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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:
Метод і програмний засіб рекомендацій рекреаційних зон

Виконав: студент II курсу
групи 1ПІ-22М спеціальності
121 – Інженерія програмного забезпечення
Яровий Ігор Олексійович

Керівник: к.т.н., асистент каф. ПЗ Тужанський С.Є.
«08» грудня 2023 р.

Опонент: к.т.н., доц. каф. ОТ Кожем'яко А. В.
«08» грудня 2023 р.

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

ВНТУ – 2023

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«19» вересня 2023 р.

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

Яровому Ігорю Олексійовичу

1. Тема роботи – Метод і програмний засіб рекомендацій рекреаційних зон.

Керівник роботи: Тужанський Станіслав Євгенович, к.т.н., асистент кафедри ПЗ, затверджені наказом вищого навчального закладу від «18» вересня 2023 р. № 247.

2. Строк подання студентом роботи – 5 грудня 2023 р.

3. Вихідні дані до роботи: середовище розробки XCode 15 та Visual Studio Code, мови програмування Swift та Python, операційна система MacOS, сумісна фільтрація, фільтрація на основі вмісту, подання алгоритму TF-IDF та його модифікацій.

4. Зміст текстової частини: вступ; аналіз та постановка задачі; перелік критеріїв для формування рекомендацій; проектування архітектури системи; розробка алгоритмів та методів системи; розробка модуля оновлення датасету; розробка методу навчання моделі; розробка методу прогнозування рекомендацій; хостинг проекту; розробка клієнтської частини системи; тестування системи; економічна частина; висновки; перелік посилань; додатки.

5. Перелік ілюстративного матеріалу: блок-схеми алгоритмів роботи програмного засобу; тестування додатку.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Тужанський С.Є., асистент кафедри ПЗ	19.09.2023	05.12.2023
5	Причепа І.В., к.е.н., доцент каф. ЕПВМ	25.11.2023	01.12.2023

7. Дата видачі завдання 19 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи	Примітка
1	Аналіз систем рекомендацій	20.09.2023-09.10.2023	вик
2	Розробка методу навчання моделі	10.10.2023-29.10.2023	вик
3	Розробка методу рекомендацій	30.10.2023-14.10.2023	вик
4	Розробка клієнтської частини системи	15.11.2023-24.11.2023	вик
5.	Економічна частина	25.11.2023-05.12.2023	вик

Студент _____ **Яровий І.О.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Тужанський С.Є.**
(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Стрімкий ритм життя та розвиток урбаністичного середовища вимагають сучасного підходу до планування дозвілля. У цьому контексті рекомендаційні інформаційні системи виступають як важливий інструмент у підвищенні якості життя мешканців та гостей міста, пропонуючи оптимальні варіанти для відпочинку, спорту та інших видів активностей на основі їхніх індивідуальних уподобань.

З огляду на стрімке зростання цифрових технологій, рекомендаційні системи дедалі більше впроваджуються в повсякденне життя. Про це свідчить успіх персоналізованих цифрових сервісів Netflix, Spotify та інших, де використання рекомендаційних алгоритмів забезпечує високий рівень задоволення користувачів [1].

Така ж потреба існує у сфері рекреаційних послуг, де можливість отримувати персоналізовані пропозиції щодо місць відпочинку та культурних подій має значно підвищити ефективність використання урбаністичного простору та забезпечити кращий сервіс та досвід користувачів [2].

Таким чином, дослідження та розробка рекомендаційних інформаційних систем для рекреаційних зон є актуальними задачами сьогодення. Це не лише сприяє покращенню досвіду користувачів, але й відкриває нові можливості для оптимізації використання міських ресурсів, підвищення загальної продуктивності та ефективності урбаністичних послуг. В цьому контексті, важливим є дослідження сучасних тенденцій, методів та технологій, що застосовуються у рекомендаційних системах, з метою їх подальшого вдосконалення та інноваційного розвитку.

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

Мета та завдання дослідження. Метою дослідження є розробка ефективної інформаційної системи рекомендацій рекреаційних зон міста, яка

сприятиме підвищенню якості та комфорту відпочивальників, враховуючи їх індивідуальні потреби та уподобання.

Для досягнення поставленої мети було визначено такі основні завдання:

- провести аналіз існуючих методів і засобів рекомендаційних систем для визначення напрямків підвищення їх продуктивності та точності;
- удосконалити метод прогнозування рекомендацій та метод навчання моделі;
- визначити перелік критеріїв для формування рекомендацій;
- розробити веб-сервіс для прогнозування рекомендацій;
- розробити мобільний клієнт для роботи з модулем прогнозування рекомендацій;
- провести тестування програмного додатку та експериментальні дослідження розробленого програмного засобу.

Об’єктом дослідження є процес формування рекомендацій щодо рекреаційних зон міста.

Предметом дослідження є методи та алгоритми, які використовуються у рекомендаційній системі.

Методи дослідження. У процесі досліджень використовувались методи дослідження:

- методи побудови та навчання нейронних мереж для побудови нейронної мережі;

- методи статистичного моделювання для визначення вихідного шару нейронної мережі;

- методи обробки та моделювання природної мови для побудови числової моделі тексту;

- методи регуляризації нейронних мереж для підвищення надійності роботи мережі.

Наукова новизна отриманих результатів.

1. Удосконалено метод навчання моделі для рекомендаційних систем, в якому на відміну від існуючих після обчислення частотності терміну (TF), зворотної частотності документів (IDF), а також їхнього добутку TF-IDF з відповідною вагою для кожного з термінів, виконується обчислення косинусної схожості векторів ознак рекреаційних зон. Це дозволило більш точно сформувати модель надання рекомендацій із оптимальним використанням системних ресурсів.

2. Отримав подальшого розвитку метод прогнозування рекомендацій для користувача, особливостями якого є попередня ініціалізація запропонованого удосконаленого модуля навчання моделі з визначенням критеріїв для проведення аналізу та формування рекомендацій на основі актуалізованої моделі навчання. Це дозволило отримати більш високий рівень персоналізації рекомендацій користувача, підвищити точність системи рекомендацій, а також забезпечити її адаптивність та здатність інтеграції з зовнішніми модулями.

Практична цінність отриманих результатів. Практичне значення магістерської кваліфікаційної роботи полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби для отримання рекомендацій щодо вибору рекреаційних зон.

Особистий внесок здобувача. Усі етапи дослідження та розробки, представлені у магістерській роботі, були виконані автором особисто. Автор взяв активну участь у формулюванні основних ідей, концепцій та гіпотез дослідження, а також у плануванні та організації робочого процесу. Автору належать такі результати: алгоритм обміну повідомлень; алгоритм оновлення датасету; алгоритм навчання моделі; метод прогнозування рекомендацій.

Апробація матеріалів магістерської кваліфікаційної роботи. Результати магістерської кваліфікаційної роботи доповідалися та обговорювалися на міжнародній науково-практичній інтернет-конференції

«Електронні інформаційні ресурси: створення, використання, доступ» (Вінниця, 2023).

Публікації. Основні результати дослідженні опубліковані у тезах доповіді на міжнародній науково-практичній інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ».

Структура та обсяг роботи. Магістерська кваліфікаційна робота складається зі вступу, п’яти розділів, висновків, списку літератури, що містить 20 найменування, 10 додатки. Робота містить 48 ілюстрацію, 9 таблиць.

1. АНАЛІЗ СУЧАСНОГО СТАНУ ПИТАННЯ ТА ОБҐРУНТУВАННЯ ЗАДАЧІ

1.1. Основні поняття

Рекомендаційна система є підтипом систем фільтрації даних, мета якої - передбачити «рейтинг» або «перевагу», яку користувач повинен надати товару.

Рекомендаційні системи використовуються в різноманітних сферах, але найбільш широко відомі як генератори списків відтворення для відео та музичних сервісів, таких як Netflix, YouTube та Spotify, рекомендації щодо продуктів таких служб, як Amazon, або як рекомендатори контенту для платформ соціальних мереж, таких як Facebook та Twitter. Такі системи можуть працювати, використовуючи один вхід для даних, наприклад музику, або декілька входів всередині і між платформами, такі як новини, книги і пошукові запити. Існують також популярні рекомендаційні системи для конкретних тем, такі як ресторани або онлайн-знайомства [1], [2], [3].

В рекомендаційних системах зазвичай використовується або сумісна фільтрація або фільтрація на основі вмісту. Підходи до спільної фільтрації будують модель на основі минулої поведінки користувача (предмети, придбані раніше або вибрані та / або числові рейтинги, що даються цим елементам), а також подібних рішень, прийнятих іншими користувачами. Потім ця модель використовується для прогнозування елементів (або рейтингів предметів), які користувача можуть зацікавити. Підходи до фільтрації на основі вмісту використовують ряд дискретних, попередньо позначених характеристик елемента, щоб рекомендувати додаткові елементи зі схожими властивостями. Поточні системи рекомендацій зазвичай поєднують один або кілька підходів у гібридну систему.

1.2. Сумісна фільтрація

Колаборативна фільтрація – це техніка, яку застосовують у системах рекомендацій. Цей метод базується на автоматичному передбаченні інтересів користувача, зібраних на основі уподобань інших користувачів. Головна ідея колаборативної фільтрації полягає в тому, що якщо одна особа (А) ділиться подібними думками з іншою особою (Б) щодо певного питання, то ймовірно, що А матиме схожі погляди з Б з приводу іншого питання, у порівнянні з думками випадкової особи [1], [4].

Основною перевагою методу колаборативної фільтрації є його незалежність від аналізу контенту, дозволяючи ефективно рекомендувати складні об'єкти, як-от фільми, без необхідності "розуміння" самого об'єкта.

Методи колаборативної фільтрації зазвичай зіштовхуються з трьома основними проблемами: проблемою холодного старту, проблемами масштабування та проблемою розрідженості даних.

Холодний старт виникає, коли для нового користувача або продукту існує недостатня кількість даних, аби забезпечити точні рекомендації.

Проблема масштабованості полягає в тому, що в багатьох середовищах, де застосовуються рекомендаційні системи, існують мільйони користувачів та продуктів, що вимагає значних обчислювальних ресурсів для генерації рекомендацій.

Проблема розрідженості виникає через те, що велика кількість продуктів на сайтах електронної комерції часто отримує оцінки лише від малої частини користувачів. Таким чином, навіть популярні товари можуть мати дуже обмежену кількість рейтингів.

Соціальні мережі первісно використовували колаборативну фільтрацію для рекомендації нових друзів, груп та інших соціальних зв'язків, аналізуючи мережеві зв'язки між користувачем та їхніми друзями.

1.3. Фільтрація на основі вмісту

Фільтрація на основі вмісту (Content-based filtering) є іншим популярним підходом у створенні рекомендаційних систем. Вона заснована на аналізі опису елемента та профілю уподобань користувача. Цей метод особливо ефективний у випадках, коли доступні дані про продукт (наприклад, ім'я, місцезнаходження, опис), але відсутні дані про користувача.

У системах, що базуються на вмісті, процес рекомендації розглядається як задача класифікації, специфічна для кожного користувача. Вони навчають класифікатор, щоб визначити, які елементи можуть сподобатися або не сподобатися користувачеві, засновуючись на характеристиках продукту.

У таких системах ключові слова використовуються для опису елементів, тоді як профіль користувача створюється для ідентифікації типу товарів, що він зазвичай вважає за краще. Іншими словами, ці алгоритми прагнуть рекомендувати продукти, схожі на ті, які користувач раніше оцінив позитивно або які він вивчає на даний момент. Важливо, що цей підхід не вимагає активного входу від користувача для створення його профілю. Замість цього, потенційні продукти порівнюються з тими, які раніше оцінив користувач, та рекомендуються ті, що найкраще відповідають його уподобанням [1], [5].

Для створення профілю користувача система в основному фокусується на двох типах інформації:

1. Модель уподобань користувача.

2. Історія взаємодії користувача з системою рекомендацій.

Методи фільтрації на основі вмісту використовують профіль елемента, який складається з різних дискретних атрибутів та ознак, що характеризують елемент у системі. Для представлення особливостей елементів використовується алгоритм подання елементів, серед яких поширеним є подання за допомогою tf-idf (подання векторного простору).

Система створює контентний профіль користувачів, використовуючи зважений вектор ознак елементів. Вагові коефіцієнти визначають важливість

кожної характеристики для користувача, які можна обчислити за допомогою різних методів, включаючи середні значення оціненого вектора елемента або більш складних методів машинного навчання, таких як байєсівські класифікатори, кластерний аналіз, дерева рішень та штучні нейронні мережі.

Основна проблема фільтрації на основі вмісту полягає у здатності системи переносити уподобання користувачів з одного типу вмісту на інший. Якщо система обмежена рекомендаціями лише того типу вмісту, який вже споживає користувач, цінність рекомендаційних систем зменшується. Наприклад, рекомендація новин на основі перегляду новин корисна, але більш цінною була б можливість рекомендувати музику, відео, товари та інші типи вмісту з різних джерел на основі перегляду новин. Для подолання цієї обмеженості більшість сучасних систем рекомендацій використовують гібридні підходи.

1.4. Багатокритеріальні системи рекомендацій

Багатокритеріальні системи рекомендацій розширюють традиційний підхід рекомендацій, включаючи інформацію про переваги користувачів за різними критеріями. На відміну від систем, які базуються на одному критерії для визначення загальної переваги користувача U щодо елемента I, багатокритеріальні системи враховують різні аспекти або критерії, що формують загальне уподобання. Таким чином, ці системи намагаються передбачити рейтинг невивчених предметів U, використовуючи багатовимірні дані про переваги, що дозволяє забезпечити більш точні та персоналізовані рекомендації [4], [6].

1.5. Системи рекомендацій з урахуванням ризиків

Більшість існуючих рекомендаційних систем зосереджують увагу на наданні найбільш релевантного вмісту користувачам, використовуючи

контекстну інформацію, але часто не враховують ризик створення незручностей для користувачів через небажані сповіщення. Важливо враховувати можливість засмучувати користувачів, надсилаючи рекомендації у недоречний час, наприклад, під час робочих зустрічей, рано вранці або пізно вночі. Таким чином, ефективність рекомендаційної системи частково залежить від її здатності уникати ризиків та неприємностей для користувачів під час процесу рекомендації [5], [7]

1.6. Мобільні рекомендаційні системи

Мобільні рекомендаційні системи використовують смартфони з доступом до Інтернету для надання персоналізованих контекстно-залежних рекомендацій. Це складна дослідницька сфера, оскільки мобільні дані є більш складними порівняно з даними, з якими зазвичай працюють традиційні рекомендаційні системи. Мобільні дані характеризуються своєю неоднорідністю, шумністю, необхідністю просторової та часової автокореляції та складностями у валідації та загальному застосуванні [6], [8].

Три основні фактори, які впливають на мобільні рекомендаційні системи та точність їхніх прогнозів, включають контекст, метод рекомендацій та питання конфіденційності. Також існує проблема трансплантації: рекомендації можуть не бути актуальними або застосовними у всіх регіонах. Наприклад, рекомендувати рецепт у місцевості, де важко або неможливо знайти потрібні інгредієнти, було б непрактично. Такі системи мають враховувати різноманітність місцевих умов та доступності ресурсів для ефективного надання персоналізованих рекомендацій.

1.7. Гібридні рекомендаційні системи

Сучасні рекомендаційні системи часто використовують гібридний підхід, який об'єднує різні методи, такі як колаборативна фільтрація, фільтрація на

основі вмісту та інші стратегії. Такий гібридний підхід може бути реалізований кількома способами, включаючи окреме використання методів на основі вмісту та співпраці з подальшим їх комбінуванням, інтеграцію можливостей одного підходу в інший, або розробку єдиної моделі, яка включає обидва підходи [8], [9], [10].

Емпіричні дослідження показали, що гібридні методи можуть забезпечити більш точні рекомендації порівняно з чистими методами, які використовують тільки колаборативну фільтрацію або фільтрацію на основі вмісту. Гібридні системи також можуть ефективно вирішувати загальні проблеми рекомендаційних систем, такі як холодний старт та розрідженість даних. Вони також допомагають подолати обмеження у підходах, заснованих на знаннях, зокрема складнощі пов'язані з інженерією знань.

1.8. Приклади застосування рекомендаційних систем

1.8.1. Last.fm

Last.fm є інтернет-проектом, зосередженим на музиці, який надає унікальну функціональність для збору інформації про музичні вподобання користувачів і створення індивідуальних та загальних музичних чартів. Сервіс Last.fm використовує колаборативну фільтрацію для створення персоналізованих "станцій" з рекомендованими піснями, аналізуючи, які групи та пісні користувач слухає регулярно, і порівнюючи цю інформацію з музичними перевагами інших користувачів [11].

На сайті також працює система тегування, де користувачі можуть маркувати альбоми, пісні та виконавців за своїм смаком. Це дозволяє слухати музику на основі певних тегів.

Для збору статистики та трансляції музики з сайту, а також для розміщення тегів, використовується програма Last.fm, доступна для різних операційних систем. Ця програма також надає додаткову інформацію, таку як

біографії музикантів, фотографії, обкладинки альбомів, і може використовуватися для ведення особистого блогу. Статистика, зібрана програмою, відправляється на сайт, коли з'являється з'єднання з Інтернетом. Постійне з'єднання з Інтернетом потрібне для доступу до біографій, зображень, тегування та прослуховування музики з архівів сайту.

Головна сторінка сервісу Last.fm показана на рисунку 1.1

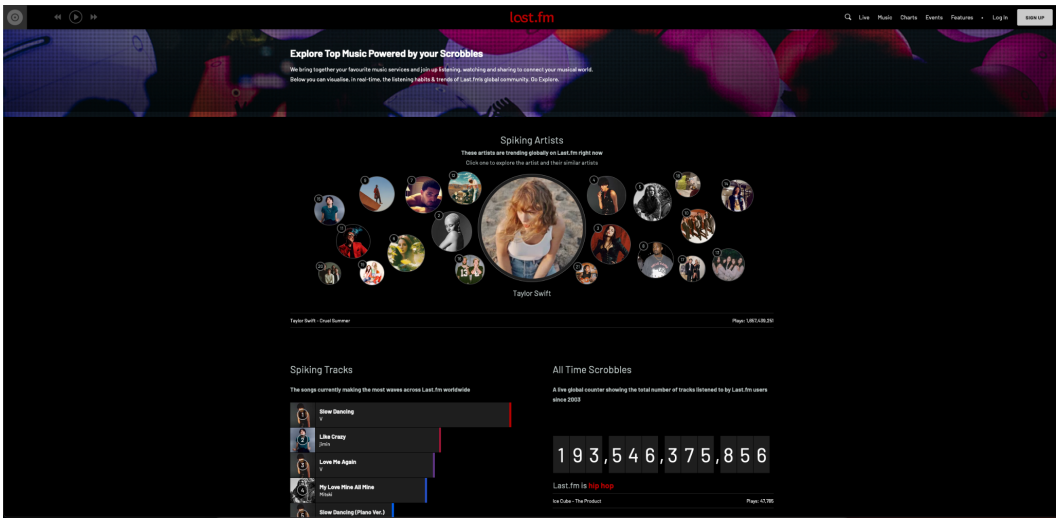

Рисунок 1.1 – Головна сторінка сервісу Last.fm

1.8.2. Pandora

Pandora представляє собою американський сервіс потокової трансляції музики, що функціонує на основі підписки і є власністю корпорації Sirius XM Holdings. Цей сервіс, розташований у місті Окленд, штат Каліфорнія, фокусується на наданні музичних рекомендацій, заснованих на унікальній методології "Проекту музичного геному", яка включає в себе класифікацію окремих треків за їхніми музичними характеристиками. Первісно представлений на ринку як послуга інтернет-радіо, цей сервіс був розроблений для створення персоналізованих музичних каналів, що базуються на цих характеристиках та виборі пісень користувачів, доступних у двох варіантах: з

рекламою та платною підпискою. В 2017 році Pandora представила Pandora Premium – версію на замовлення, яка конкурує з іншими лідерами ринку [12].

Заснована у 2000 році під назвою Savage Beast Technologies, компанія спочатку була створена як комерційне підприємство, що надавало ліцензії на використання проекту Music Genome для роздрібних продавців у якості платформи для рекомендацій. Однак у 2005 році компанія змінила свою стратегію, зосередившись на споживчому ринку, і запустила Pandora як інноваційний продукт інтернет-радіо. Пандора пропонує основні послуги безкоштовно, хоча вони можуть містити рекламу або мати певні обмеження, тоді як додаткові функції, такі як вища якість потокової трансляції, можливість завантаження музики та доступ до офлайн-каналів, доступні через платні підписки.

У своїй роботі Пандора використовує унікальну підмножину з 400 атрибутів від проекту Music Genome Project для кожного треку або виконавця, створюючи "станції", які відтворюють музику з схожими музичними характеристиками. Система адаптується до зворотного зв'язку користувачів, змінюючи вагу певних атрибутів в залежності від того, чи "подобається" або "не подобається" користувачеві конкретна пісня. Цей підхід є прикладом фільтрації на основі вмісту.

