suggested_time
        ).join(Schedule, Plan.plan_id == Schedule.plan_id) \
            .filter(Schedule.date == date, Plan.user_id == user_id, Plan.status !=
"DONE", Plan.status != "NOT_ACTUAL") \
            .order_by(Schedule.day_of_week)

        results = query.all()

        result_dicts = [
            {
                "planId": row.plan_id,
                "userId": row.user_id,
                "status": row.status,
                "name": row.name,
                "priority": row.priority,
                "deadline": row.deadline,
                "dayOfWeek": row.day_of_week,
                "duration": row.duration,
                "suggestedTime": row.suggested_time,
            }
            for row in results
        ]
        return result_dicts


from flask import Blueprint, request, jsonify

from repositories.recommendation_repository import
RecommendationRepository
from services.recommendation_service import RecommendationService
from extensions import db
```

```
recommendation_bp = Blueprint('recommendation', __name__)
recommendation_service = RecommendationService(db)

@recommendation_bp.route('/generate-recommendations',
methods=['POST'])
def get_recommendations():
    data = request.json
    collection_id = str(data.get('collectionId'))
    watched_ids = data.get('watchedIds')
    user_id = data.get('userId')

    if not collection_id or not watched_ids:
        return jsonify({'error': 'Invalid input'}), 400

    recommendations, error =
recommendation_service.get_recommendations(collection_id, watched_ids,
user_id)

    if error:
        return jsonify({'error': error}), 404

    return jsonify(recommendations), 200

@recommendation_bp.route('/films-data', methods=['POST'])
def get_films_data():
    data = request.json
    collection_id = '2435466'
    limit = data.get('limit', 10)
    offset = data.get('offset', 0)
    search = data.get('search', '').strip()
    print(search)

    items, error = recommendation_service.get_items_from_collection(
        collection_id=collection_id,
        limit=limit,
        offset=offset,
        search=search
    )

    print(error)

    if error:
        return jsonify({'error': error}), 404
```

```
    return jsonify(items), 200



@recommendation_bp.route('/books-data', methods=['POST'])
def get_books_data():
    data = request.json
    collection_id = '9875768'
    limit = data.get('limit', 10)
    offset = data.get('offset', 0)
    search = data.get('search', '').strip()

    items, error = recommendation_service.get_items_from_collection(
        collection_id=collection_id,
        limit=limit,
        offset=offset,
        search=search
    )

    if error:
        return jsonify({'error': error}), 404

    return jsonify(items), 200


@recommendation_bp.route('/get-user-recommendations', methods=['POST'])
def get_user_recommendations():
    data = request.json
    collection_id = data.get('collectionId')
    user_id = data.get('userId')

    if not collection_id or not user_id:
        return jsonify({'error': 'Invalid input'}), 400

    recommendations =
RecommendationRepository.get_recommendations_by_user_and_collection(c
ollection_id, user_id)

    recommendations_list = [{
        'id': recommendation.id,
        'collectionId': recommendation.collection_id,
        'userId': recommendation.user_id,
        'recommendationId': recommendation.recommendation_id,
        'title': recommendation.title,
        'image': recommendation.image
```

```
    } for recommendation in recommendations]

    return jsonify(recommendations_list), 200


import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.preprocessing import MultiLabelBinarizer

from repositories.recommendation_repository import
RecommendationRepository
from utils.als_model import ALSModel


class RecommendationService:

    def __init__(self, db):
        self.db = db
        self.recommendation_repo = RecommendationRepository(db)

        self.films_df = pd.read_csv('films.csv')
        self.books_df = pd.read_csv('books.csv', delimiter=';', encoding='utf-8')

        self.films_df = self.preprocess_data(self.films_df, 'film')
        self.books_df = self.preprocess_data(self.books_df, 'book')

    def preprocess_data(self, df, collection_type):
        if collection_type == 'film':
            df['Genres'] = df['Genres'].apply(lambda x: x.split(',') if isinstance(x,
str) else [])
            df = self.encode_genres(df, 'Genres')
        else:
            df['Category'] = df['Category'].apply(lambda x: x.split(',') if
isinstance(x, str) else [])
            df = self.encode_genres(df, 'Category')
        return df

    def encode_genres(self, df, column):
        mlb = MultiLabelBinarizer()
        df_genres_encoded = pd.DataFrame(mlb.fit_transform(df[column]),
columns=mlb.classes_, index=df.index)
        df = pd.concat([df, df_genres_encoded], axis=1)
        df.drop(columns=[column], inplace=True)
        return df
```