Головна сторінка сервісу Pandora показано на рисунку 1.2.

1.8.3. Uber та Lyft

Мобільні системи рекомендацій, як ті, що використовуються компаніями Uber [13] і Lyft [14] для створення маршрутів для водіїв таксі у містах, є хорошим прикладом використання передових технологій в транспортній галузі. Ці системи використовують дані GPS для збору інформації про маршрути, якими водії пересуваються під час роботи, включаючи такі дані, як місцезнаходження (координати широти та довготи), часові мітки та статус поїздки (з пасажирами або без них). На основі цієї інформації система здатна

надавати рекомендації щодо оптимальних точок посадки на маршруті, максимізуючи таким чином ефективність часу водіїв та їхній потенційний дохід.

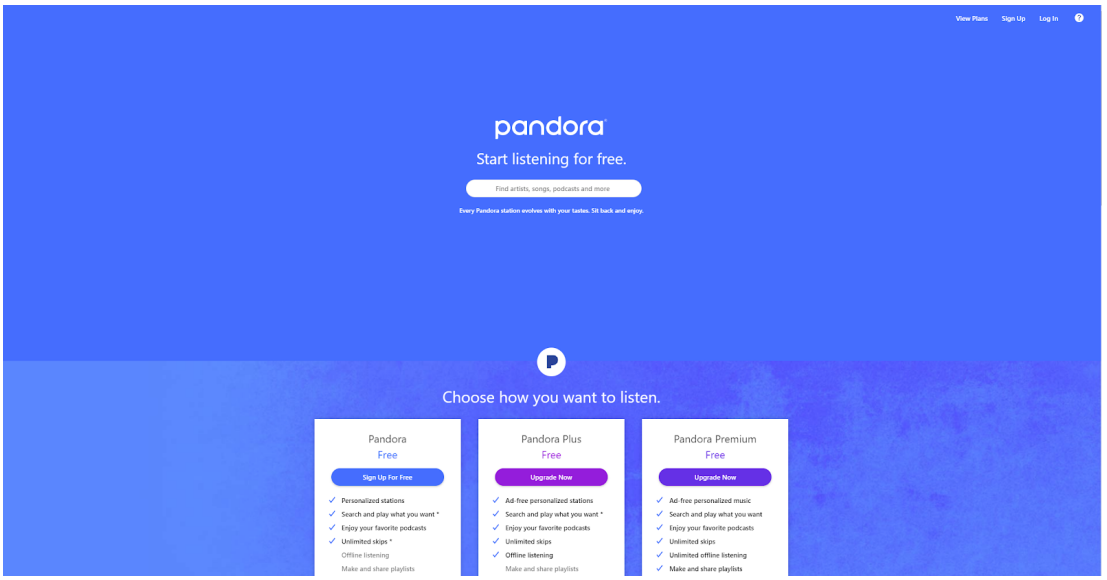

Рисунок 1.2 – Головна сторінка сервісу Pandora

Головна сторінка сервісу Uber та Lyft показані на рисунку 1.3 та рисунку 1.4 відповідно.

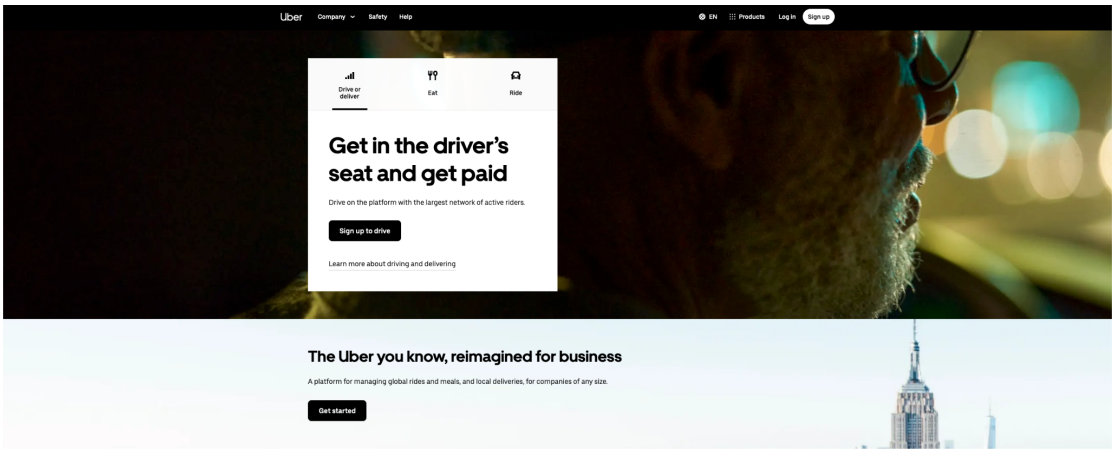

Рисунок 1.3 – Головна сторінка сервісу Uber

1.8.4. Netflix

Netflix є чудовим прикладом застосування гібридних рекомендаційних систем [15]. Їх веб-сайт надає рекомендації, аналізуючи звички перегляду користувача та порівнюючи їх із переглядовими звичками інших, схожих на них користувачів (це приклад колаборативної фільтрації). Крім того, Netflix рекомендує фільми, які мають подібні характеристики до тих, які користувач раніше високо оцінив, використовуючи фільтрацію на основі вмісту. Цей підхід дозволяє Netflix надавати персоналізовані рекомендації, які враховують індивідуальні уподобання та переваги користувачів.

Головна сторінка сервісу показана на рисунку 1.5.

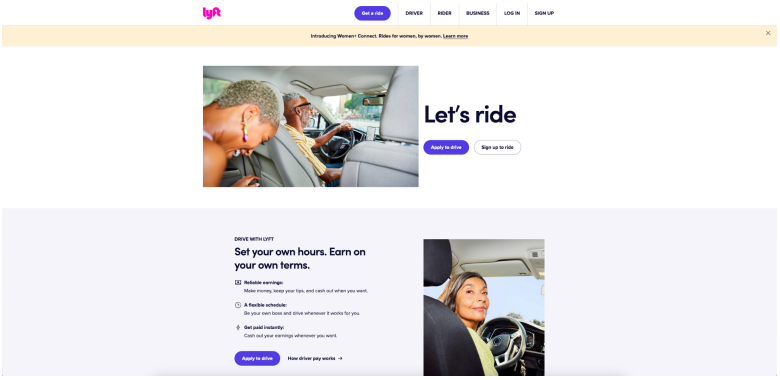

Рисунок 1.4 – Головна сторінка сервісу Lyft

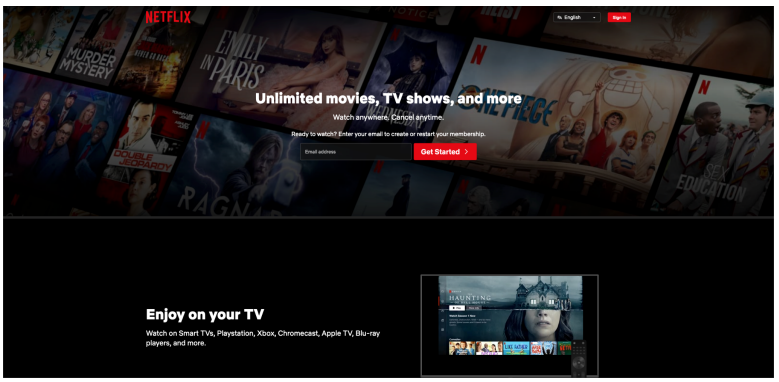

Рисунок 1.5 – Головна сторінка сервісу Netflix

Порівняння систем рекомендацій наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмних продуктів

Сервіс	*Тип рекомендації*	*Основні дані для рекомендацій*	*Використання TF-IDF*	*Інші методи/алгоритми*
Last.fm	Музика	Проходжені треки, вподобання	Так	Колаборативна фільтрація
Pandora	Музика	Аналіз атрибутів пісень	Ні	Music Genome Project
Uber/Lyft	Поїздки	Історія поїздок, геолокація	Ні	Прогностичні моделі
Netflix	Фільми/серіали	Історія переглядів, вподобання	Так	Глибоке машинне навчання

Рекомендаційні системи зарекомендували себе як потужний інструмент для підвищення користувацького досвіду в різних сферах, від розваг до транспортних послуг. Їх значущість і користь стає особливо очевидною, коли ми розглядаємо переваги, які вони можуть принести:

В епоху інформаційного перевантаження можливість отримувати персоналізовані рекомендації дозволяє користувачам швидше знаходити те, що вони шукають, забезпечуючи більш задоволені та лояльні користувачі.

Рекомендаційні системи зменшують кількість часу та зусиль, які користувачі повинні витратити на пошук ідей для дозвілля або відпочинку.

Користувачі можуть відкрити для себе нові рекреаційні зони, які вони можливо не розглядали б або не знали раніше.

Рекомендаційні системи можуть сприяти популяризації менш відомих, але якісних рекреаційних зон, сприяючи розвитку місцевого туризму та господарства.

Збір даних та відгуків про рекреаційні зони може допомогти у покращенні інфраструктури та сервісів в цих зонах.

Створення рекомендаційної системи для рекреаційних зон не лише поліпшить користувацький досвід, але й сприятиме розвитку та популяризації

цих зон. Це інвестиція в майбутнє, яка принесе користь як користувачам, так і господарям рекреаційних територій.

1.9. Постановка задачі для системи рекомендацій рекреаційних зон

Проаналізувавши питання розробки систем для рекомендацій рекреаційних зон, було визначено завдання, які необхідно виконати для розробки програмного додатку:

- визначити перелік критеріїв для формування рекомендацій;
- розробити алгоритму обміну повідомлень;
- розробити алгоритму оновлення датасету;
- розробка алгоритму навчання моделі;
- розробка метод прогнозування рекомендацій;
- розробка модуля прогнозування рекомендацій;
- розробка мобільного клієнта для роботи з модулем прогнозування рекомендацій;
- провести тестування програмного додатку.

1.10. Висновки

У першому розділі розглянуто стан питання рекомендаційних систем на сьогоднішній день.

Проаналізовано існуючі рекомендаційні системи та проведено їх порівняння. У результаті порівняння було доведено доцільність розробки власної рекомендаційної системи.

Проведено аналіз існуючих підходів до вирішення поставленої задачі. В результаті аналізу було обрано алгоритм фільтрації на основі вмісту з використанням подання TF-IDF (Term Frequency - Inverse Document Frequency) [16].

Сформульовано основні завдання, які необхідно виконати для розробки програмного додатку.

2. ПРОЕКТУВАННЯ СИСТЕМИ

2.1. Перелік критеріїв для формування рекомендацій

Перед тим як надавати конкретні рекомендації, необхідно спершу розробити критерії, на основі яких рекомендаційна система буде функціонувати. Ці критерії відіграють ключову роль у процесі фільтрації даних. Визначаючи ці критерії, користувач формує свої особисті інтереси, дозволяючи системі надавати найбільш відповідну йому інформацію.

Людська природа відзначається певною вимогливістю, що особливо проявляється при виборі місця відпочинку. У процесі визначення ідеальної рекреаційної зони, більшість людей враховують декілька ключових критеріїв.

Вибір рекреаційної зони базується на багатоманітних критеріях, які разом створюють багатоаспектний простір характеристик. Розглядаючи лише тип відпочинку, такий як активний або пасивний, ми розмежовуємо активності як "футбол" і "медитація" в різні категорії. Однак додавши параметр "відпочинок на відкритому повітрі", обидва варіанти відпочинку можуть опинитися в одному сегменті. Така динаміка вказує на відсутність жорстких меж у класифікації рекреаційних зон, зробивши критерії елементами, що перетинаються [17].

У таблиці 2.1 наведено приклад, коли множини критеріїв перетинаються.

Аналізуючи дані з Таблиці 2.1, стає зрозумілим, що критерії для вибору рекреаційних зон включають як активні, так і пасивні форми відпочинку. Це вказує на те, що різноманітні форми відпочинку можуть розглядатися в контексті їх суміжності. Таким чином, вибір місця для відпочинку на відкритому повітрі стає гнучкішим і може базуватися на комбінації критеріїв, які відповідають особистим уподобанням користувача.

Таким чином, перелік критеріїв для формування рекомендацій є симбіоз пасивних та активних критеріїв.

Таблиця 2.1 Перетин множин критеріїв фільтрації

Пасивний Активний	Бесідки	Медитація	Йога
Футбол	Відпочинок на свіжому повітрі	Відпочинок на пляжі	Риболовля
Велоспорт	Баскетбол	Відпочинок на свіжому повітрі	Настільні ігри
Прогулянки	Волейбол	Плавання	Відпочинок на свіжому повітрі

2.2. Структура системи

На рисунку 2.1 зображена архітектура рекомендаційної системи, призначеної для рекреаційних зон. Ця система складається з чотирьох основних модулів. Найважливішим серед них є модуль прийому та відправки повідомлень, який виконує функції комунікаційного вузла між сервером та користувачем. Цей модуль відповідає за прийом запитів від користувачів та надання їм відповідних рекомендацій.

Модуль оновлення датасету в рекомендаційній системі має критично важливу роль: він адаптує систему до нових даних, що надходять від користувачів. Це забезпечує постійну актуальність та релевантність рекомендацій, оскільки база даних системи регулярно оновлюється з новою інформацією. Після кожного оновлення бази даних активізується модуль навчання, який здійснює процес повторного навчання моделі, використовуючи оновлені дані. Після завершення навчання, оновлена модель передається модулю прогнозування, де на основі вхідних даних формуються та видаються нові рекомендації для користувачів. Цей процес забезпечує, що система постійно вдосконалюється та надає рекомендації, які відповідають поточним потребам та інтересам користувачів.

У рекомендаційній системі ключову роль відіграють модулі прогнозування та навчання, оскільки саме вони визначають ефективність та точність рекомендацій, які система надає. Модуль прогнозування відповідає за аналіз вхідних даних та формування відповідних рекомендацій на основі цих даних, використовуючи модель, розроблену та навчену модулем навчання. Модуль навчання, у свою чергу, займається адаптацією і вдосконаленням цієї моделі, використовуючи оновлені дані для забезпечення більш точних та актуальних прогнозів. Ці два модулі разом формують основу системи рекомендацій, забезпечуючи її здатність динамічно реагувати на змінні потреби та уподобання користувачів.

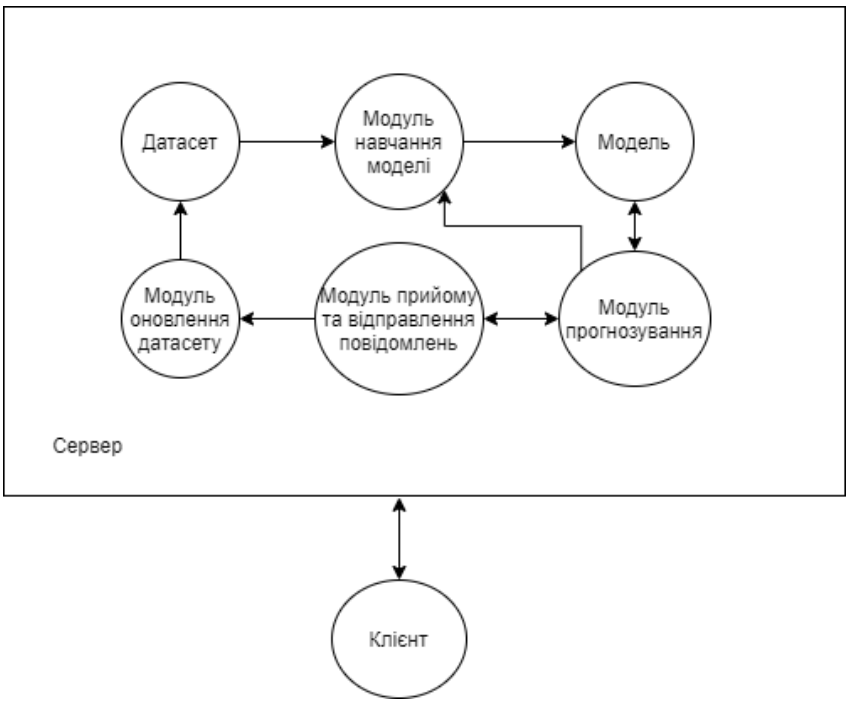

Рисунок 2.1 – Загальна структура системи

2.3. Алгоритм роботи модуля прийому та відправлення повідомлень

Алгоритм роботи модуля прийому та відправлення повідомлень є ключовим для взаємодії між серверною та клієнтською частинами рекомендаційної системи. Цей алгоритм визначає, як сервер реагує на

повідомлення, отримані від клієнтів, та як формуються відповіді на клієнтські запити. Він описує послідовність кроків, які виконує сервер для обробки запитів та надання відповідних даних користувачам. На блок-схемі, представленій на рисунку 2.2, детально зображено всі етапи цього алгоритму, від моменту отримання повідомлення від клієнта до відправлення відповіді назад до клієнта. Ця схема допомагає у візуалізації та розумінні процесу взаємодії між різними компонентами системи.

Робота алгоритму модуля прийому та відправлення повідомлень розпочинається з ініціалізації двох важливих компонентів: модуля прогнозування та модуля оновлення датасету. Це необхідно, оскільки модуль прийому та відправлення повідомлень використовує ці компоненти для надання важливої інформації користувачам. Наприклад, він може використовувати модуль прогнозування для створення рекомендацій на основі запиту користувача або ж звертатися до модуля оновлення датасету для отримання оновленої інформації, яка впливає на рекомендації. Ці ініціалізації забезпечують, що модуль прийому та відправлення повідомлень має доступ до актуальних даних та аналітичних інструментів, необхідних для ефективного обслуговування запитів користувачів.

Після завершення ініціалізації, алгоритм модуля прийому та відправлення повідомлень переходить до етапу прослуховування або очікування повідомлень від користувачів. На цьому етапі важливою є перша перевірка стану модуля прогнозування, щоб визначити, чи завершилося його виконання.

Якщо процес прогнозування завершений успішно, модуль формує повідомлення з інформацією про успішне виконання операції та відправляє його клієнту. Це може включати деталі про рекомендовані рекреаційні зони або інші відповідні дані згідно з запитом користувача.

У випадку, якщо процес формування рекомендацій не вдався або виникли якісь помилки, модуль також формує відповідне повідомлення, що інформує клієнта про проблему, і відправляє його. Після цього модуль продовжує

прослуховувати канал повідомлень, готовий до обробки нових запитів або взаємодії з іншими компонентами системи. Цей підхід дозволяє забезпечити безперервне функціонування системи та своєчасне реагування на запити користувачів.

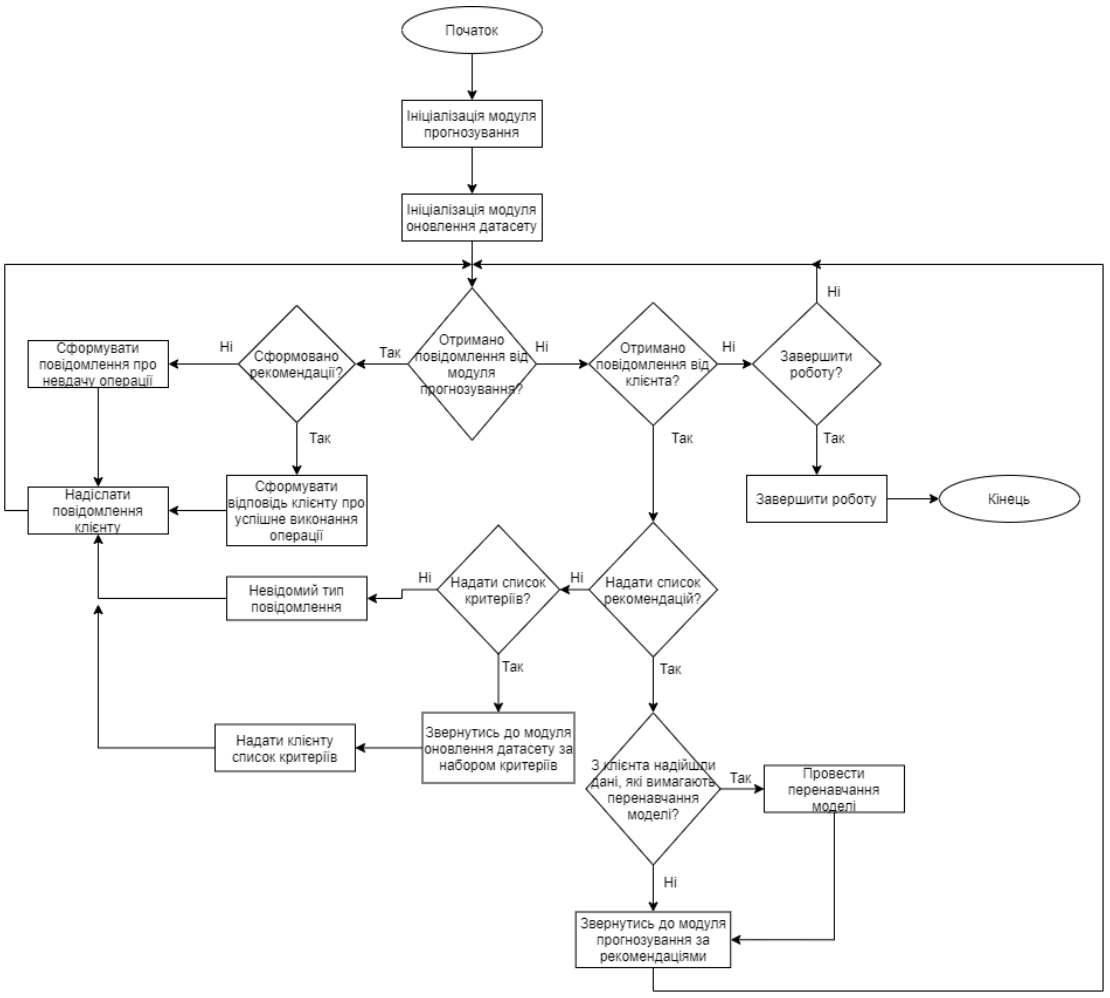

Рисунок 2.2 – Блок-схема алгоритму роботи модуля прийому та відправки повідомлень

Коли модуль прийому та відправлення повідомлень отримує повідомлення від клієнта, він починає з аналізу типу запиту клієнта. У разі, якщо клієнт бажає отримати список рекомендацій згідно з певними критеріями, модуль спочатку виконує перевірку, чи потрібне оновлення датасету перед тим, як приступити до формування рекомендацій. Це важливий крок для

забезпечення актуальності та точності рекомендацій, які надаються користувачеві.

Якщо оновлення датасету не потрібне або після його завершення, модуль звертається до модуля прогнозування для виконання процесу створення рекомендацій. Цей процес включає аналіз вхідних даних, використання навченої моделі для визначення потенційно цікавих рекомендацій і відправлення цих рекомендацій клієнту. У цей час модуль прийому та відправлення повідомлень продовжує слухати канал повідомлень для виявлення нових запитів від клієнтів або інших повідомлень, що можуть надійти. Ця послідовність дій забезпечує плавне та ефективне функціонування рекомендаційної системи, реагуючи на запити користувачів в оперативному режимі.

Якщо клієнт зробив запит на отримання списку критеріїв для рекомендацій, алгоритм модуля прийому та відправлення повідомлень виконує наступні дії. Перш за все, відбувається звернення до модуля оновлення датасету для отримання оновленого та актуального списку критеріїв. Це важливо для забезпечення того, що клієнт отримує найновішу інформацію, яка може бути використана для формування рекомендацій.

Після отримання цього списку, модуль формує відповідь, яка містить необхідні критерії, і відправляє її клієнту. Це гарантує, що користувач має усю необхідну інформацію для подальшого вибору або уточнення своїх запитів.

В іншому випадку, коли від клієнта надходить запит незрозумілого або непідтримуваного типу, модуль генерує повідомлення про те, що дана операція не може бути виконана або не підтримується системою. Таке повідомлення також відправляється клієнту, надаючи зрозуміле пояснення ситуації. Цей підхід забезпечує чітке комунікаційне взаємодія з користувачами та допомагає підтримувати ефективну роботу системи.

Коли до модуля прийому та відправлення повідомлень надходить команда на завершення роботи, алгоритм вміє розпізнавати таке повідомлення. У

відповідь на це, алгоритм ініціює процедуру завершення своєї роботи. Це включає закриття всіх активних процесів, збереження необхідних даних та налаштувань, а також забезпечення, що всі ресурси системи звільняються належним чином.

Такий механізм завершення роботи є важливим для забезпечення стабільності та безпеки системи, особливо в середовищі, де обробляється велика кількість даних або виконується багато паралельних завдань. Це також дозволяє системі ефективно перезапускатися або оновлюватися без втрати критично важливої інформації.

2.4. Алгоритм роботи модуля оновлення датасету

Алгоритм роботи модуля оновлення датасету, який було розроблено, описує дії системи у відповідь на зміни, такі як зміна кількості критеріїв для рекомендацій або зміна кількості рекреаційних зон. Цей алгоритм важливий для забезпечення актуальності даних, що використовуються системою для формування рекомендацій.

Відповідно до блок-схеми алгоритму, представленої на рисунку 2.3, система виконує певні кроки для адаптації до цих змін. Процес може включати перевірку оновлень у базі даних, аналіз змінених даних, переоцінку або оновлення існуючих критеріїв та внесення змін до списку рекреаційних зон.

Це забезпечує, що рекомендаційна система завжди використовує найновішу інформацію для надання релевантних та точних рекомендацій користувачам. Регулярне оновлення датасету є ключовим аспектом у підтримці ефективності та надійності системи рекомендацій.

Алгоритм роботи модуля оновлення датасету проходить кілька ключових етапів для ефективного управління даними. Робота алгоритму розпочинається з ініціалізації доступу до датасету, що включає налаштування необхідних з'єднань та ресурсів для роботи з даними.

Другим етапом є ініціалізація вхідних параметрів. На цьому етапі збираються та визначаються дані, які потрібно додати або оновити в датасеті.

Наступною стадією є підготовка даних для збереження. Це включає їх перевірку, очищення та приведення до відповідного формату або шаблону, який використовується в датасеті, щоб гарантувати їх сумісність та коректність.

Після підготовки даних відбувається їх збереження в датасеті. Це критичний крок, який оновлює базу даних з новою або зміненою інформацією.

Одразу після збереження даних модуль відправляє сповіщення про оновлення датасету. Це сповіщення може бути використане іншими модулями системи, щоб відреагувати на зміни в даних, наприклад, запустити процес перенавчання моделі або оновлення рекомендацій.

Останнім кроком є завершення роботи алгоритму, що включає деініціалізацію всіх ресурсів, які використовувались протягом процесу, і закриття роботи алгоритму. Це забезпечує коректне відключення від ресурсів та звільнення пам'яті, підготовуючи систему до наступних циклів роботи або завершення сесії.

2.5. Метод навчання моделі

Метод навчання моделі, який було розроблено, детально описує процес навчання моделі, яка використовується в рекомендаційній системі. Навчання моделі є фундаментальним кроком, який впливає на точність та релевантність рекомендацій, що надаються користувачам.

На блок-схемі, представленій на рисунку 2.4, відображено послідовність кроків, які входять до процесу навчання. Це може включати підготовку та очищення даних, вибір відповідного алгоритму машинного навчання, налаштування параметрів моделі, тренування моделі на основі існуючих даних, та оцінку її ефективності.

Рисунок 2.3 – Блок-схема алгоритму роботи модуля оновлення датасету

Цей алгоритм забезпечує, що модель вчиться із накопичених даних та адаптується до нових шаблонів або тенденцій, що можуть з'явитися. Це критично важливо для підтримки високої якості рекомендаційних систем, особливо у сферах, де уподобання користувачів та умови ринку постійно змінюються. Такий детальний і добре структурований підхід до навчання моделі дозволяє системі залишатися конкурентоспроможною та відповідати потребам користувачів.

Робота алгоритму навчання моделі в рекомендаційній системі для рекреаційних зон розпочинається з критичного кроку ініціалізації доступу до датасету. Це важливо для забезпечення, що алгоритм має доступ до потрібних даних для навчання моделі.

Рисунок 2.4 – Блок-схема алгоритму роботи методу навчання моделі

Після успішної ініціалізації відбувається зчитування вмісту датасету. Це включає збір усієї інформації, яка описує різні рекреаційні зони, для подальшої обробки.

Наступним етапом є обчислення частотності терміну (TF – Term Frequency). Цей процес полягає в аналізі того, наскільки часто певний термін зустрічається в описі кожної рекреаційної зони. Ця статистика допомагає

визначити ключові слова або фрази, які найбільше характеризують кожну зону, і може використовуватися для підвищення точності та релевантності рекомендацій, що генеруються системою.

Така інформація є критично важливою для процесу навчання моделі, оскільки вона дозволяє системі краще розуміти властивості та особливості кожної рекреаційної зони, а також виявити шаблони, які можуть бути використані для формування більш точних рекомендацій для користувачів.

Під час обчислення частотності терміну (TF – Term Frequency) в алгоритмі навчання моделі, важливо врахувати розмір документу, оскільки у великих документах певний термін може зустрічатися частіше порівняно з меншими документами. Це означає, що абсолютне число зустрічей терміна не завжди є показовим для визначення його важливості або релевантності в контексті документа.

Тому в алгоритмі використовуються відносні значення TF. Це робиться шляхом поділу кількості разів, коли необхідний термін було знайдено в документі, на загальну кількість слів у цьому документі. Такий підхід дозволяє нормалізувати частотність терміну, зробивши її порівняльною між документами різного розміру.

Це допомагає системі точніше оцінити значимість певних термінів в контексті кожної рекреаційної зони, незалежно від її опису та розміру. Таким чином, частотність терміну стає ефективним інструментом для аналізу та навчання моделі, що підвищує якість та точність рекомендацій системи. Тобто:

$$TF_a = \frac{N_a}{N}, \tag{2.1}$$

де N_a – кількість разів, коли термін a було знайдено у тексті, N – кількість всіх слів у тексті, TF_a – частотність терміну a.

Після обчислення частотності терміну (TF), наступним важливим кроком в алгоритмі навчання моделі є обчислення зворотної частотності документів (IDF – Inverse Document Frequency). IDF допомагає визначити важливість терміну в контексті всієї колекції документів.

Хоча обчислення TF вважає всі терміни однаково важливими, IDF диференціює їх важливість. Звичайні слова, які часто вживаються в багатьох документах, наприклад прийменники, можуть мати низьку інформативну цінність. Ці слова, які часто зустрічаються, але не вносять значного внеску у суть тексту, отримують нижчу вагу в розрахунку IDF.

IDF кожного терміну обчислюється як логарифм відношення загальної кількості документів у колекції до кількості документів, які містять цей термін. Високий показник IDF вказує на те, що термін з'являється в меншій кількості документів, тобто має вищу рідкісність і потенційно більшу важливість.

Застосування IDF дозволяє збалансувати вагу термінів, надаючи більшої ваги тим словам, які унікальні для певних документів, і зменшуючи вагу загальновживаних слів. Це підвищує якість аналізу та ефективність моделі, що використовується у рекомендаційній системі. Тобто:

$$IDF_a = log(\frac{N}{N_a}) \ , \tag{2.2}$$

де N – загальна кількість документів, N_a – кількість документів в яких зустрічається термін a, IDF_a – зворотна частотність документів для терміна a.

По завершенню обчислення IDF починається процес обчислення величини TF-IDF, тобто:

$$TF - IDF_a = TF_a \times IDF_a, \quad (2.3)$$

де TF_a – частотність терміну a, IDF_a – зворотна частотність документів для терміна a [18].

В алгоритмі навчання моделі для рекомендаційної системи після обчислення частотності терміну (TF) та зворотної частотності документів (IDF) відбувається їх інтеграція для отримання значення TF-IDF кожного терміну в корпусі текстів. Це виконується наступним чином:

1. Для кожного тексту обчислюється TF кожного слова, що зустрічається в тексті. Це означає визначення частоти, з якою кожен термін з'являється у тексті, відносно загальної кількості слів у тексті.
2. Для кожного слова в кожному тексті також обчислюється IDF. Це вимірює, наскільки унікальним є слово в контексті всього корпусу текстів.
3. Кожний TF множиться на відповідний IDF для кожного слова. Таким чином, слова, що часто зустрічаються в одному тексті, але рідко в інших, отримують вищу вагу TF-IDF.
4. Кожний розрахунок TF-IDF для слова в тексті зберігається у словник, який потім додається до списку. Цей список відображає порядок текстів, які були на вході, забезпечуючи збереження структури даних.
5. На завершення, алгоритм повертає цей список, що містить розраховані значення TF-IDF для кожного терміну в кожному тексті.

Отримані таким чином значення TF-IDF допомагають визначити релевантність та важливість слова в контексті окремого тексту та всього корпусу, що є ключовим для точності та ефективності моделі, яка використовується в рекомендаційній системі.

Після обчислення TF-IDF, починається обчислення косинусної схожості. Щоб рекомендувати рекреаційні зони, які найбільш підходять за критеріями, які вказав користувач, ми обчислюємо косинусну схожість між рекреаційними

зонами. Найбільш схожі будуть рекомендовані. Якщо дано два вектори ознак, A та B, то косинусна схожість, cos(θ), може бути представлена, використовуючи скалярний добуток та норму, тобто:

$$cos(\Theta) = \frac{A \cdot B}{||A||||B||}, \tag{2.4}$$

де A та B вектори ознак, $cos(\Theta)$ – косинусна схожість [18].

Після завершення процесу обчислення TF-IDF для кожного терміну в корпусі текстів, алгоритм роботи модуля навчання моделі досягає своєї кульмінаційної фази – формування моделі. Ця модель, обладнана вагами TF-IDF для різних термінів, стає інструментом для надання рекомендацій. Вона використовує отримані ваги для визначення ступеня релевантності різних рекреаційних зон або інших об'єктів, які аналізуються системою, відносно інтересів та переваг користувача.

Останній етап алгоритму – це завершення його роботи. На цій стадії відбувається деініціалізація всіх використаних ресурсів. Це може включати закриття доступу до датасету, звільнення пам'яті, яка використовувалася для обчислень, та закриття будь-яких відкритих файлів або з'єднань. Цей крок забезпечує, що алгоритм завершує свою роботу коректно та ефективно, не залишаючи відкритих ресурсів або незавершених процесів, що може вплинути на продуктивність системи.

Таким чином, завершений метод навчання моделі не тільки готує модель для надання рекомендацій, але й гарантує, що весь процес відбувається з оптимальним використанням системних ресурсів.

2.6. Метод прогнозування рекомендацій

Метод прогнозування рекомендацій відіграє ключову роль у процесі формування рекомендацій у рекомендаційній системі. Цей алгоритм включає кроки, які дозволяють системі аналізувати вхідні дані, використовувати модель, яка була навчена раніше, та генерувати релевантні рекомендації для користувачів на основі цього аналізу. Блок-схема, представлена на рисунку 2.5, детально ілюструє послідовність дій, які виконує алгоритм при формуванні рекомендацій.

Рисунок 2.5 – Блок-схема алгоритму роботи методу прогнозування рекомендацій

Перший крок алгоритму полягає в ініціалізації модуля навчання моделі, що може включати завантаження необхідних бібліотек, конфігураційних файлів, а також запуск початкових процедур для підготовки до процесу навчання.

Для того щоб рекомендації були корисними та відповідними, необхідно визначити критерії, за якими буде проводитись аналіз та формування рекомендацій.

Модель, яка буде використовуватися для видачі рекомендацій, ініціалізується шляхом завантаження вагів, параметрів та структури, яка була підготовлена раніше.

Перш ніж приступити до формування рекомендацій, система перевіряє, чи актуальна модель. Якщо вона була навчена на застарілих даних, то відбувається її перенавчання з використанням актуальних даних.

На основі ініціалізованих критеріїв та перенавченої моделі алгоритм формує рекомендації для користувача.

Після того, як рекомендації були сформовані та передані відповідним модулям (наприклад, модулю прийому та відправки повідомлень), робота алгоритму завершується. При цьому відбувається деініціалізація ресурсів, що означає очищення пам'яті, закриття файлів тощо.

Даний метод пропонує має наступні особливості:

- Враховує особистісні критерії від користувача, що забезпечує високий рівень персоналізації рекомендацій. Такий підхід може підвищити задоволення користувача, оскільки пропозиції будуть максимально відповідати його індивідуальним уподобанням;
- Можливість перенавчання моделі дозволяє системі адаптуватися до змінних уподобань користувачів та актуальних даних про рекреаційні зони, забезпечуючи актуальність рекомендацій;

- Система здатна працювати з різними джерелами та обсягами даних, а також здатна інтегруватися з іншими зовнішніми модулями, що робить її гнучкою до розширення та інтеграції.

Використання рекомендаційних систем, які базуються на особистісних уподобаннях користувачів, має важливе значення для забезпечення високого рівня користувацького задоволення та лояльності. Коли користувачі бачать, що система враховує їхні унікальні переваги та інтереси, вони відчувають, що система «розуміє» їхні потреби, що підвищує їхнє задоволення та залученість.

Постійне оновлення моделі на основі нових даних є ключовим аспектом у підвищенні точності рекомендацій. Це допомагає зменшити кількість невдалих або неактуальних пропозицій, оскільки модель адаптується до змінних уподобань користувачів та тенденцій ринку.

Автоматизація процесу рекомендацій також призводить до значного зниження потреби в ручній роботі, економлячи час та ресурси. Це особливо важливо для масштабованих систем, де обробка великої кількості даних та запитів вручну була б нереалістичною.

Крім того, такі системи можуть бути легко адаптовані до змінних потреб ринку та користувачів. Вони здатні масштабуватися з ростом кількості користувачів та обсягу даних, що робить їх ефективними для використання у різноманітних секторах та за різних умовах ринку.

Таким чином, використання персоналізованих рекомендаційних систем може стати значним конкурентною перевагою, підвищуючи задоволеність та лояльність користувачів, а також оптимізуючи внутрішні ресурси та процеси.

2.7. Висновки

У другому розділі було розроблено алгоритмічне забезпечення системи. Було розроблено наступні алгоритми: роботи модуля прийому та відправки

повідомлень, роботи модуля оновлення датасету, а також методи прогнозування рекомендацій.

Метод прогнозування рекомендації рекреаційних зон, що базується на індивідуальних критеріях користувачів, відіграє ключову роль у створенні більш персоналізованого та задовільного досвіду. Його важливість полягає в здатності адаптуватися до постійно змінних уподобань і трендів, підтримуючи актуальність і релевантність рекомендацій. Основними перевагами є ефективність вибору для користувачів, підвищення їхньої взаємодії та залучення, а також оптимізація маркетингових зусиль. Особливістю методу є його гнучкість і масштабованість, а також можливість перенавчання для забезпечення неперервної актуалізації рекомендацій. Цей метод може надати вирішальну конкурентну перевагу рекреаційним сервісам, підвищуючи задоволення користувачів та їх відданість.

Метод тренування моделі забезпечує підготовку моделі до створення рекомендацій, одночасно забезпечуючи ефективне використання системних ресурсів протягом усього процесу.

Таким чином, було розроблено алгоритмічну базу системи.

3. РОЗРОБКА СИСТЕМИ

3.1. Розробка серверної частини

Розробка серверної частини системи була виконана з використанням мови програмування Python і веб-фреймворку Flask. Flask є популярним мікрофреймворком для створення веб-додатків, відомим своєю гнучкістю та легкістю у використанні. Основними перевагами Flask є його легка структура та здатність легко інтегруватися з різними розширеннями [19], [20].

Використання Flask дозволяє розширювати функціонал серверної частини системи за допомогою великої кількості доступних розширень. Ці розширення можуть включати додаткові інструменти для роботи з базами даних, формування веб-форм, автентифікації користувачів, управління сесіями та багато іншого [21]. Вони інтегруються таким чином, що здається, ніби ці функції є частиною Flask з самого початку, що робить роботу розробників більш ефективною та зручною [22].

Завдяки своїй простоті та гнучкості, Flask є ідеальним вибором для розробки серверної частини рекомендаційної системи, особливо в умовах, де необхідна швидка розробка та прототипування. Python, у свою чергу, забезпечує потужність та гнучкість необхідні для обробки великих обсягів даних та реалізації складних алгоритмів, що є критично важливими для рекомендаційних систем [23].

3.1.1. Розробка модуля прийому та відправлення повідомлень

Розробка модуля для прийому та відправлення повідомлень є ключовою частиною створення ефективної рекомендаційної системи. Цей модуль втілює в собі алгоритм, описаний у розділі 2.3, і забезпечує важливу функціональність: взаємодію з клієнтами через передачу та прийом повідомлень.

Блок-схема роботи модуля, представлена на рисунку 3.1, детально ілюструє, як модуль взаємодіє з клієнтами. Це допомагає зрозуміти потік даних та логіку обробки запитів.

Код модуля, наведений у додатку В, надає технічні деталі реалізації цього модуля, включаючи структуру коду, використані бібліотеки, методи обробки даних та інтерфейси для комунікації з іншими компонентами системи.

Даний модуль включає в себе два ключові класи: CriteriaAPI та RecommendationAPI, кожен з яких виконує специфічні функції у контексті рекомендаційної системи.

Клас CriteriaAPI займається обробкою GET-запитів від клієнтів, які хочуть отримати список критеріїв. Він має один основний метод getCriterias. Коли клієнт виконує GET-запит, цей метод активізується. Він звертається до бази даних, щоб отримати потрібний список критеріїв. Після отримання даних з бази, метод формує відповідь, яка включає необхідну інформацію про критерії, і відправляє цю відповідь назад клієнту. Це забезпечує ефективну комунікацію між сервером і клієнтом, дозволяючи користувачам отримувати актуальні дані про доступні критерії.

Клас RecommendationAPI у рекомендаційній системі відіграє ключову роль у обробці запитів користувачів, що стосуються отримання рекомендацій. Два методи цього класу, getRecommendation та getRecommendations, забезпечують гнучкість у взаємодії з користувачами.