```
    def get_recommendations(self, collection_id, watched_ids, user_id):
        if collection_id == '2435466':
            df = self.films_df
            features = df.drop(columns=[
                'id', 'url', 'Name', 'PosterLink', 'Actors', 'Director',
                'Description', 'DatePublished', 'Keywords', 'ReviewAurthor',
                'ReviewDate', 'ReviewBody', 'duration'
            ])
            watched_items = df[df['id'].isin(watched_ids)]
        elif collection_id == '9875768':
            df = self.books_df
            features = df.drop(columns=[
                'id', 'Filename', 'PosterLink', 'Name',
                'Author', 'Category ID'
            ])
            watched_ids = list(map(str, watched_ids))
            watched_items = df[df['id'].astype(str).isin(watched_ids)]
        else:
            return None, "Invalid collection ID"

        if watched_items.empty:
            return None, "No valid watched items found"

        input_dim = features.shape[1]
        output_dim = input_dim
        self.als_model = ALSModel(input_dim=input_dim,
output_dim=output_dim)

        labels = features.copy()
        self.als_model.train_model(features, labels)

        embeddings = self.als_model.generate_embeddings(features)
        watched_indices = watched_items.index.to_numpy()
        watched_indices_tensor = tf.convert_to_tensor(watched_indices,
dtype=tf.int32)
        watched_embeddings = tf.gather(embeddings, watched_indices_tensor)

        similarity_scores = self.als_model.calculate_similarity(embeddings,
watched_embeddings)
        top_indices = tf.argsort(similarity_scores, axis=1,
direction='DESCENDING')[:, :10]
        recommended_items = [df.iloc[idx].to_dict() for idx in
top_indices.numpy().flatten()]
```

```
        self.recommendation_repo.delete_recommendations(int(collection_id),
user_id)

        for item in recommended_items[:10]:
            try:
                recommendation_id = item['id']

                if isinstance(recommendation_id, str) and
recommendation_id.isdigit():
                    recommendation_id = int(recommendation_id)
                elif not isinstance(recommendation_id, int):
                    print(f"Skipping invalid recommendation_id:
{recommendation_id}")
                    continue

                image = item.get('PosterLink')
                title = item.get('Name')

                self.recommendation_repo.add_recommendations(
                    collection_id=int(collection_id),
                    user_id=user_id,
                    recommendation_id=recommendation_id,
                    image=image,
                    title=title
                )
            except (KeyError, ValueError) as e:
                print(f"Error saving recommendation: {e}")

        recommendations_list = [{
            'collectionId': collection_id,
            'userId': user_id,
            'recommendationId': recommendation['id'],
            'title': recommendation.get('Name'),
            'image': recommendation.get('PosterLink')
        } for recommendation in recommended_items[:10]]

        return recommendations_list, None

    def get_items_from_collection(self, collection_id, limit=10, offset=0,
search=''):

        if collection_id == '2435466':
            df = self.films_df
        elif collection_id == '9875768':
            df = self.books_df
```

```
        else:
            return None, "Invalid collection ID"

        if search:
            df = df[df['Name'].str.contains(search, case=False, na=False)]

        items = df[['id', 'Name']].iloc[offset:offset + limit]
        items = items[items['id'].apply(lambda x: str(x).isdigit())]

        items['id'] = items['id'].astype(np.int64)

        items_list = items.to_dict(orient='records')

        return items_list, None



import tensorflow as tf

from loger import Logger2


class ALSModel:

    def __init__(self, input_dim, output_dim):
        self.model = tf.keras.Sequential([
            tf.keras.layers.Dense(256, activation='relu',
input_shape=(input_dim,)),
            tf.keras.layers.BatchNormalization(),
            tf.keras.layers.Dropout(0.3),
            tf.keras.layers.Dense(128, activation='relu'),
            tf.keras.layers.BatchNormalization(),
            tf.keras.layers.Dense(64, activation='relu'),
            tf.keras.layers.Dense(output_dim, activation='relu')
        ])
        self.model.compile(optimizer='adam', loss='mse', metrics=['accuracy'])

    def train_model(self, features, labels, epochs=5, batch_size=100):
        history = self.model.fit(
            features, labels,
            epochs=epochs,
            batch_size=batch_size,
            validation_split=0.2,
            verbose=0,
            callbacks=[Logger2]
```