Метод getRecommendation використовується для обробки GET-запитів від клієнтів, які хочуть отримати рекомендації за одним конкретним критерієм, ідентифікованим за його ідентифікатором. Це може бути корисним для користувачів, які мають специфічний інтерес або потребу і хочуть отримати рекомендації, які відповідають лише цьому конкретному аспекту.

Метод getRecommendations на відміну від getRecommendation, цей метод обробляє POST-запити, де клієнти можуть передати список ідентифікаторів критеріїв, за якими вони хочуть отримати рекомендації. Це дозволяє

користувачам отримувати більш комплексні та різноманітні рекомендації, базуючись на багатьох критеріях одночасно.

Застосування цих двох методів у RecommendationAPI забезпечує ефективне та гнучке вирішення різних сценаріїв запитів користувачів. Можливість обирати між отриманням рекомендацій за одним критерієм або кількома забезпечує, що система може задовольнити різноманітні потреби та переваги користувачів, підвищуючи користувацьке задоволення та ефективність системи.

Підхід до обробки запитів у методах getRecommendation і getRecommendations класу RecommendationAPI включає ефективну взаємодію між різними компонентами системи для генерації рекомендацій.

Коли один із цих методів отримує ідентифікатор критерія, він спочатку виконує запит до бази даних для отримання повної інформації про цей критерій. Це може включати деталі як-от опис критерію та його характеристики.

Після отримання критерія з бази даних, він передається до модуля прогнозування рекомендацій. В цьому модулі використовується попередньо навчена модель для визначення рекомендацій, які відповідають заданому критерію. Модель аналізує критерій, порівнювати його з іншими даними в системі і виходячи з цього генерувати список рекомендованих опцій.

Модуль прогнозування повертає список готових рекомендацій, які потім відправляються назад клієнту через getRecommendation або getRecommendations. Ці рекомендації засновані на обраному критерії та адаптовані до потреб та інтересів користувача.

Підхід до обробки запитів на рекомендації в методах getRecommendation і getRecommendations класу RecommendationAPI розрізняється залежно від кількості критеріїв, на основі яких формуються рекомендації.

```
from Database import database_service
from config import app
from flask import request
from flask import redirect
from flask import jsonify
from flask import make_response

class CriteriaAPI:
    @app.route("/criterias", methods = ['GET'])
    def getCriterias():
        criterias = database_service.DBService.criterias()
        return make_response(jsonify({ "data": criterias }), 200)

class RecommendationAPI:
    @app.route("/recommendation/<id>", methods = ['GET'])
    def getRecommendation(id):
        criteria = database_service.DBService.criteriaBy(id)
        recommendations = recommendation.Recommendation.getRecommendations(criteria.title)
        return make_response(jsonify({ "data": list(recommendations) }), 200)

    @app.route("/recommendations", methods = ['POST'])
    def getRecommendations():
        data = request.get_json(force=True)
        result = list()

        for id in data['id']:
            criteria = database_service.DBService.criteriaBy(id)
            recommendations = recommendation.Recommendation.getRecommendations(criteria.title)
            result.append(list(recommendations))

        return make_response(jsonify({ "data": list(result) }), 200)
```

Рисунок 3.1 – Код модуля прийому та відправлення повідомлень

Метод getRecommendation зосереджений на обробці запиту, заснованого на одному критерії. Коли метод отримує ідентифікатор критерія, він виконує запит до бази даних для отримання інформації про цей критерій, передає його до модуля прогнозування для генерації відповідних рекомендацій, а потім формує та відправляє ці рекомендації назад клієнту.

Метод getRecommendations, на відміну від getRecommendation, працює з декількома критеріями одночасно. Він отримує список ідентифікаторів критеріїв у POST-запиті і виконує аналогічні кроки (запит до бази даних, передача даних у модуль прогнозування, формування рекомендацій), але робить це ітеративно для кожного критерія у списку. Кінцевий результат — це набір рекомендацій, що відображають усі зазначені критерії.

Така структура дозволяє користувачам бути більш гнучкими у виборі, як отримувати рекомендації — чи то зосередитися на одному конкретному аспекті,

чи розглядати ширший діапазон інтересів [24]. Це забезпечує кращу адаптацію системи до різних сценаріїв використання та підвищує задоволеність користувачів завдяки більш точним та персоналізованим рекомендаціям.

3.1.2. Розробка модуля оновлення датасету

Розробка модуля оновлення датасету є важливою частиною рекомендаційної системи, оскільки вона забезпечує актуальність та релевантність даних, які використовуються для генерації рекомендацій. Цей модуль відповідає за регулярне оновлення інформації в датасеті, що є критично важливим для підтримки точності та ефективності системи.

Модуль розроблений згідно з алгоритмом, описаним у розділі 2.4. Це означає, що він включає всі необхідні процедури для перевірки, оновлення та збереження даних.

Основна функція модуля полягає в тому, щоб регулярно оновлювати датасет, враховуючи нову інформацію або зміни в існуючих даних. Це може включати оновлення даних про рекреаційні зони, користувацькі переваги, зміни в рейтингах або відгуках та інше.

Модуль спроектований для автоматичного виконання оновлень, зменшуючи потребу в ручному втручанні та підвищуючи ефективність процесів.

Код модуля, який наведено у додатку Г, надає технічні деталі його реалізації, дозволяючи зрозуміти його структуру та логіку роботи.

Візуалізація роботи модуля на рисунку 3.2 допомагає зрозуміти його місце та роль у загальній архітектурі системи. Це дозволяє розглянути взаємодію

модуля з іншими компонентами системи та його вплив на загальний процес генерації рекомендацій.

```
import os
import pandas as pd

class Store:
    @staticmethod
    def addOrUpdate(entity):
        currentDir = os.path.dirname(os.path.abspath(__file__))
        currentDir = os.path.join(currentDir, "dataset.csv")
        pd.write(currentDir, entity)

    @staticmethod
    def remove(id):
        currentDir = os.path.dirname(os.path.abspath(__file__))
        currentDir = os.path.join(currentDir, "dataset.csv")
        pd.removeBy(id, currentDir)
```

Рисунок 3.2 – Код модуля оновлення датасету

реалізація класу Store в модулі для роботи з датасетом, який є CSV-файлом, ефективно використовує можливості бібліотеки pandas для здійснення ключових операцій з даними.

Метод addOrUpdate виконує ініціалізацію шляху до датасету, що гарантує, що метод знає місцезнаходження файлу датасету, з яким він буде працювати. За допомогою pandas метод читає вміст CSV-файлу, виконує необхідні дії для додавання нового запису або оновлення існуючого. Це може включати пошук відповідного рядка за унікальним ідентифікатором та зміну його вмісту або додавання нового рядка.

Метод remove виконує ініціалізацію шляху до датасету, аналогічно методу addOrUpdate, це забезпечує доступ до датасету. Цей метод використовує pandas для пошуку та видалення відповідного запису в CSV-файлі. Це може включати фільтрацію датасету для знаходження відповідного рядка і його видалення.

Використання pandas для роботи з CSV-файлами є ефективним рішенням, оскільки ця бібліотека надає широкий спектр функціональностей для аналізу та маніпулювання даними у вигляді табличних структур [25]. Це робить процес роботи з датасетами більш зручним і зменшує ризик помилок, які можуть виникнути при ручному управлінні даними. Цей підхід забезпечує гнучкість та ефективність управління даними в рекомендаційній системі.

3.1.3. Розробка методу навчання моделі

Розробка методу навчання моделі є ключовим компонентом рекомендаційної системи, оскільки він безпосередньо впливає на якість та точність рекомендацій, які система здатна надавати.

Модуль навчання моделі виконує алгоритм, описаний у розділі 2.5, що включає кроки для обробки даних, визначення важливих характеристик, навчання моделі на основі цих даних та її оцінки.

Як один з головних модулів системи, він відіграє центральну роль у процесі рекомендації, адже якість навченої моделі безпосередньо впливає на релевантність та корисність рекомендацій для кінцевих користувачів.

В рамках цього модуля проводяться дослідження, спрямовані на вдосконалення процесу навчання моделі та підвищення ефективності рекомендацій. Результати цих досліджень можуть включати аналіз різних

підходів до навчання, оцінку точності моделі та її здатності адаптуватися до змінних умов.

Технічні деталі та реалізація модуля навчання моделі доступні у додатку Д, де можна ознайомитися зі структурою коду, використаними алгоритмами, бібліотеками та методами обробки даних.

Частина коду модуля наведено на рисунку 3.3 допомагає зрозуміти його роль та взаємодію з іншими частинами системи, підкреслюючи важливість точного та ефективного процесу навчання для успішної роботи рекомендаційної системи.

```
import os
import pandas as pd

class ModelLearning:

    @staticmethod
    def learn():
        currentDir = os.path.dirname(os.path.abspath(__file__))
        currentDir = os.path.join(currentDir, "dataset.csv")
        ds = pd.read_csv(currentDir)

        tf = TfidfVectorizer(analyzer='word', ngram_range=(1, 3), min_df=0, stop_words=get_stop_words('ukrainian'))
        tfidf_matrix = tf.fit_transform(ds['description'])
        cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)
        mapping = pd.Series(ds.index, index = ds['criteria'])
        return (mapping, cosine_similarities)
```

Рисунок 3.3 – Код модуля навчання моделі

Підхід до навчання моделі у класі ModelLearning модуля використовує ряд важливих кроків обробки даних та алгоритмічних технік.

Ініціалізація шляху до датасету - це перший крок, який гарантує, що програма знає, де знаходиться файл датасету для подальшої обробки.

Зчитування даних з датасету, використовуючи засоби для роботи з даними (бібліотеку pandas), датасет зчитується для подальшого аналізу.

Обчислення Term Frequency (TF), використовуючи бібліотеку sklearn, проводиться аналіз частотності термінів у датасеті. Це дозволяє оцінити, наскільки часто певні терміни зустрічаються в документах.

Після обчислення TF виконується розрахунок TF-IDF (Term Frequency-Inverse Document Frequency), який допомагає оцінити важливість термінів у контексті всього корпусу.

Обчислення косинусної схожості - цей крок включає визначення схожості між різними документами або термінами на основі їх векторного представлення. Косинусна схожість дозволяє оцінити, наскільки близько або віддалено розташовані терміни відносно один одного у векторному просторі, що може використовуватися для визначення, до якого критерія краще відноситься кожен термін.

Після завершення всіх обчислень, метод learn повертає результати, які можуть використовуватися для подальшого аналізу або як вхідні дані для модуля прогнозування рекомендацій [26].

Використання таких технік як TF-IDF та косинусна схожість дозволяє глибше зрозуміти зміст даних та покращити якість наданих рекомендацій.

3.1.4. Розробка методу прогнозування рекомендацій

Розробка методу прогнозування рекомендацій є важливим кроком у створенні ефективної рекомендаційної системи. Цей модуль відіграє центральну роль у визначенні того, які рекомендації будуть надані користувачам, засновані на їхніх перевагах та історії взаємодій.

Модуль прогнозування рекомендацій використовує алгоритм, описаний у розділі 2.6, який може включати обробку даних, алгоритми машинного навчання, аналіз шаблонів та поведінки користувачів.

Інтеграція цього модуля з іншими компонентами системи, описаними у розділі 3.1.3, є ключовою для забезпечення безперервного потоку даних і функцій між різними частинами системи.

Проведені в рамках цього модуля дослідження допомагають виявити найбільш ефективні методи прогнозування та підходи до формування рекомендацій, що сприяє підвищенню точності та релевантності рекомендацій.

Код модуля, наведений у додатку Е, дозволяє глибше зануритися в технічні деталі його реалізації, включаючи логіку обробки даних, використані алгоритми та методики програмування.

Частина коду наведено на рисунку 3.4, що надає візуальне уявлення про його роль та функціонування в контексті загальної архітектури системи.

```
class Recommendation:

    @staticvar
    result = ModelLearning.learn()

    @staticmethod
    def getRecommendations(criteria):
        currentDir = os.path.dirname(os.path.abspath(__file__))
        currentDir = os.path.join(currentDir, "dataset.csv")
        ds = pd.read_csv(currentDir)

        placeIndex = result.mapping[criteria]

        for index in placeIndex:
            similarityScore = list(enumerate(result.cosine_similarities[index]))
            similarityScore = sorted(similarityScore, key=lambda x: x[1], reverse=True)

            if similarityScore.value >= 0.75:
                places += [similarityScore.id]

        return ds['description'].iloc[places]
```

Рисунок 3.4 – Код модуля прогнозування рекомендацій

Розробка модуля прогнозування рекомендацій з класом Recommendation і методом getRecommendations є важливою частиною вашої рекомендаційної системи.

Ініціалізація шляху до датасету - це перший крок, який забезпечує доступ до необхідних даних, які будуть використовуватися для формування рекомендацій.

Використання бібліотеки pandas для зчитування та обробки датасету є ефективним способом роботи з даними, оскільки ця бібліотека надає широкі можливості для аналізу та маніпуляції табличними даними.

Використання статичної змінної класу - цей крок може включати звернення до попередньо визначених даних або конфігурацій, які важливі для роботи алгоритму.

Ініціалізація локальної змінної placeIndex - ця змінна може використовуватися для ідентифікації або відстеження певних місць чи рекреаційних зон у процесі формування рекомендацій.

Метод проходить через всі оцінки, сортує їх, і перевіряє, чи значення score перевищує порогове значення 0.75. Якщо так, то відповідне місце занотовується як потенційний кандидат для рекомендації.

Після завершення циклу, на основі відібраних місць формується і повертається список, який містить описи рекреаційних зон. Цей список може використовуватися для надання кінцевих рекомендацій користувачам.

Такий підхід забезпечує, що рекомендації, які надаються користувачам, базуються на аналітичній обробці та відборі даних, підвищуючи релевантність та персоналізацію рекомендацій. Використання порогового значення score для

відбору рекомендацій також допомагає забезпечити високу якість та точність пропонованих варіантів.

3.1.5. Хостинг серверу

Розміщення серверної частини рекомендаційної системи на хмарній платформі Heroku є ефективним рішенням, що надає ряд переваг.

Використання безкоштовного тарифного плану на Heroku дозволяє розмістити сервер без додаткових витрат. Це особливо корисно для тестування, розвитку проектів або для малих систем, які не потребують значних обчислювальних ресурсів.

Heroku підтримує широкий спектр мов програмування, включаючи Python, який ви використовували для розробки вашої системи. Це гарантує гнучкість та зручність у виборі технологій для реалізації проекту.

Використання хмарної платформи забезпечує високу доступність та надійність сервера, оскільки хмарні рішення зазвичай мають високий рівень відмовостійкості та підтримки.

Heroku дозволяє легко масштабувати ресурси сервера відповідно до зростаючих потреб системи, що особливо важливо для систем, які можуть зазнавати підвищеного навантаження або росту кількості користувачів.

Наявність сторінки сервера на Heroku, як показано на рисунку 3.5, забезпечує зручний інтерфейс для моніторингу стану та управління сервером.

Завдяки хостингу на Heroku, ваш сервер стає доступним у глобальній мережі Інтернет. Це означає, що користувачі з будь-якої точки світу можуть взаємодіяти з вашою системою, відправляючи запити та отримуючи відповіді.

Таким чином, вибір Heroku як платформи для рекомендаційної системи підкреслює намір створити легко доступну, гнучку та ефективну систему, здатну взаємодіяти з широким спектром користувачів та інших систем.

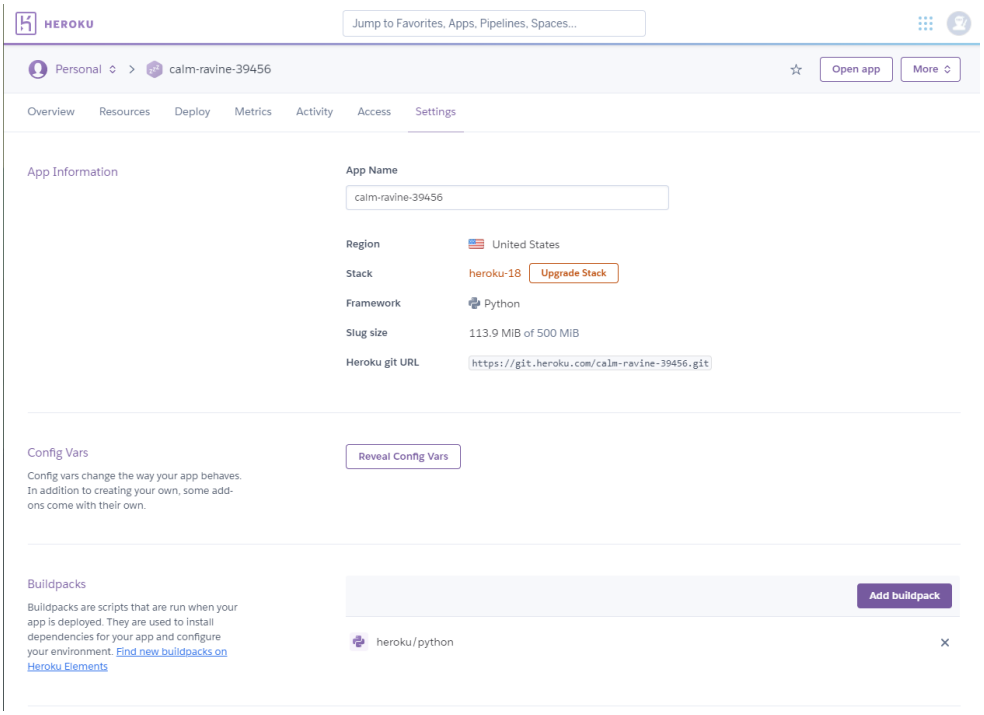

Рисунок 3.5 – Сторінка серверу на платформі Heroku

3.2. Розробка клієнтської частини

Розробка клієнтської частини системи для iOS з використанням Swift та Xcode є чудовим вибором, що відображає сучасні тенденції у розробці мобільних додатків.

Клієнтська частина на iOS буде служити як важливий інтерфейс для кінцевих користувачів, дозволяючи їм взаємодіяти з системою та отримувати персоналізовані рекомендації. Це забезпечує більш глибоке залучення користувачів і покращує загальний досвід використання вашої системи.

3.2.1. Відправка запиту до серверу для отримання списку критеріїв

Реалізація функціоналу відправки запитів з мобільного додатку на iOS до сервера для отримання списку критеріїв є важливим компонентом, що забезпечує зв'язок між користувачем та системою. Це дозволяє користувачам отримувати актуальну інформацію про критерії, які можуть бути використані для уточнення рекомендацій. Частину коду наведено на рисунку 3.6.

Повний код реалізації наведено в додатку Ж дозволяє детально ознайомитися з технічними аспектами роботи цієї частини системи, що може бути корисним для розуміння загальної архітектури та взаємодії між клієнтом та сервером.

Імплементація сутності "Критерій" на клієнті та сервері спрощує обмін та синхронізацію даних між цими частинами системи. Це гарантує, що обидві сторони системи "розуміють" одне одного, і значно зменшує ймовірність помилок при обробці даних. Частину коду наведено на рисунку 3.7.

Єдиний формат сутності "Критерій" на клієнті та сервері допомагає уникнути помилок, пов'язаних з вказівкою неправильних параметрів при виконанні запитів. Це забезпечує більш плавну та надійну роботу системи в цілому.

Використання такого структурованого та узгодженого підходу в розробці та взаємодії між клієнтською та серверною частинами системи є ключовим фактором для створення ефективного, користувацьки зручного та надійного програмного продукту.

```
final class RecommendationService: RecommendationUseCase {
    let network: Network

    init(network: Network) {
        self.network = network
    }

    func criterias() -> AsyncTask<[Criteria]> {
        network.reactive
        .request(API.Criterias.getCriterias)
        .decode(APIResponse<[Criteria.Response]>.self)
        .map { $0.data.map { Criteria(from: $0) }}
        .observe(on: UIScheduler())
    }
        // ...
}
```

Рисунок 3.6 – Код запиту для отримання списку критеріїв

```
struct Criteria {
    let id: Int
    let title: String
    let description: String
}

extension Criteria {
    struct Response: Decodable {
        let id: Int
        let title: String
        let description: String
    }

    init(from response: Response) {
        self.init(id: response.id, title: response.title, description: response.description)
    }
}
```

Рисунок 3.7 – Модель Критерій на клієнтській частині системи

Метод criterias в класі RecommendationService, що наведено на рисунку 3.6, виконує асинхронний запит до серверу для отримання списку критеріїв. Асинхронність важлива для забезпечення того, щоб користувацький інтерфейс залишався відгуковим під час очікування відповіді від сервера. Після отримання даних від серверу відбувається їхнє декодування для перетворення в масив об'єктів критеріїв.

Модель Criteria включає ідентифікатор, заголовок та опис, відповідає моделі серверу. Це забезпечує консистентність та спрощує обмін даними між клієнтом та сервером.

Структура Response, що наведено на рисунку 3.7, відповідає інтерфейсу Decodable в Swift, дозволяє зручно декодувати відповідь від серверу з формату Data у структуру Response. Використання Decodable спрощує обробку JSON-відповідей та інтеграцію з сервером.

Ініціалізатор, представлений у нижній частині рисунка 3.6, дозволяє ефективно створювати об'єкти Criteria на основі декодованої відповіді Response. Це важливо для перетворення сирої відповіді сервера у зручний для використання в додатку формат.

Цей підхід забезпечує плавну взаємодію між клієнтською частиною та сервером, дозволяючи користувачам отримувати актуальну інформацію про критерії для формування рекомендацій. Використання стандартних засобів Swift для обробки та декодування даних робить додаток ефективним та надійним з точки зору програмування та користувацького досвіду.

3.2.2. Відправка запиту до серверу для отримання списку рекомендацій

Частина сервісу RecommendationService, наведена на рисунку 3.8, включає методи для виконання запитів до серверу з метою отримання рекомендацій. Повний код сервісу наведено у додатку И. Це важливий компонент, який забезпечує інтерактивність між клієнтом та сервером.

Методи сервісу відрізняються залежно від того, чи вони приймають один критерій або масив критеріїв. Це дозволяє користувачам вибирати, чи вони

хочуть отримати рекомендації на основі одного конкретного критерію, або на основі кількох критеріїв одночасно.

Виконання запитів асинхронно є ключовим для забезпечення відгукового інтерфейсу. Це означає, що додаток не буде "зависати" або виходити з ладу під час очікування відповіді від сервера.

Після отримання відповіді від серверу відбувається декодування даних і створення сутності Recommendations, яка відображає отримані рекомендації. Це забезпечує легке перетворення даних з сервера у формат, зрозумілий для користувача.

Модель Recommendations наведено на рисунку 3.9 та містить в собі масив рядків типу String, представляє отримані рекомендації. Це спрощує відображення рекомендацій у додатку та їх подальше використання.

```
final class RecommendationService: RecommendationUseCase {
        // ...
    func recommendations(by criteria: Criteria) -> AsyncTask<Recommendations> {
        network.reactive
        .request(API.Recommendation.getRecommendations(criteriaID: criteria.id))
        .decode(APIResponse<[String]>.self)
        .map { Recommendations(data: $0.data) }
        .observe(on: UIScheduler())
    }

    func recommendations(for criterias: [Criteria]) -> AsyncTask<[[String]]> {
        network.reactive
            .request(API.Recommendation.recommendations(params: ["id": criterias.map { $0.id }]))
            .decode(APIResponse<[[String]]>.self)
            .map(\.data)
            .observe(on: UIScheduler())
    }
}
```

Рисунок 3.8 – Код запитів для отримання списку рекомендацій

```
struct Recommendations {
    let data: [String]
}
```

Рисунок 3.9 – Модель Рекомендації на клієнтській частині системи

3.2.3. Реалізація графічного інтерфейсу користувача

Для створення користувацького інтерфейсу iOS додатку використано трьохкомпонентну структуру: контролер відображення (view controller), модель відображення (view model) та файл-форму Storyboard. Контролер відображення є центральним елементом, він керує відображеннями в інтерфейсі, включаючи взаємодію з користувачем та відображення даних. Модель відображення, частина патерну MVVM, відокремлює бізнес-логіку від контролера відображення, полегшуючи тестування і підтримку коду. Файл-форма Storyboard дозволяє нам візуально розробляти елементи інтерфейсу, розміщуючи UI компоненти та налаштовуючи їхні взаємодії, що сприяє більш ефективному та інтуїтивному процесу дизайну інтерфейсу.

На рисунку 3.10 наведено фрагмент коду, що належить до контролера представлення нашого додатку, де докладніше описано його функціонал в додатку І. На рисунку 3.11 наведено частину реалізації моделі представлення, з повним описом, розміщеним в додатку Й. На рисунку 3.12 показано візуальне оформлення нашого файлу-форми для користувацького інтерфейсу.

Цей файл-форма, створений у Xcode, відображає макет інтерфейсу користувача, де за допомогою миші ми додали і розташували всі необхідні елементи управління. В нашому випадку, основна частина екрану призначена для UITableView, який використовується для відображення та навігації по списку критеріїв, отриманих від сервера. Також на екрані розташована кнопка, яка при натисканні відкриває список рекомендацій.

Опис класу контролера представлення, який міститься на рисунку 3.10, включає аутлети, пов'язані з елементами управління, розташованими в файлі-формі (рисунок 3.12). Нижче в коді розміщено метод, асоційований з кнопкою, який активується при її натисканні. Цей метод слугує в якості обробника подій для кнопки, що знаходиться в файлі-формі.

```
final class ChooseCriteriaVC: BaseVC, ViewModelContainer {

    @IBOutlet private(set) weak var proceedButton: UIButton!
    @IBOutlet private(set) weak var tableView: UITableView!

    @IBAction private func makeRecommendations(_ sender: UIButton) {
        viewModel.fetchRecommendations()
    }

    func didSetViewModel(_ viewModel: ChooseCriteriaVM, lifetime: Lifetime) {
        reactive.activity <~ viewModel.actions.isExecuting
        reactive.errors <~ viewModel.actions.errors
        reactive.makeBindingTarget { viewController, criterias in
            viewController.viewModel.updateDatasource(with: criterias)
        } <~ viewModel.criterias.signal.take(during: lifetime)
        tableView.reactive.reloadData <~ viewModel.datasource.signal.mapToVoid()
        proceedButton.reactive.isEnabled <~ viewModel.selectedCriteria.producer.map { !$0.isEmpty }

        reactive.makeBindingTarget { viewController, _ in
            viewController.viewModel.showListOfRecommendations()
        } <~ viewModel.recommendations.signal.take(during: lifetime)
    }
    // ...
}

extension ChooseCriteriaVC: UITableViewDataSource {
    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        viewModel.datasource.value.count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCell(cellClass: CriteriaTVC.self, for: indexPath)
        cell.configure(with: viewModel.datasource.value[indexPath.row])
        return cell
    }
}
```

Рисунок 3.10 – Частина коду контролера представлення

Метод didSetViewModel в контролері представлення задіяний для налаштування взаємодії з моделлю представлення. Контролер представлення, зосереджений на управлінні інтерфейсом користувача, не зберігає в собі даних, а замість цього використовує модель представлення як джерело інформації для відображення. Ця взаємодія працює таким чином: коли користувач взаємодіє з додатком (наприклад, скролить список або натискає кнопку), контролер представлення реагує на ці дії та повідомляє про них модель представлення. У свою чергу, модель представлення може звертатися до сервісів для виконання потрібних запитів. Після отримання або обробки необхідних даних, модель представлення інформує контролер представлення, який потім оновлює

інтерфейс. Саме в методі didSetViewModel встановлюються параметри для цих взаємодій, визначаючи, як контролер представлення має реагувати на події від моделі представлення.

Останні два методи, представлені на рисунку 3.10, займаються відображенням даних у таблиці, яка є частиною файлу-форми Storyboard (рисунок 3.12). Ця таблиця отримує свої дані безпосередньо з моделі представлення, підкреслюючи синергію між цими двома компонентами додатку.

```
final class ChooseCriteriaVM: UseCasesConsumer {
    typealias UseCases = HasRecommendationUseCase

    let actions = ActionGroup()
    let criterias = MutableProperty<[Criteria]>([])
    var title: String { R.string.localizable.chooseCriteriaTitle() }
    var description: String { R.string.localizable.chooseCriteriaDescription() }
    private(set) var datasource = MutableProperty<[UserInterestView.Model]>([])
    private lazy var fetchCriteriasAction = Action(execute: useCases.recommendation.criterias)
    private weak var delegate: ChooseCriteriaVMDelegate?

    init(useCases: UseCases, delegate: ChooseCriteriaVMDelegate) {
        self.useCases = useCases
        self.delegate = delegate
        setupObservers()
    }

    func fetchCriterias() {
        fetchCriteriasAction.apply().start()
    }

    func updateDatasource(with criterias: [Criteria]) {
        criterias.forEach {
            let model = UserInterestView.Model(
                criteria: $0,
                tapHandler: { [weak self] criteria, isOn in
                    isOn ? self?.addCriteria(criteria)
                         : self?.removeCriteria(criteria)
                })
            datasource.value.append(model)
        }
    }

    private func addCriteria(_ criteria: Criteria) {
        selectedCriteria.value.append(criteria)
    }

    private func removeCriteria(_ criteria: Criteria) {
        selectedCriteria.value = selectedCriteria.value.filter { $0.id != criteria.id }
    }

    private func setupObservers() {
        actions.append(fetchCriteriasAction)
        actions.append(fetchRecommendationAction)
        criterias <~ fetchCriteriasAction.values.take(duringLifetimeOf: self)
        recommendations <~ fetchRecommendationAction.values.take(duringLifetimeOf: self)
    }
    // ...
}
```

Рисунок 3.11 – Частина коду моделі представлення

На рисунку 3.11 можна побачити витяг з коду моделі представлення. Цей ключовий клас у структурі нашої програми ініціюється з набором функціональних сервісів, відомих як useCases, які активуються всередині самої

моделі. Це дає можливість моделі представлення використовувати ці сервіси для різноманітних завдань. Крім того, існує механізм делегування, що дозволяє моделі делегувати певні дії іншим частинам програми, що сприяє більшій гнучкості у взаємодії.

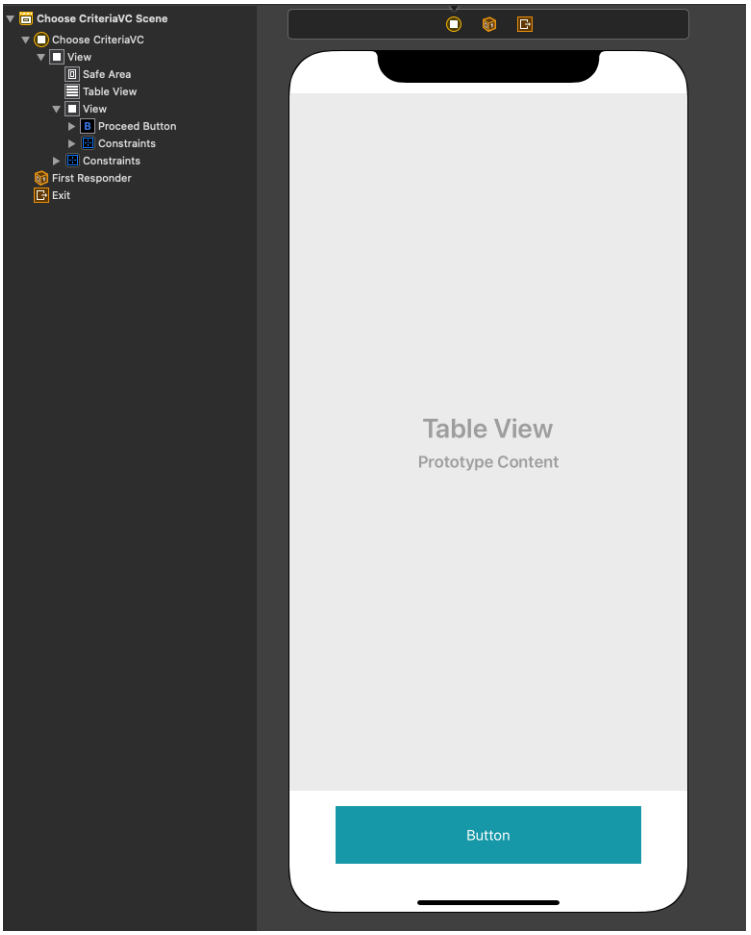

Рисунок 3.12 – Файл-форма з контролерами інтерфейсу користувача

У центрі роботи моделі представлення лежить обробка та управління списком критеріїв. Цей список може бути змінений чи оновлений відповідно до інструкцій, отриманих від контролера представлення, що демонструє тісну взаємодію між цими двома компонентами. Модель представлення також обладнана спеціалізованими методами для виконання запитів, наприклад, для отримання оновленого списку критеріїв.

Важливість цієї моделі полягає у її ролі як посередника між відображенням даних в контролері представлення та різними сервісами в програмі. Така організація сприяє не тільки чистоті та структурованості коду, але й забезпечує його гнучкість та здатність до масштабування, уможливлюючи легкість внесення змін та додавання нових функцій у майбутньому.

3.3. Тестування системи

Після завершення розробки системи проведено її ретельне тестування. На рисунку 3.13 представлено користувацький інтерфейс, який відкривається перед користувачем при запуску додатку.

На початковому екрані користувачу пропонується вибрати критерії, згідно з якими він бажає отримати рекомендації. Спочатку жоден з критеріїв не вибраний, і відповідно кнопка "Отримати Рекомендації" залишається неактивною. Це дозволяє забезпечити, що користувач свідомо вибирає параметри, за якими хоче отримати інформацію. Користувач має можливість вказати кілька критеріїв одночасно, підвищуючи гнучкість вибору. Як тільки хоча б один критерій обрано, кнопка "Отримати Рекомендації" стає активною, що демонструється на рисунку 3.14, дозволяючи користувачу продовжити процес отримання рекомендацій.

3.3.1. Підключення до серверу

Як тільки користувач запускає додаток, відбувається автоматичне підключення до серверу для завантаження актуального списку критеріїв. Цей процес відбувається також при кожному подальшому запиті користувача. У

випадку виникнення проблем з підключенням до серверу, користувачеві відображається повідомлення про помилку, процес якого ілюстровано на рисунку 3.15. Це забезпечує користувачів інформацією про статус їхнього підключення та можливі проблеми.

У ситуації, коли підключення до серверу не вдається та дані не можуть бути отримані, користувачеві показується спеціальний плейсхолдер. Цей плейсхолдер інформує, що дані відсутні, як демонструється на рисунку 3.16. Такий підхід допомагає уникнути плутанини у користувачів щодо стану системи та забезпечує їх ясністю щодо наявності чи відсутності інформації.

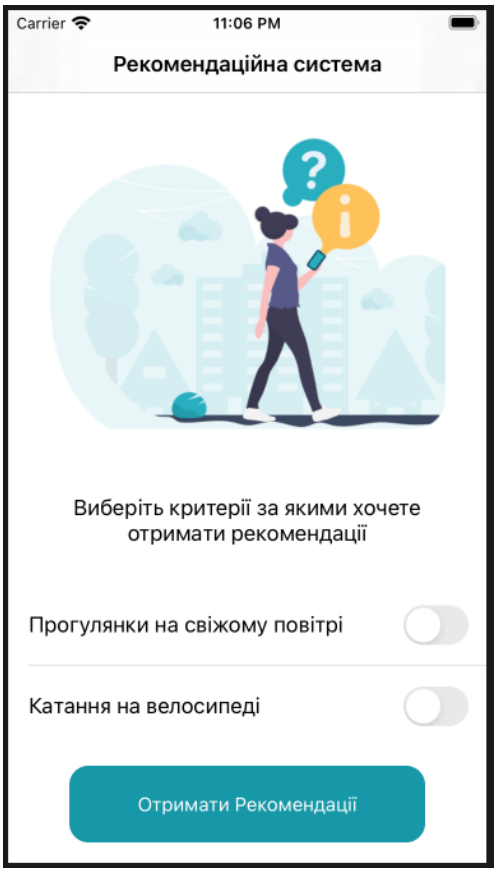

Рисунок 3.13 – Головний екран додатку зі списком критеріїв

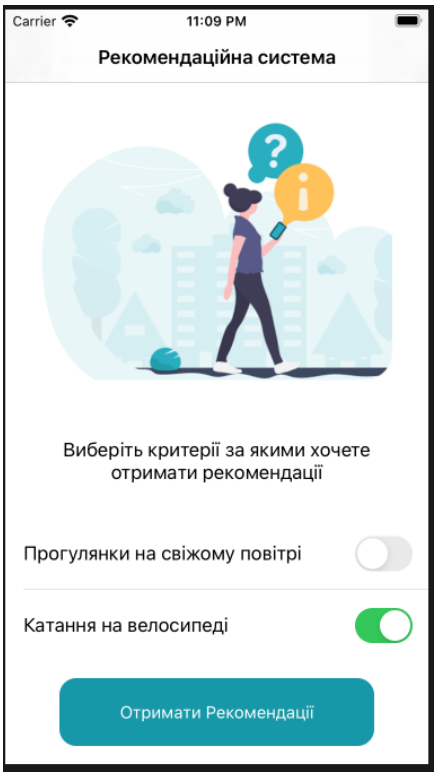

Рисунок 3.14 – Стан інтерфейсу, коли обрано хоча б один критерій

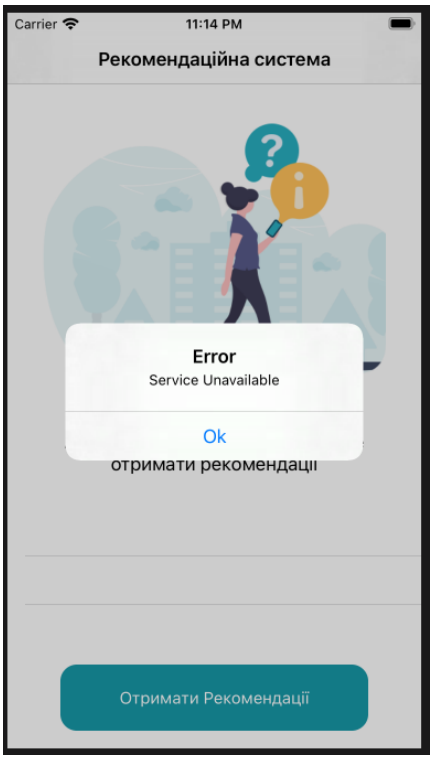

Рисунок 3.15 – Помилка, яка показується користувачу у разі невдачі підключення до серверу

3.2.2. Отримання списку критеріїв

Для моніторингу та аналізу мережевих запитів, які відправляються клієнтом та відповідей, які надходять від сервера, було використано програму Proxyman. Це дозволило детально відслідковувати взаємодію між додатком та сервером.

Запит на отримання списку критеріїв ініціюється автоматично під час запуску додатку. Використання Proxyman дозволило підтвердити, що клієнт успішно відправляє запит, про що свідчить рисунок 3.17. Це вказує на те, що клієнтська частина додатку правильно сформувала та відправила запит.

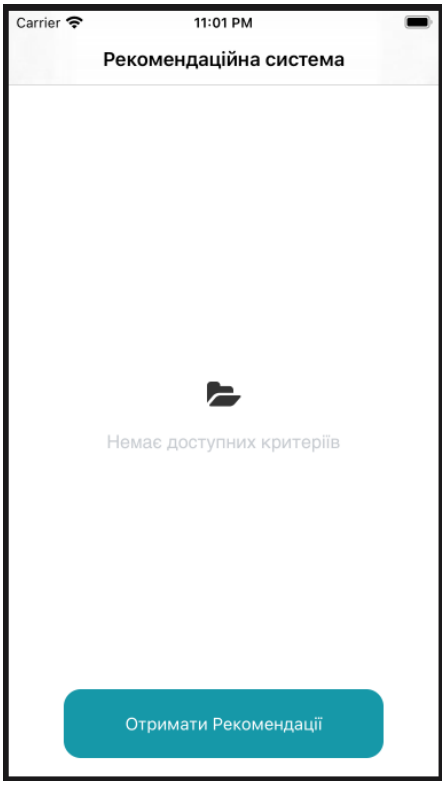

Рисунок 3.16 – Плейсхолдер екрану, коли не вдалось завантажити дані з серверу

Окрім цього, переглянуто логи сервера, які показали, що сервер належним чином обробив запит клієнта. Підтвердження наведено на рисунку 3.18. Такий підхід дозволяє не тільки переконатися у правильності відправлення та обробки запитів, але й забезпечує додаткову впевненість у надійності та стабільності системи.

Після успішного виконання запиту на головному екрані додатку було відображено список критеріїв, які були отримані в результаті. Це відображення, зафіксоване на рисунку 3.19, свідчить про те, що додаток коректно обробив запит, отримав дані від сервера та належним чином представив їх користувачу.