```
        )

        return history

    @tf.function
    def generate_embeddings(self, features):
        features = tf.cast(features, dtype=tf.float32)
        return self.model(features)

    @tf.function
    def calculate_similarity(self, embeddings, watched_embeddings):
        embeddings_normalized = tf.nn.l2_normalize(embeddings, axis=1)
        watched_embeddings_normalized =
tf.nn.l2_normalize(watched_embeddings, axis=1)
        similarity_scores = tf.matmul(watched_embeddings_normalized,
embeddings_normalized, transpose_b=True)
        return similarity_scores


from extensions import db
from models.recommendation import Recommendation


class RecommendationRepository:
    def __init__(self, db):
        self.db = db

    @staticmethod
    def delete_recommendations(collection_id, user_id):
        print(f"Deleting recommendations where collection_id={collection_id}
and user_id={user_id}")

        query = f"DELETE FROM recommendation WHERE collection_id =
{collection_id} AND user_id = {user_id}"

        result = db.session.execute(query)

        db.session.commit()

        print(f"Rows deleted: {result.rowcount}")

        db.session.close()

        if db.session.is_active:
            print("Session is still active.")
```

```
        else:
            print("Session is closed.")

        return result.rowcount

    @staticmethod
    def add_recommendations(collection_id, user_id, recommendation_id, title,
image=None):
        recommendation = Recommendation(
            collection_id=collection_id,
            user_id=user_id,
            recommendation_id=recommendation_id,
            title=title,
            image=image
        )
        db.session.add(recommendation)
        db.session.commit()

    @staticmethod
    def get_recommendations_by_user_and_collection(collection_id, user_id):
        recommendations = Recommendation.query.filter_by(
            collection_id=collection_id,
            user_id=user_id
        ).all()

        return recommendations


from flask import Blueprint, request, jsonify
from services.decomposition_service import DecompositionService
from extensions import db

decomposition_bp = Blueprint('decomposition', __name__)
decomposition_service = DecompositionService(db)

@decomposition_bp.route('/decompose-task', methods=['POST'])
def decompose_task():
    data = request.json

    planId = data.get('planId')
    task_description = data.get('name')

    if not task_description:
        return jsonify({'error': 'Invalid input'}), 400
```

```
    steps, error = decomposition_service.decompose_task(task_description,
planId)
    print(error)

    if error:
        return jsonify({'error': error}), 500

    return jsonify(steps), 200


import openai

from repositories.plan_repository import PlanRepository


class DecompositionService:
    def __init__(self, db):
        self.plan_repo = PlanRepository(db)

    def decompose_task(self, task_description, plan_id, min_points=6,
max_points=10):
        try:
            openai.api_key = 'sk-proj-44r9K0FsBWn3ylxIpdL-
wHF64sacIjyKap5dqEQ0UbtB1v3iFMGEs8QgS7DlbuPDxTyDPmi3JKT3Bl
bkFJesVLtN5YejEOU60vEK_RbFnvP4QO5xayplK6M0qqtzeWDUGRt96cH
M8-vC0wOy34Y6_C-yPvgA'

            response = openai.ChatCompletion.create(
                model="gpt-3.5-turbo",
                messages=[
                    {
                        "role": "user",
                        "content": f"Decompose the task '{task_description}' into
{min_points} to {max_points} steps. "
                    }
                ]
            )

            message = response['choices'][0]['message']['content']
            steps = message.strip().split('\n')

            decomposition_text = "\n".join(steps)

            updated_plan = self.plan_repo.update_plan_decomposition(plan_id,
decomposition_text)
```

```
            if not updated_plan:
                return [], f"Plan with ID {plan_id} not found."