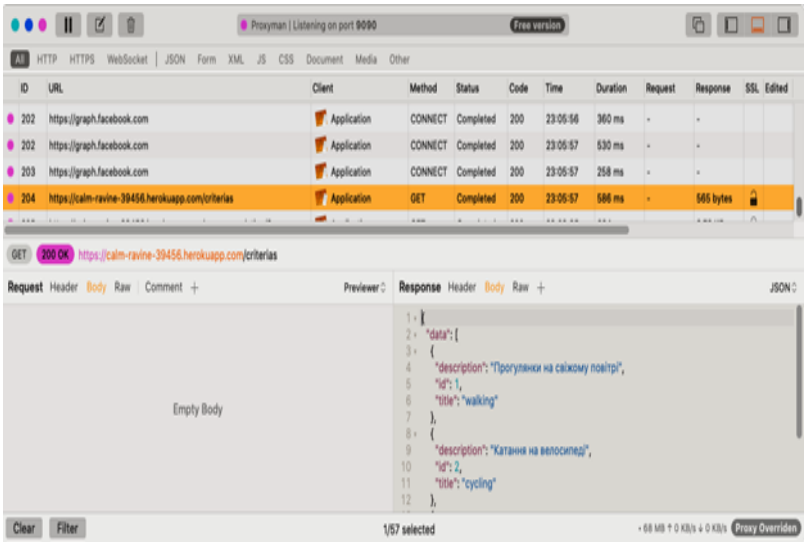

Рисунок 3.17 – Відправка запиту з клієнта до серверу для отримання списку критеріїв

```
2020-12-01T21:05:57.970353+00:00 app[web.1]: 10.171.230.57 - - [01/Dec/2020 21:05:57] "GET /criterias HTTP/1.1" 200 -
2020-12-01T21:05:57.970977+00:00 heroku[router]: at=info method=GET path="/criterias" host=calm-ravine-39456.herokuapp.com request_
id=82119ed8-34f0-4112-bd29-6dc2e84b47a3 fwd="93.76.59.93" dyno=web.1 connect=1ms service=6ms status=200 bytes=712 protocol=https
```

Рисунок 3.18 – Логи серверу, які показують, що сервер коректно опрацював запит від клієнту на отримання списку критеріїв

3.3.3. Отримання списку рекомендацій

У процесі тестування системи перевірено сценарій, коли користувач виконує запит на отримання рекомендацій, використовуючи конкретний критерій. З використанням списку критеріїв, отриманого у розділі 3.2.2, користувач обрав критерій "Прогулянки на свіжому повітрі", що зображено на рисунку 3.20. Вибір цього критерію призвів до активації кнопки "Отримати Рекомендації".

Цей тест підтверджує, що інтерфейс користувача відповідає на дії користувача як очікувалося: активуючи відповідні елементи управління, коли користувач виконує певні дії. Це ключовий аспект для забезпечення інтуїтивно зрозумілого та взаємодіючого користувацького інтерфейсу.

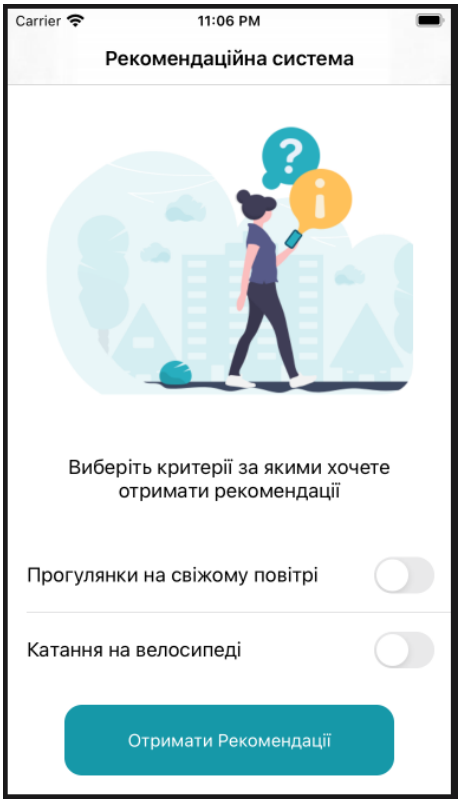

Рисунок 3.19 – Головний екран зі списком критеріїв

Після активації кнопки "Отримати Рекомендації" у додатку, клієнтська частина ініціювала запит до серверу, процес якого демонструється на рисунку 3.21. Під час очікування відповіді від серверу, додаток візуально інформує користувача про цей процес, відображаючи лоадер, як видно на рисунку 3.22. Це важливо для підтримки відмінного користувацького досвіду, оскільки користувачі отримують зворотний зв'язок, що їхній запит обробляється.

Зі свого боку, сервер успішно обробив запит і надіслав назад список рекомендацій, як це показано на рисунку 3.23. Після отримання відповіді від серверу, клієнтський додаток відобразив ці рекомендації користувачеві, що можна побачити на рисунку 3.24. Такий підхід забезпечує, що користувачі отримують актуальну та корисну інформацію, яка відповідає їхнім запитам та інтересам.

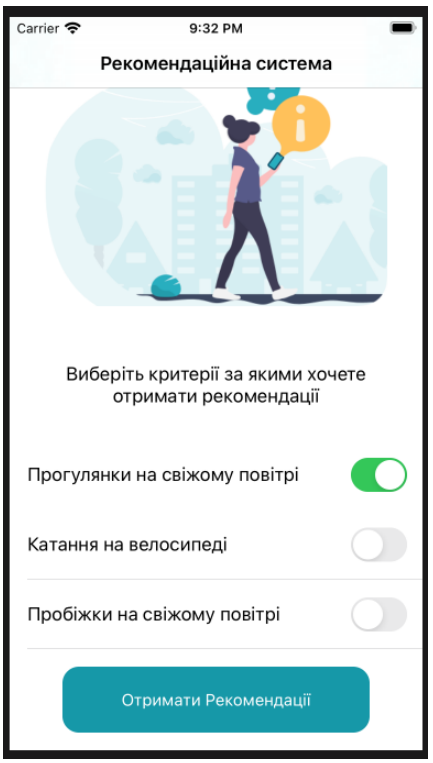

Рисунок 3.20 – Критерій за яким проводилось тестування надання рекомендацій

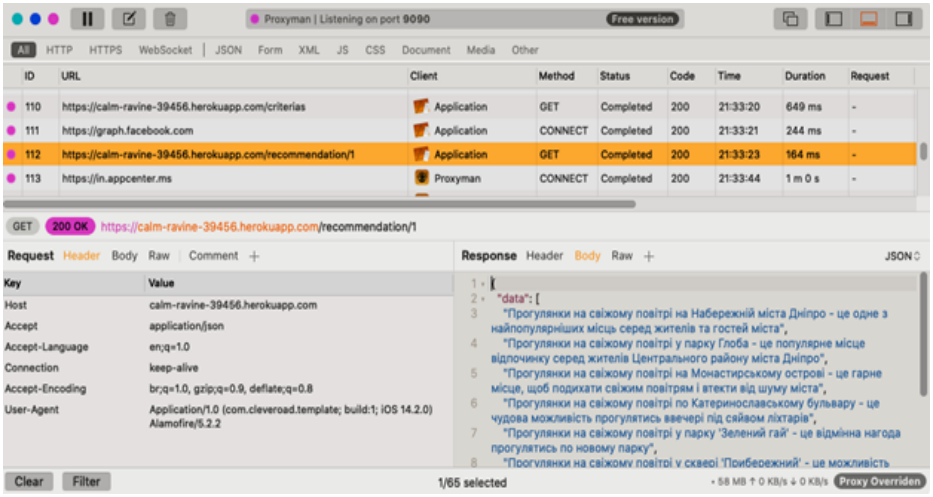

Рисунок 3.21 – Запит до серверу для отримання рекомендацій за вказаним критерієм

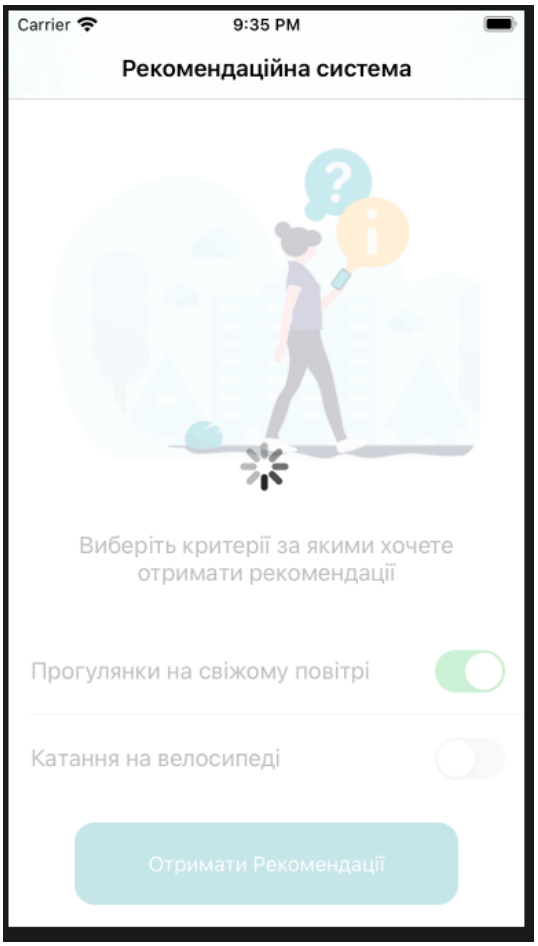

Рисунок 3.22 – Лоадер, що відображається користувачеві під час очікування відповіді від серверу

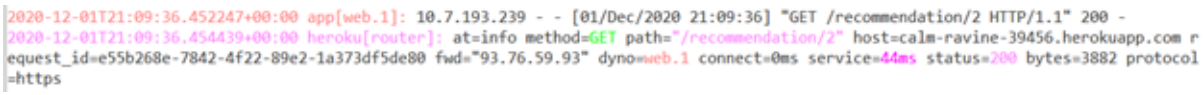

Рисунок 3.23 – Логи серверу, які показують успішне опрацювання запита клієнта на отримання списку рекомендацій

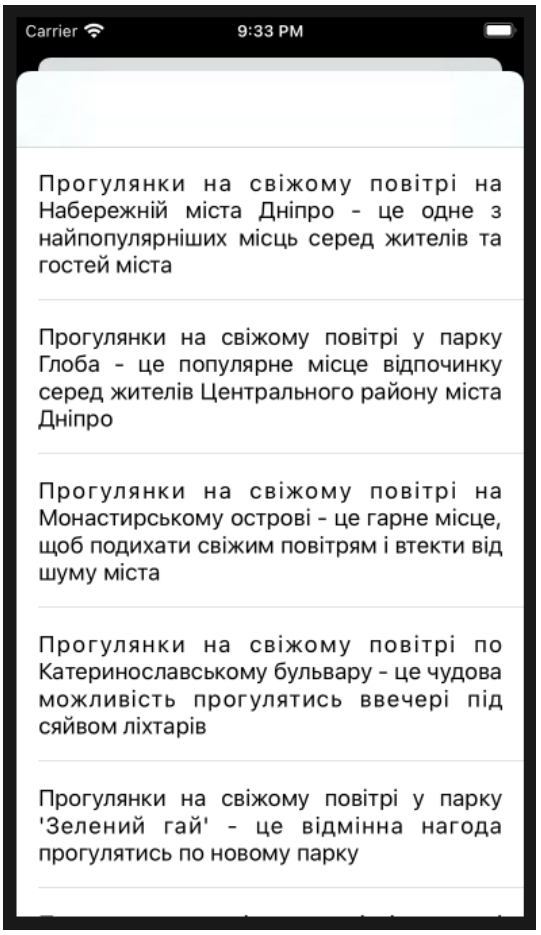

Рисунок 3.24 – Список рекомендацій, що надійшов з серверу

3.4. Висновки

Система є взірцем використання комбінації мов програмування для реалізації різних аспектів системи. Для серверної частини обрано Python для розробки алгоритмів, що були детально описані у другому розділі документації. Вибір Python для цієї частини проекту підкреслює його ефективність у

вирішенні складних алгоритмічних завдань, а також його здатність до швидкого прототипування та гнучкості у впровадженні комплексних рішень.

Для розробки клієнтської частини, яка відповідає за інтерактивне спілкування з користувачами та надання їм рекомендацій, обрано Swift. Цей вибір відзеркалює переваги Swift у створенні інтуїтивних та відповідальних мобільних додатків, особливо для екосистеми Apple. Swift забезпечує високу продуктивність, сучасні можливості безпечного програмування та чистоту коду, що є критично важливим для розробки надійних додатків.

Ретельне тестування системи на кожному етапі розробки гарантувало її стабільність та функціональність. Це включало перевірку зв'язку клієнт-сервер, відслідковування запитів та відповідей за допомогою інструментів моніторингу, таких як Proxyman, та візуалізацію даних на користувацькому інтерфейсі. Підхід до тестування забезпечив, що система не тільки відповідає технічним вимогам, але й пропонує високоякісний користувацький досвід, включаючи інтуїтивно зрозумілі елементи навігації та відображення інформації.

Таким чином, система (програмний засіб) є прикладом ефективного використання різних технологій та методологій, що забезпечує комплексний підхід до розробки та впровадження сучасних програмних рішень.

4. ДОСЛІДЖЕННЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ РЕКРЕАЦІЙНИХ ЗОН

По завершенню розробки системи, було ініційовано аналітичне дослідження процедури створення рекомендацій для рекреаційних областей. Детальний опис цього дослідження, а також висновки, до яких ви прийшли, були викладені у цьому розділі.

4.1. Набір даних для проведення дослідження

Перед початком процесу дослідження, фундаментальним кроком є розробка та підготовка комплексного датасету, який служитиме як основа для аналізу та вивчення системи. Важливість цього набору даних не можна недооцінювати, адже від його якості, структури та вичерпності безпосередньо залежатиме рівень точності та релевантності рекомендацій, які система зможе згенерувати.

Склад датасету полягає у великому переліку рекреаційних зон, де кожен елемент містить унікальний ідентифікатор та детальний опис кожної зони відпочинку. Для дослідження зібрано дані про приблизно сто таких місць, які містяться в місті, і кожне з них було уважно вибране та описане особисто. Цей підхід до збору даних свідчить про вашу грунтовність і прагнення забезпечити найвищу якість даних, які ляжуть в основу майбутнього аналізу та оцінювання ефективності системи рекомендацій.

4.2. Проведення досліджень

У рамках першого експерименту проведено аналіз, використовуючи традиційний підхід TF-IDF та косинусну схожість для формування рекомендацій.

Рисунок 4.1 демонструє вибрані критерії для отримання рекомендацій, тоді як рисунок 4.2 показує можливі варіанти рекомендацій, які присутні у датасеті. Рисунок 4.3 відображає кінцевий список рекомендацій, що були

вибрані системою. На рисунку 4.4 представлені логи сервера, які фіксують процес формування цих рекомендацій.

З цих рисунків можна зробити висновок, що застосування методу TF-IDF та косинусної схожості, а також інструментів з бібліотеки sklearn, привело до отримання задовільних результатів. Особливо це підтверджується рівнем значення score, зафіксованим у логах сервера на рисунку 4.4. Вище значення score означає кращу відповідність рекомендацій до обраних критеріїв. Результати показали, що значення score коливаються в діапазоні від 0.75 до 0.9, що свідчить про високу точність вибору рекомендацій системою. Середнє значення становить 0.85.

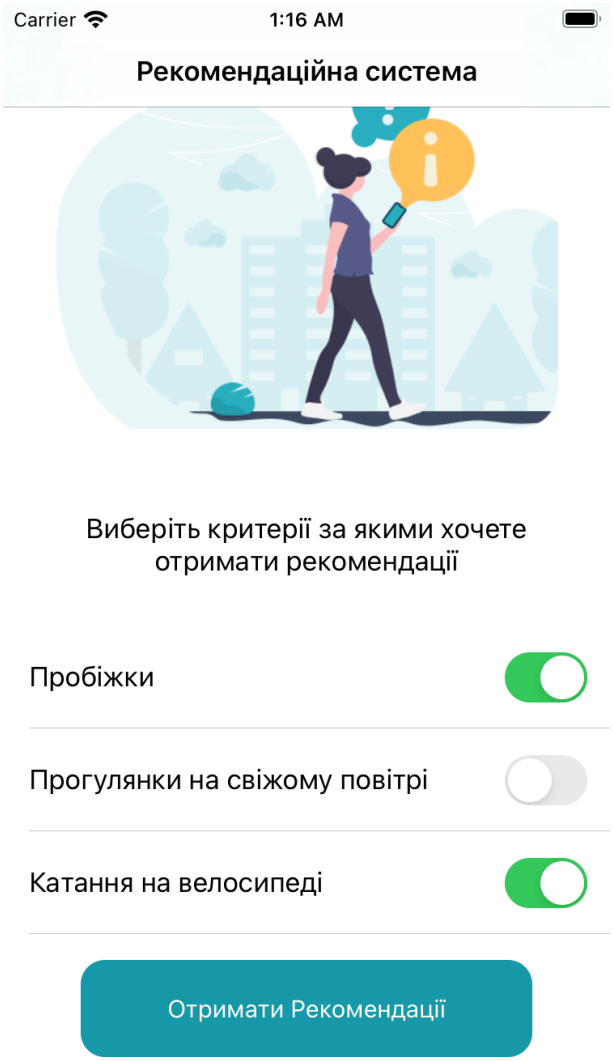

Рисунок 4.1 – Обрані критерії для рекомендацій

```
9,"Пробіжки у парку Глоба - це можливість для бігу в локації в центрі міста"
10,"Пробіжки на Монастирському острові - це пробіжки по рельєфній місцевості, де знаходиться невелика кількість людей"
11,"Пробіжки у парку 'Зелений гай' - це чудова місцевість, гарні краєвиди, обладнані доріжки для бігу"
```

```
18,"Катання на велосипеді у сквері 'Прибережний' - це оснащені велосипедні доріжки для комфортної їзди"
19,"Катання на велосипеді у сквері 'Героїв' - це невелика зона з асфальтними доріжками для помірної їзди"
```

Рисунок 4.2 – Потенційні рекомендації за обраними критеріями

Рисунок 4.3 – Отримані рекомендації за обраними критеріями

```
Recommended: Катання на велосипеді на Набережній міста Дніпро - це розлога довга алея в центрі міста (score: 0.8549852)
Recommended: Катання на велосипеді у парку Глоба - це просторний парк з рельєфними алеями (score: 0.8392481)
Recommended: Катання на велосипеді на Монастирському острові - це можливість покататись библизу пляжу та проїхатись по пішоходному містку (score: 0.8009831)
Recommended: Катання на велосипеді у парку 'Зелений гай' - це великий парк який підійде для велопрогулянок, як для новачків так і для професіоналів (score: 0.7883163)
Recommended: Катання на велосипеді у сквері 'Прибережний' - це оснащені велосипедні доріжки для комфортної їзди (score: 0.78749912)
Recommended: Катання на велосипеді у сквері 'Героїв' - це невелика зона з асфальтними доріжками для помірної їзди (score: 0.7791692)
Recommended: Пробіжки на Набережній міста Дніпро - це чудова локація для бігу для підтримки своєї форми з можливістю подихати свіжим повітрям з річки Дніпро (score: 0.89039283)
Recommended: Пробіжки у парку Глоба - це можливість для бігу в локації в центрі міста (score: 0.86038592)
Recommended: Пробіжки на Монастирському острові - це пробіжки по рельєфній місцевості, де знаходиться невелика кількість людей (score: 0.82848591)
Recommended: Пробіжки у парку 'Зелений гай' - це чудова місцевість, гарні краєвиди, обладнані доріжки для бігу (score: 0.81470481)
Recommended: Пробіжки у сквері 'Прибережний' - це обладнані доріжки в оновленому парку в локації ж\м Перемога (score: 0.76372048)
```

Рисунок 4.4 – Логи серверу під час генерації рекомендацій

У другому експерименті внесено зміни у метод обчислення Term Frequency (TF), перейшовши до бінарного підходу. Це означає, що замість підрахунку кількості разів, коли термін зустрічається у тексті, просто визначається наявність терміну, присвоюючи значення одиниці у випадку його присутності [27].

Для цього експерименту використано ті самі критерії відбору рекомендацій, що й у попередньому експерименті, які були представлені на рисунку 4.1. Аналогічно, потенційний список рекомендацій, показаний на рисунку 4.2, також залишився незмінним. Результати цього експерименту, у вигляді отриманих рекомендацій, були представлені на рисунку 4.5, а процес генерації рекомендацій та відповідні логи серверу можна побачити на рисунку 4.6.

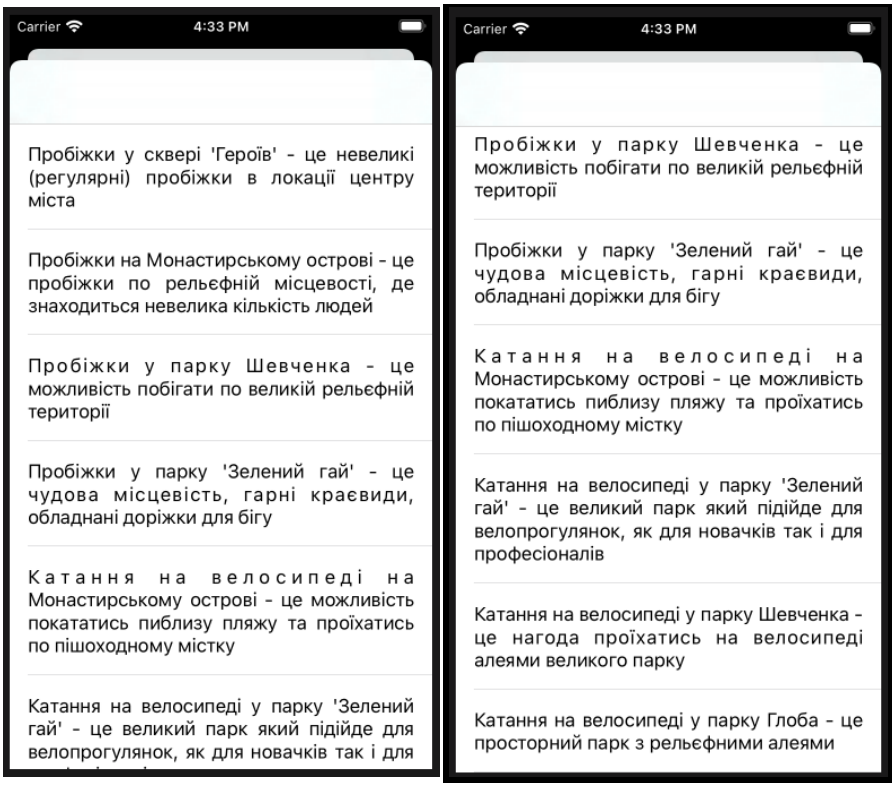

Рисунок 4.5 – Отримані рекомендації при застосуванні бінарного обчислення TF

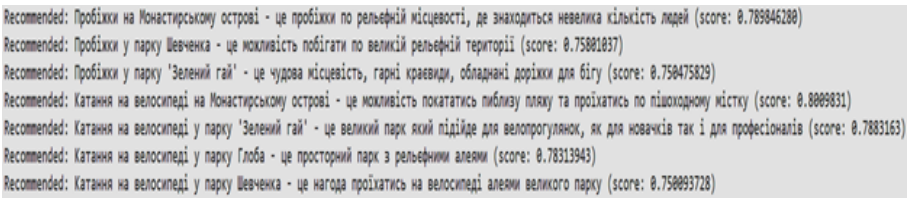

Рисунок 4.6 – Логи серверу під час генерації рекомендацій при застосуванні бінарного обчислення TF

Аналізуючи результати другого експерименту, представлені на рисунках 4.5 та 4.6, виявлено, що ефективність рекомендаційних результатів в цьому випадку була трохи гіршою порівняно з попереднім експериментом. Один із індикаторів цього – зменшення загальної кількості рекомендацій: якщо в попередньому експерименті за двома критеріями було більше десяти рекомендацій, то в цьому випадку ви отримали лише вісім.

Крім того, значення score, які служать мірою відповідності рекомендацій до заданих критеріїв, також були нижчими в цьому експерименті. Середнє значення score становило приблизно 0.77, у той час як у попередньому експерименті це значення досягало близько 0.85. Це свідчить про те, що зміна методу обчислення TF зі звичайного на бінарний може мати значний вплив на релевантність та точність рекомендацій, що надає система.

У третьому експерименті було вирішено використати метод обчислення Term Frequency (TF) без нормалізації. Це означає, що частота слова, що зустрічається в тексті, вимірювалася без поділу цієї частоти на загальну кількість слів у тексті. Цей підхід дозволяє оцінити вплив абсолютної частоти термінів на якість рекомендацій [22], [28].

Обрані для експерименту критерії залишилися незмінними, як і на рисунку 4.1, та потенційні рекомендації, представлені на рисунку 4.2. Результати, у вигляді отриманих рекомендацій, були представлені на рисунку 4.7, тоді як процес генерації цих рекомендацій та відповідні логи серверу зображені на рисунку 4.8.

Метою цього експерименту було виявити, наскільки відсутність нормалізації частоти термінів впливає на релевантність та точність рекомендацій системи, у порівнянні з попередніми методами обчислення TF.

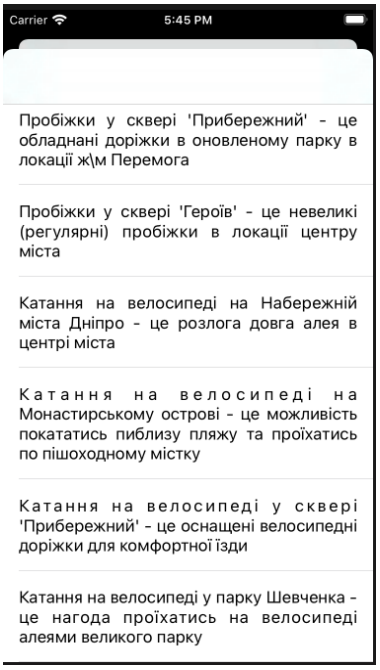

Рисунок 4.7 – Отримані рекомендації при застосуванні обчислення TF без нормалізації

```
Recommended: Пробіжки у сквері 'Прибережний' - це обладнані доріжки в оновленому парку в локації ж\м Перемога (score: 0.78372048)
Recommended: Пробіжки у сквері 'Героїв' - це невеликі (регулярні) пробіжки в локації центру міста (score: 0.75048248)
Recommended: Катання на велосипеді на Набережній міста Дніпро - це розлога довга алея в центрі міста (score: 0.8149852)
Recommended: Катання на велосипеді на Монастирському острові - це можливість покататись поблизу пляжу та проїхатись по пішоходному містку (score: 0.8009831)
Recommended: Катання на велосипеді у сквері 'Прибережний' - це оснащені велосипедні доріжки для комфортної їзди (score: 0.78749912)
Recommended: Катання на велосипеді у парку Шевченка - це нагода проїхатись на велосипеді алеями великого парку (score: 0.750093728)
```

Рисунок 4.8 – Логи серверу під час генерації рекомендацій при застосування обчислення TF без нормалізації

Аналізуючи дані, представлені на рисунках 4.7 та 4.8, можна зробити висновок, що результати третього експерименту були схожі на результати другого. Основним показником цього є кількість отриманих рекомендацій, зображених на рисунку 4.7, де було надано 6 рекомендацій, у порівнянні з 8 у попередньому експерименті. Також важливим індикатором є значення score,

зафіксовані на рисунку 4.8, де середнє значення score також становило приблизно 0.77, що аналогічно до попереднього експерименту.

Ці результати вказують на те, що зміна методу обчислення TF без нормалізації не мала значного впливу на кількість та якість рекомендацій, порівняно з бінарним підходом до обчислення TF.

У четвертому експерименті вирішено застосувати метод обчислення Term Frequency (TF) з нормалізацією за допомогою логарифму. Це означає, що частота кожного слова в тексті була нормалізована логарифмічно, щоб зменшити вплив високочастотних слів і підвищити значущість рідкісних слів, тобто:

$$TF_a = 1 + lg(N_a), \tag{4.1}$$

де N_a – кількість разів, коли термін a зустрічається у тексті, TF_a – частотність терміну a [24], [25].

Для четвертого експерименту було збережено ті ж критерії вибору рекомендацій, що й у попередніх дослідженнях, які демонструються на рисунку 4.1. Також не зазнали змін і потенційні рекомендації, що відображені на рисунку 4.2. На рисунку 4.9 представлено список рекомендацій, які були отримані в результаті цього експерименту. Логи серверу, що зафіксували процес генерації цих рекомендацій, наведені на рисунку 4.10.

Ці матеріали дозволяють оцінити ефективність використання логарифмічно нормалізованого підходу до обчислення TF і його вплив на результати рекомендаційної системи.

Рисунок 4.9 – Отримані рекомендації при застосуванні обчислення TF з нормалізацією частоти слова за допомогою логарифму

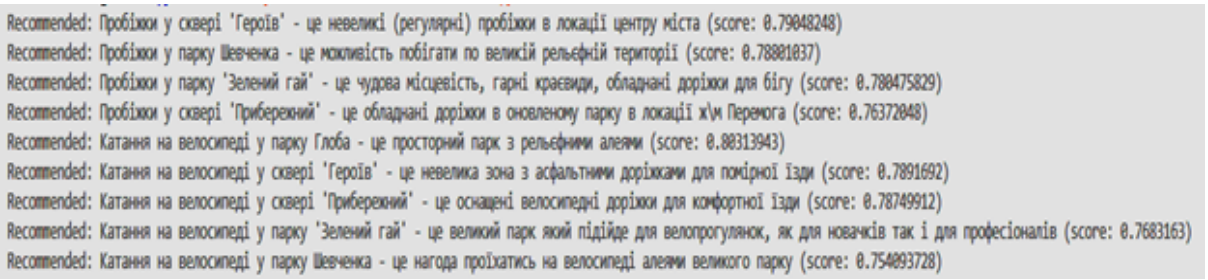

Рисунок 4.10 – Логи серверу під час генерації рекомендацій при застосуванні обчислення TF з нормалізацією частоти слова за допомогою логарифму

Оглядаючи дані, представлені на рисунках 4.9 та 4.10, можна відмітити, що результати четвертого експерименту з логарифмічно нормалізованим підходом до обчислення TF виявилися трохи кращими, порівняно з двома попередніми експериментами. Проте вони все ще були не настільки ефективними, як результати, отримані в першому експерименті. У цьому

експерименті було визначено 9 рекомендацій, як видно на рисунку 4.9, та середнє значення score досягло приблизно 0.78.

Ці результати вказують на те, що використання логарифмічної нормалізації в методі TF може покращити точність рекомендацій, хоча і не досягло рівня результатів першого експерименту.

У п'ятому експерименті було застосовано метод обчислення Term Frequency (TF) з подвійною нормалізацією частоти слова. Цей підхід передбачає ще більш точне вирівнювання ваги термінів у тексті, що може підвищити релевантність та точність рекомендацій, згенерованих системою. Подвійна нормалізація має на меті збалансувати вплив як високо-, так і низькочастотних слів, щоб забезпечити більш глибокий аналіз контенту і, як наслідок, більш точні рекомендації, тобто:

$$TF_a = 0.5 + 0.5(\frac{N_a}{N}), \quad (4.2)$$

де N_a – кількість разів, де термін a зустрічався у тексті, N – кількість разів, яку зустрівся найчастотніший термін цього тексту [26], [27].

У рамках п'ятого експерименту з подвійною нормалізацією частоти слів у підході TF, критерії вибору рекомендацій залишилися незмінними, як і було вказано на рисунку 4.1. Також не були змінені потенційні рекомендації, які вказано на рисунку 4.2. Результати цього експерименту, у вигляді отриманих рекомендацій, були представлені на рисунку 4.11. Логи серверу, які зафіксували процес генерації цих рекомендацій, зображені на рисунку 4.12.

Ці матеріали дають можливість оцінити, наскільки ефективною була подвійна нормалізація у методі обчислення TF та який вплив це мало на кінцеві результати рекомендаційної системи.

Рисунок 4.11 – Отримані рекомендації при застосуванні обчислення TF з подвійною нормалізацією частоти слова

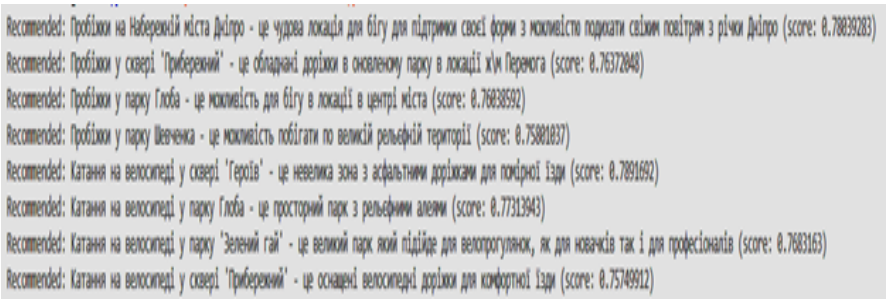

Рисунок 4.12 – Логи серверу під час генерації рекомендацій при застосуванні обчислення TF з подвійною нормалізацією частоти слова

На основі даних, представлених на рисунках 4.11 та 4.12, можна відзначити, що результати п'ятого експерименту з подвійною нормалізацією TF виявилися схожими на результати попереднього, четвертого експерименту. В цьому експерименті система сформувала 8 рекомендацій, як видно на рисунку 4.11. Також середнє значення score, що відображає релевантність рекомендацій

до заданих критеріїв, склало приблизно 0.78, що є порівнянним з результатами попереднього експерименту.

Ці результати свідчать про те, що хоча подвійна нормалізація TF має свої переваги, вона не призвела до значного покращення результатів у порівнянні з одинарною нормалізацією, яка була використана в четвертому експерименті.

У шостому експерименті було обрано метод обчислення Inverse Document Frequency (IDF) з пом'якшенням. Такий підхід включає зміну традиційної формули IDF, щоб зменшити вплив рідкісних термінів, які можуть бути надмірно вагомими у стандартному розрахунку IDF. Застосування пом'якшеного IDF має на меті створити більш збалансований та реалістичний підхід до визначення важливості слів у документах та покращити точність рекомендацій, тобто:

$$IDF_a = lg(1 + (\frac{N}{N_a})), \quad (4.3)$$

де N – загальна кількість документів, N_a – кількість документів в яких зустрічається термін a [27], [28].

У рамках шостого експерименту з використанням пом'якшеного IDF, критерії для відбору рекомендацій залишалися такими ж, як було визначено на рисунку 4.1, а також не змінилися потенційні рекомендації, представлені на рисунку 4.2. Результати цього експерименту, у вигляді отриманих рекомендацій, були ілюстровані на рисунку 4.13. Також на цьому ж рисунку (4.13) представлені логи серверу, які зафіксували процес генерації рекомендацій.

Цей етап дослідження дозволяє оцінити, як зміна методу обчислення IDF вплине на результативність та релевантність рекомендацій, порівняно з попередніми методами.

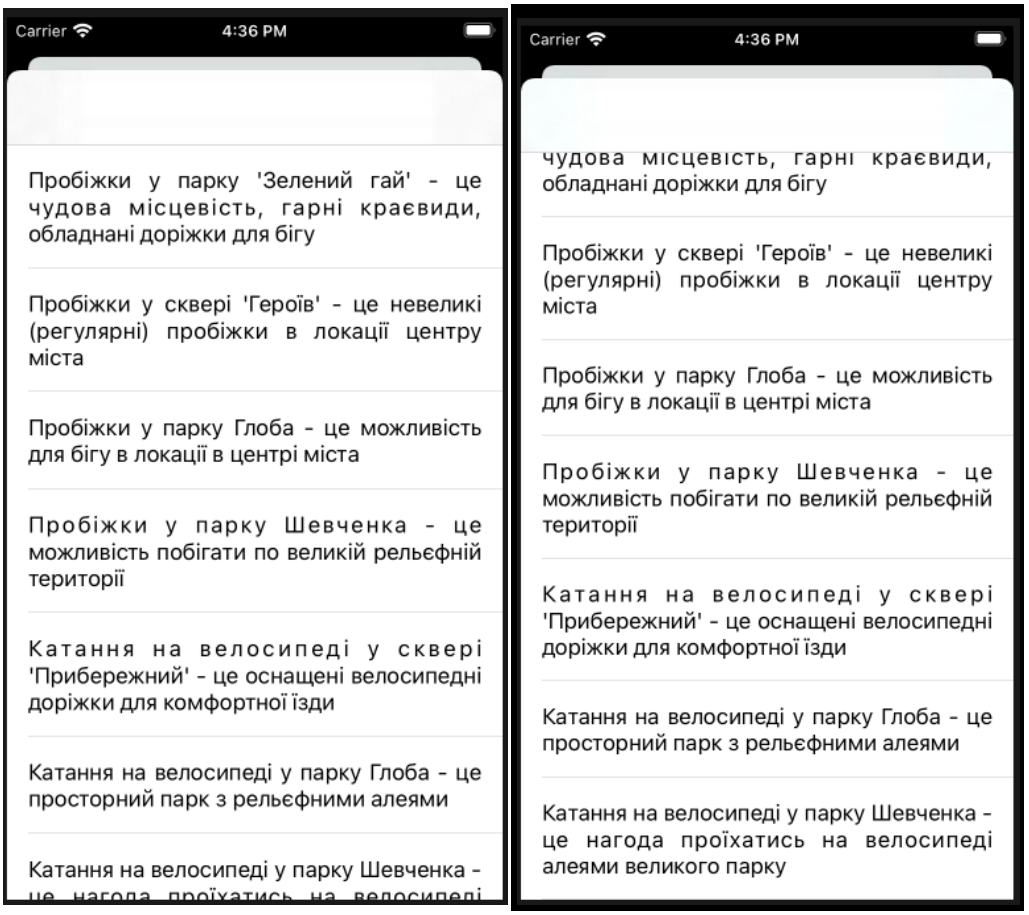

Рисунок 4.13 – Отримані рекомендації при застосуванні обчислення IDF з пом’якшенням

```
Recommended: Пробіжки у парку 'Зелений гай' - це чудова місцевість, гарні краєвиди, обладнані доріжки для бігу (score: 0.770478429)
Recommended: Пробіжки у сквері 'Героїв' - це невеликі (регулярні) пробіжки в локації центру міста (score: 0.76142248)
Recommended: Пробіжки у парку Глоба - це можливість для бігу в локації в центрі міста (score: 0.76038862)
Recommended: Пробіжки у парку Шевченка - це можливість побігати по великій рельєфній території (score: 0.75109517)
Recommended: Катання на велосипеді у сквері 'Прибережний' - це оснащені велосипедні доріжки для комфортної їзди (score: 0.75749912)
Recommended: Катання на велосипеді у парку Глоба - це просторний парк з рельєфними алеями (score: 0.75590243)
Recommended: Катання на велосипеді у парку Шевченка - це нагода проїхатись на велосипеді алеями великого парку (score: 0.7540908328)
```

Рисунок 4.14 – Логи з серверу під час генерації рекомендацій при застосуванні обчислення IDF з пом’якшенням

На основі даних, наведених на рисунках 4.13 та 4.14, можна зробити висновок, що результати шостого експерименту з пом'якшеним IDF виявилися схожими на ті, які були отримані у третьому експерименті. У цьому

експерименті система надала 7 рекомендацій, з середнім значенням score приблизно 0.76. Такий результат можна пояснити особливостями застосування пом'якшеного IDF, який забезпечує більш гладке розподілення вагомості між часто та рідко використовуваними термінами. Такий підхід призводить до розмивання межі між важливими та менш важливими термінами у контексті документа, що може вплинути на загальну якість рекомендацій, як це й було виявлено у даному експерименті.

Результати експериментів (значення показника score) наведено у таблиці 4.1.

Таблиця 4.1 - Результати шістьох експериментів (значення показника *score*)

№ експерименту	1	2	3	4	5	6
Значення score	0.85	0.77	0.77	0.78	0.78	0.76

4.3. Висновки

Було проведено експерименти з формуванням рекомендацій рекреаційних зон міста, в яких використовувалися різні методи обчислення Term Frequency (TF) та Inverse Document Frequency (IDF). Основна увага була зосереджена на стандартній реалізації TF-IDF за допомогою інструментів бібліотеки sklearn, а також на альтернативних підходах, які включали бінарне обчислення TF, обчислення TF без нормалізації, з логарифмічною нормалізацією, подвійною нормалізацією, а також змінене обчислення IDF з пом’якшенням.

Експериментально було встановлено, що найефективнішою для формування рекомендацій рекреаційних зон міста виявилася стандартна конфігурація TF-IDF з використанням інструментів бібліотеки sklearn. Важливим аспектом цього успіху є можливість ігнорування певних слів, які несуть мінімальну інформаційну цінність для алгоритму (наприклад, сполучники, займенники), що дозволяє підвищити вагу ключових термінів. Цей

підхід сприяє більшій кількості та якості рекомендацій, оскільки збільшується релевантність ключових слів у контексті рекомендацій.

У порівнянні зі стандартним підходом, модифікації TF-IDF не показали значного покращення у якості рекомендацій. Це може бути зумовлено втратою можливості ігнорування менш важливих слів, що знижує різноманітність та точність визначення важливості термінів.

Загалом, проведені експерименти вказують на важливість правильного вибору методів обчислення TF та IDF для створення ефективних рекомендаційних систем. Класичний підхід TF-IDF, незважаючи на свою простоту, продемонстрував найкращі результати, що свідчить про його надійність і ефективність у визначенні релевантності контенту.

Таким чином, у даній роботі було проведено порівняльний аналіз якості рекомендацій при використанні стандартного TF-IDF та його модифікацій, демонструючи, що класичний підхід з використанням інструментів бібліотеки sklearn залишається найбільш ефективним для формування рекомендацій рекреаційних зон міста.

5. ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має важливе значення та перспективу застосування, коли вона відповідає сучасним вимогам і тенденціям, як у сфері наукових досягнень, так і в аспектах економічної ефективності. Оцінка економічної ефективності науково-дослідної роботи є критичною для визначення її практичної цінності та можливостей впровадження.

Магістерська кваліфікаційна робота на тему "Метод і програмний засіб рекомендацій рекреаційних зон" належить до категорії науково-технічних розробок, що орієнтовані на виходження на ринок або на прийняття рішення про комерціалізацію під час самої роботи. Такий підхід є пріоритетним, оскільки результати розробки можуть бути використані широким колом споживачів, що приносить економічну вигоду. Однак для цього необхідно залучити потенційного інвестора та довести йому економічну доцільність реалізації проекту.

Для успішної реалізації задуму слід виконати такі етапи:

1. Провести комерційний аудит науково-технічної розробки для визначення її науково-технічного рівня та комерційних перспектив;
2. Розрахувати витрати, пов'язані з реалізацією науково-технічної розробки;
3. Визначити економічну ефективність розробки у випадку її впровадження та комерціалізації потенційним інвестором, а також обґрунтувати економічну доцільність таких дій.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного та технологічного аудиту є оцінювання комерційного потенціалу розробка методу і засобів рекомендацій рекреаційних зон.