            plan_data = {
                "planId": updated_plan.plan_id,
                "userId": updated_plan.user_id,
                "status": updated_plan.status,
                "name": updated_plan.name,
                "description": updated_plan.description,
                "duration": updated_plan.duration,
                "priority": updated_plan.priority,
                "deadline": updated_plan.deadline,
                "dayOfWeek": updated_plan.day_of_week,
                "decomposition": updated_plan.decomposition
            }

            return plan_data, None

        except Exception as e:
            return [], str(e)

from models.plan import Plan


class PlanRepository:
    def __init__(self, db):
        self.db = db

    def get_plans_by_user_before_date(self, user_id, date):
        return Plan.query.filter(Plan.user_id == user_id, Plan.deadline <=
date).all()

    def update_plan_decomposition(self, plan_id, decomposition):
        plan = Plan.query.filter_by(plan_id=plan_id).first()
        if not plan:
            return None

        print(plan.decomposition)
        plan.decomposition = decomposition
        print(plan.decomposition)
        self.db.session.commit()

        return plan
```

ДОДАТОК Д

Результати перевірки роботи на співпадіння

StrikePlagiarism.com | National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute | Дата звіту 12/4/2024 | Дата редагування 12/4/2024 | Документ прийнятий

Звіт подібності

метадані

Заголовок
ІП-32мп_Куржумова_ПЗ

Науковий керівник / Експерт

Автор **Родіонов П.Ю.**

підрозділ
ФІОТ, К-а інформатики та програмної інженерії

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	·	1
Білі знаки	ß	0
Парафрази (SmartMarks)	a	41

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

10	10490	80608
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	2023_61260001_Shybunko_Mariia_Ivanivna_227325 [redacted] 11/21/2024 National University "Lviv Politechnika" (National University Lviv Politechnika)	81	0.77 %
2	Романко В.В._ІТІС-21_робота [redacted] 10/30/2024 National University "Lviv Politechnika" (NULP2)	57	0.54 %
3	Романко В.В._ІТІС-21_робота [redacted] 10/30/2024 National University "Lviv Politechnika" (NULP2)	43	0.41 %

4	2023_61260001_Shybunko_Mariia_Ivanivna_227325 [redacted] 11/21/2024 National University "Lviv Politechnika" (National University Lviv Politechnika)	40	0.38 %
5	Романко В.В._ІТІС-21_робота [redacted] 10/30/2024 National University "Lviv Politechnika" (NULP2)	40	0.38 %
6	Романко В.В._ІТІС-21_робота [redacted] 10/30/2024 National University "Lviv Politechnika" (NULP2)	36	0.34 %
7	2023_61260001_Shybunko_Mariia_Ivanivna_227325 [redacted] 11/21/2024 National University "Lviv Politechnika" (National University Lviv Politechnika)	29	0.28 %
8	ФКНТ_2024_121_магістр_Шепеть_Туча_С_В [redacted] 11/24/2024 Ukrainian national aviation university (ФКНТ Кафедра інженерії програмного забезпечення)	26	0.25 %
9	2023_61260001_Shybunko_Mariia_Ivanivna_227325 [redacted] 11/21/2024 National University "Lviv Politechnika" (National University Lviv Politechnika)	25	0.24 %
10	Романко В.В._ІТІС-21_робота [redacted] 10/30/2024 National University "Lviv Politechnika" (NULP2)	24	0.23 %

з домашньої бази даних (0.20 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	Бакалаври СП ІПСА 2024 [redacted] 6/19/2024 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ІПСА, К-ра системного проектування)**	**21 (1)**	**0.20 %**

з програми обміну базами даних (4.58 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	2023_61260001_Shybunko_Mariia_Ivanivna_227325 [redacted] 11/21/2024 **National University "Lviv Politechnika" (National University Lviv Politechnika)**	**226 (9)**	**2.15 %**
2	Романко В.В._ІТІС-21_робота [redacted] 10/30/2024 **National University "Lviv Politechnika" (NULP2)**	**212 (6)**	**2.02 %**
3	ФКНТ_2024_121_магістр_Шепеть_Туча_С_В [redacted] 11/24/2024 **Ukrainian national aviation university (ФКНТ Кафедра інженерії програмного забезпечення)**	**42 (2)**	**0.40 %**

з Інтернету (0.22 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	**https://ir.nmu.org.ua/bitstream/handle/123456789/164833/23_06_22_%D0%92%D0%BE%D0%BB%D0%BE%D1%88%D0%BA%D0%BE%20%D0%9C.%D0%9C.pdf?sequence=1**	**12 (1)**	**0.11 %**