Для проведення технологічного аудиту було залучено 3-х незалежних експертів Вінницького національного технічного університету: Тужанський Станіслав Євгенович (к.т.н., доц. кафедри ПЗ ВНТУ), Збитківський Владислав Сергійович (студент кафедри ПЗ ВНТУ), Нестерук Віталій Андрійович (студент кафедри ПЗ ВНТУ).

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [29].

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

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою

Продовження табл. 5.1

7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансуванн я є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисл овому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлен ня відповідним органам про виробництв о та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Таблиця 5.2 – Рівні комерційного потенціалу розробки [29]

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вищий середнього
21…30	Середній
11…20	Нижчий середнього
0…10	Низький

В таблиці 5.3 наведено результати оцінювання експертами комерційного потенціалу розробки.

Таблиця 5.3 – Результати оцінювання комерційного потенціалу розробки

Критерії	Експерт (ПІБ, посада)		
	Тужанський С.Є.	Збитківський В.С.	Нестерук В.А.
	Бали, виставлені експертами:		
1. Технічна здійсненність концепції	4	4	4
2. Ринкові переваги (наявність аналогів)	2	2	2
3. Ринкові переваги (ціна продукту)	3	2	3
4. Ринкові переваги (технічні властивості)	3	4	3
5. Ринкові переваги (експлуатаційні витрати)	2	2	2
6. Ринкові перспективи (розмір ринку)	4	3	3
7. Ринкові перспективи (конкуренція)	3	3	3
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	2	2	1
10. Практична здійсненність (необхідність нових матеріалів)	3	2	3

Продовження табл. 5.3.

11. Практична здійсненність (термін реалізації)	3	4	4
12. Практична здійсненність (розробка документів)	4	3	3
Сума балів	$СБ_1$=37	$СБ_2$=35	$СБ_3$=35
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{37+35+35}{3} = 35.6$		

Середньоарифметична сума балів, розрахована на основі висновків експертів склала 35.6 бали, що згідно таблиці 5.2 вважається, що рівень комерційного потенціалу проведених досліджень є вище середнього.

Метод і програмний засіб рекомендацій рекреаційних зон, що розробляються в магістерській роботі будуть цікаві підприємствам, що займаються створенням туристичних екскурсій та маршрутів. Розроблений програмний засіб допоможе їм створити туристичні пропозиції, які будуть більш наближені до потреб користувачів.

Оскільки в основі рекомендаційних систем лежать алгоритми фільтрації даних, які і забезпечують коректне надання рекомендацій користувачам. В нашому випадку система використовує фільтрацію на основі вмісту, тому порівняємо нашу нову розробку, що розробляється в магістерській роботі з аналогом, який існує на ринку і використовує також фільтрацію на основі вмісту.

В якості аналога для розробки було обрано сервіс рекомендацій музики Last.fm. Основним недоліком аналога є те, що він не надає можливості працювати з модифікаціями алгоритму TF-IDF, а тільки лише зі стандартною реалізацією.

У розробці дана проблема вирішується за рахунок використання мови Python та бібліотеки sklearn та власноруч написаних модифікацій алгоритму TF-IDF.

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

Таблиця 5.4 – Основні параметри нової розробки та товару-конкурента

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
1	2	3	4	5
Точність пошуку, бали	9	9	1	30%
Точність пошуку на даних з помилками, бали	5	8	1,6	30%
Середній час відповіді на запит *(менше – краще)*, мс	110	100	1,1	10%
Кількість параметрів слова при аналізі, шт	768	768	1	20%
Використання ресурсів комп'ютера, %	80	80	1	10%

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

Визначимо відносні одиничні показники якості по кожному параметру за формулами (5.1) та (5.2) [29]і занесемо їх у відповідну колонку табл. 5.5.

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

або

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

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

$$q_1 = \frac{9}{9} = 1;$$

$$q_2 = \frac{8}{5} = 1,6;$$

$$q_3 = \frac{110}{100} = 1,1;$$

$$q_4 = \frac{768}{768} = 1;$$

$$q_5 = \frac{80}{80} = 1.$$

Груповий параметричний індекс за технічними показниками розраховується за формулою [29]:

$$\mathrm{I_{TP}} = \sum_{i=1}^{n} a_i \bullet q_i, \qquad (5.3)$$

де q_i - одиничний параметричний показник, a_i - вагомість параметричного показника.

Розрахуємо значення групового індекса за технічними показниками

$$I_{TP} = 1 \cdot 0,3 + 1,6 \cdot 0,3 + 1,1 \cdot 0,1 + 1 \cdot 0,2 + 1 \cdot 0,1 = 2,18$$

Зважаючи на розрахунки, можна зробити висновок, що нова розробка буде конкурентоспроможнішою, ніж конкурентна система.

5.2.Розрахунок витрат на здійснення науково-дослідної роботи

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

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

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{Т_р} \text{ (грн)} \quad (5.6)$$

де M_{ni} – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

T_p – число робочих днів в місяці; приблизно $T_p \approx 21..23$ дні;

t_i – число робочих днів роботи дослідника;

k – кількість посад.

Для розробки засобу рекомендацій рекреаційних зон необхідно залучити програміста з посадовим окладом 8000 грн. Кількість робочих днів у місяці складає 22, а кількість робочих днів програміста складає 51. Зведемо сумарні розрахунки до таблиця 5.5.

Таблиця 5.5 – Заробітна плата дослідника в науковій установі бюджетної сфери

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник	12000	545,5	51	27 820,5
Програміст	8000	363,6	51	18545
Всього				46 365,5

2. Розрахунок додаткової заробітної плати робітників

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

На даному підприємстві додаткова заробітна плата начисляється в розмірі 10% від основної заробітної плати.

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

$$З_д = 0,11 * 46\,365,5 = 5\,100,205 \text{ (грн)}$$

Нарахування на заробітну плату $Н_{ЗП}$ дослідників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою (5.3):

$$Н_{ЗП} = \left(З_о + З_д\right) * \frac{\beta}{100} \text{ (грн)} \quad (5.8)$$

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

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

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % .

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

$$Н_{ЗП} = (46\,365,5 + 5\,100,205) * \frac{22}{100} = 11\,322,4551 \text{ (грн)}$$

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

$$M = \sum_{1}^{n} H_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} B_i \cdot Ц_в \quad \text{грн.,} \qquad (5.9)$$

де H_i – витрати матеріалу *і*-го найменування, кг;

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

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

B_i – маса відходів матеріалу *і*-го найменування, кг;

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

n – кількість видів матеріалів.

Таблиця 5.6 – Матеріали, що використані на розробку

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено	Вартість витраченого матеріалу, грн.
Папір StoraEnso A4 Zoom 80	169	1	169
Канцелярське приладдя (ручки, олівці)	200	1	200
CD-диск	12	1	12
Флешка	155	1	155
Всього			536
З врахуванням коефіцієнта транспортування			589,6

5. Програмне забезпечення для наукової роботи включає витрати на розробку та придбання спеціальних програмних засобів і програмного

забезпечення необхідного для проведення дослідження. Для нової розробки використовувались безкоштовні програмні засоби. Операційна система MacOS, середовище розробки XCode та редактор коду Visual Studio Code, безкоштовний хостинг серверу Heroku. Для того, щоб реалізувати за запустити програму достатньо безкоштовних тарифних планів.

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

Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

$$A = \frac{Ц \cdot T}{Tкор \cdot 12} \quad [грн], \tag{5.10}$$

де Ц – балансова вартість даного виду обладнання (приміщень), грн.; $Т_{кор}$ – час користування; Т – термін використання обладнання (приміщень), цілі місяці.

Згідно пункту 137.3.3 Податкового кодексу амортизація нараховується на основні засоби вартістю понад 20000 грн. В нашому випадку для написання магістерської роботи використовувався ноутбук MacBook Air M1 вартістю 45000 грн..

$$\mathrm{A} = \frac{45000 \cdot 1}{2 \cdot 12} = 1875$$

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

$$\mathrm{В_е} = \sum_{\mathrm{i}=1}^{n} \frac{W_{yt} \bullet t_i \bullet \mathrm{Ц_е} \bullet \mathrm{К_{впі}}}{\eta_{\mathrm{i}}} \tag{5.11}$$

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

t_i – тривалість роботи обладнання на етапі дослідження, год;

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

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

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

Для написання магістерської роботи використовується персональний комп'ютер для якого розрахуємо витрати на електроенергію.

$$В_е = \frac{0{,}3 \cdot 250 \cdot 7{,}6 \cdot 0{,}5}{0{,}8} = 356{,}25$$

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

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

$$В_{нзв} = \left(З_о + З_р\right) \bullet \frac{Н_{нзв}}{100\%}, \qquad (5.12)$$

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

$$В_{нзв} = 46\,365,5 \cdot \frac{100}{100\%} = 46\,365,5 \text{ грн}$$

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

$$В_{заг} = 46\,365,5 + 5\,100,205 + 11\,322,4551 + 589,6 + 1875 + 356,25 + 46\,365,5 =$$
$$= 111\,974,5101 \text{ грн}$$

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

$$ЗВ = \frac{В_{заг}}{\eta}, \qquad (5.13)$$

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

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

Звідси:

$$ЗВ = \frac{111\,974,5101}{0,9} = 124\,416,122 \text{грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

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

$$\Delta П_i = (\pm\ \Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.14)$$

де $\pm\ \Delta Ц_о$ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році [29].

N – основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки; [29]

$Ц_о$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році, $Ц_о = Ц_б \pm\ \Delta Ц_о$; [29]

$Ц_б$ – основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів; [29]

ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році; [29]

λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$; [29]

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (послуги), $\rho = 0{,}25$; [29]

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор. У 2023 році – 18%. [29]

Для розглянутої системи, N є кількістю користувачів, ΔN - зміною кількості користувачів, $\pm\Delta Ц_о$ – зміна вартості після впровадження результатів розробки (взято 250 грн), $Ц_б$ - ціна реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів, приймемо як 750 грн; $Ц_о$ – вартість програми після впровадження науково-технічної розробки, відповідно становитиме 750 + 250 = 1000 грн.

Прогнозується, що протягом 3 років отримуються позитивні результати від впровадження розробки.

Передбачається, що ΔN у першому році дорівнює 100, другому 200, у третьому 300. N до реалізації становить 2500.

Розрахуємо прибуток, яке отримає підприємство протягом трьох років.

$$\Delta П_1 = (250 \cdot 2500 + (750 + 250) \cdot 100) \cdot 0,833 \cdot 0,25 \cdot (1 - 0.18) =$$

$$= 123804,62 \text{ грн.}$$

$$\Delta П_2 = (250 \cdot 2500 + (750 + 250) \cdot (100 + 200)) \cdot 0,833 \cdot 0,25 \cdot (1 - 0.18) =$$

$$= 192\,631,07 \text{ грн.}$$

$$\Delta П_3 = (250 \cdot 2500 + (750 + 250) \cdot (100 + 200 + 300)) \cdot 0,833 \cdot 0,25 \cdot (1 - 0.18) =$$

$$= 255\,106,07 \text{ грн.}$$

Приведена вартість збільшення всіх чистих прибутків від впровадження розробки обчислюється за формулою:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} \qquad (5.15)$$

де Т – роки, протягом яких очікується отримання позитивних результатів від впровадження та комерціалізації розробки, Δ – ставка дисконтування (0.15), t – період від впровадження розробки до отримання інвестором додаткових чистих прибутків у році.

$$ПП = 123804,62/(1 + 0.15) + 192\,631,07/(1 + 0.15) * 2 +$$

$$+ 255\,106,07/(1 + 0.15) * 3 = 1\,108\,160,85 \text{ грн}$$

Початкові інвестиції для вкладення інвестором обчислюються за формулою

$$PV = k_{\text{розр}} * \text{ЗВ} \quad (5.16)$$

де $K_{\text{розр}}$k розр. - коефіцієнт витрат на впровадження розробки.

Обчислимо значення початкових інвестицій для вкладання інвестором

$$PV = 2 * 124\,416,122 = 248\,832,244 \text{ грн.}$$

Абсолютний економічний ефект від впровадження розробки є чистим приведеним доходом, обчислюється за формулою

$$\text{Е}_{\text{абс}} = \text{ПП} - PV \quad (5.17)$$

В результаті абсолютний економічний ефект від впровадження розробки

$$\text{Е}_{\text{абс}} = 1\,108\,160,85 - 248\,832,244 = 859\,328,606 \text{ грн}$$

Отже, можлива доцільність впровадження розробки.

Внутрішня економічна дохідність допомагає остаточно визначити, чи доцільно впроваджувати розробку, розраховується за формулою

$$\text{Е}_{\text{в}} = \sqrt[\text{Т}_{\text{ж}}]{1 + \frac{\text{Е}_{\text{абс}}}{PV}} - 1 \quad (5.18)$$

де $\text{Т}_{\text{ж}}$ – життєвий цикл розробки.

Розрахуємо $E_в$

$$E_в = \sqrt[3]{1 + \frac{859\,328{,}606}{248\,832{,}244}} - 1 = 0{,}65 = 65\%$$

Мінімальна внутрішня економічна дохідність інвестицій r мін показує, чи буде інвестор зацікавлений в інвестиціях, обчислюється за формулою

$$\tau_{мін} = d + f \qquad (5.19)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках (обрано 0.1), f – ризикованість вкладення інвестицій (обрано 0.3).

$$\tau_{мін} = 0.1 + 0.3 = 0.4 = 40\%$$

Внутрішня економічна дохідність більше ніж мінімальна внутрішня економічна дохідність, тому інвестори повинні бути зацікавлені у фінансуванні розробки.

Період окупності обчислюється за формулою

$$T_{ок} = \frac{1}{E_в} \qquad (5.18)$$

Отримуємо значення $1 / 0{,}65 = 1.5$ роки. Отриманий період окупності менший за 3 роки, тому наукова розробка за темою “Метод і програмний засіб рекомендацій рекреаційних зон” є комерційно привабливою.

5.5 Висновки до економічного розділу

Було проведено оцінку комерційного потенціалу методу і програмного засобу рекомендацій рекреаційних зон, який є на вище середньому рівні. При порівнянні нової розробки з аналогом виявлено, що вона є якіснішою і конкурентоспроможнішою відносно аналога, а також краще по технічним і економічним показникам.

Прогнозування витрат на виконання науково-дослідної роботи по кожній з статей витрат складе 111 974,5101 грн. Загальна ж величина витрат на виконання та впровадження результатів даної НДР буде складати 124 416,122 грн.

Вкладені інвестиції в даний проект окупляться через 1.5 роки при прогнозованому прибутку 1 108 160, 85 грн. за три роки.

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи було проведено аналіз існуючих рекомендаційних систем та визначено особливості їхньої роботи. Було розглянуто методи фільтрації, які застосовуються в рекомендаційних системах, та наведено приклади їх використання. Також було досліджено рекреаційні зони міста та розроблено критерії для формування рекомендацій.

У процесі роботи було розроблено алгоритмічне забезпечення системи, що включає модулі прийому та відправки повідомлень, оновлення датасету, навчання моделі. Розроблено метод прогнозування рекомендацій на основі критеріїв від користувача. Для реалізації алгоритмів та методу прогнозування використовувалися мови програмування Python та Swift, остання з яких застосовувалася для створення клієнтського додатку..

Експериментально було досліджено процес формування рекомендацій рекреаційних зон міста. Встановлено, що оптимальна конфігурація системи, яка забезпечує найкращі результати за кількістю та якістю рекомендацій, досягається за допомогою використання стандартного TF-IDF в поєднанні з інструментами бібліотеки sklearn. Це дозволяє відкидати несуттєві для алгоритму слова, підвищуючи значимість ключових термінів, що позитивно впливає на якість рекомендацій.

Таким чином удосконалено метод прогнозування рекомендацій у якому, на відміну від існуючих, береться до уваги перетин різних критеріїв для формування більш точних рекомендацій.

Робота оформлена згідно з методичними вказівками та супроводжується інструкцією для користувачів [30].

Тестування програми довело повну працездатність даного програмного продукту та відповідність поставленому технічному завданню.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Recommender system [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Recommender_system (режим доступу до ресурсу 10.07.2023) – Заголовок з екрану.

2. What is a Recommendation System? [Електронний ресурс]. Режим доступу: https://www.nvidia.com/en-us/glossary/data-science/recommendation-system/ (режим доступу до ресурсу 10.07.2023) – Заголовок з екрану.

3. Recommender Systems — A Complete Guide to Machine Learning Models [Електронний ресурс]. Режим доступу: https://towardsdatascience.com/recommender-systems-a-complete-guide-to-machine-learning-models-96d3f94ea748 (режим доступу до ресурсу 10.07.2023) – Заголовок з екрану.

4. What are Recommendation Systems? [Електронний ресурс]. Режим доступу: https://medium.com/@khang.pham.exxact/what-are-recommendation-systems-6bb5036042db (режим доступу до ресурсу 10.07.2023) – Заголовок з екрану.

5. Recommender Systems: An Introduction [Електронний ресурс]. Режим доступу: http://pzs.dstu.dp.ua/DataMining/recom/bibl/1jannach_dietmar_zanker_markus_felfernig_alexander_friedrich.pdf (режим доступу до ресурсу 10.07.2023) – Заголовок з екрану.

6. Recommendation System - Understanding The Basic Concepts [Електронний ресурс]. Режим доступу: https://www.analyticsvidhya.com/blog/2021/07/recommendation-system-understanding-the-basic-concepts/ (режим доступу до ресурсу 10.07.2023) – Заголовок з екрану.

7. A Systematic Review and Research Perspective on Recommender Systems [Електронний ресурс]. Режим доступу: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00592-5 (режим доступу до ресурсу 10.07.2023) – Заголовок з екрану.

8. Introduction to Recommender Systems [Електронний ресурс]. Режим доступу: https://thingsolver.com/blog/introduction-to-recommender-systems/ (режим доступу до ресурсу 10.07.2023) – Заголовок з екрану.

9. Recommender Systems [Електронний ресурс]. Режим доступу: https://link.springer.com/10.1007%2F978-0-387-30164-8_705 (режим доступу до ресурсу 10.07.2023) – Заголовок з екрану.

10. Recommender System Using Machine Learning [Електронний ресурс]. Режим доступу: https://maddevs.io/blog/recommender-system-using-machine-learning/ (режим доступу до ресурсу 10.07.2023) – Заголовок з екрану.

11. Last.fm [Електронний ресурс]. Режим доступу: https://ru.wikipedia.org/wiki/Last.fm (режим доступу до ресурсу 10.07.2023) – Заголовок з екрану.

12.Pandora [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Pandora_(streaming_service) (режим доступу до ресурсу 12.07.2023) – Заголовок з екрану.

13.Uber [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Uber (режим доступу до ресурсу 15.07.2023) – Заголовок з екрану.

14.Lyft [Електронний ресурс]. Режим доступу: https://ru.wikipedia.org/wiki/Lyft (режим доступу до ресурсу 15.07.2023) – Заголовок з екрану.

15.Netflix [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Netflix (режим доступу до ресурсу 22.07.2023) – Заголовок з екрану.

16.TF-IDF [Електронний ресурс]. Режим доступу: https://ru.wikipedia.org/wiki/TF-IDF (режим доступу до ресурсу 11.08.2023) – Заголовок з екрану.

17. С.Є. Тужанський, І.О. Яровий. РЕКОМЕНДАЦІЙНА СИСТЕМА ДЛЯ ВИБОРУ РЕКРЕАЦІЙНИХ ЗОН. Міжнародна науково-практична

інтернет-конференція «Електронні інформаційні ресурси: створення, використання, доступ», Вінниця, 20-21 листопада 2023 р. – С. 323 – 325.

18.Векторна модель [Електронний ресурс]. Режим доступу: https://ru.wikipedia.org/wiki/Векторная_модель (режим доступу до ресурсу 11.08.2023) – Заголовок з екрану.

19.Flask [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Flask (режим доступу до ресурсу 02.09.2023) – Заголовок з екрану.

20. Build a Recommendation Engine With Collaborative Filtering [Електронний ресурс]. Режим доступу: https://realpython.com/build-recommendation-engine-collaborative-filtering/ (режим доступу до ресурсу 02.09.2023) – Заголовок з екрану.

21. TF-IDF з прикладами коду [Електронний ресурс]. Режим доступу: http://nlpx.net/archives/57 (режим доступу до ресурсу 05.09.2023) – Заголовок з екрану.

22. Machine Learning with Python [Електронний ресурс]. Режим доступу: http://aimotion.blogspot.com/2011/12/machine-learning-with-python-meeting-tf.html (режим доступу до ресурсу 07.09.2023) – Заголовок з екрану.

23. Machine Learning; Text feature extraction [Електронний ресурс]. Режим доступу: https://blog.christianperone.com/2011/09/machine-learning-text-feature-extraction-tf-idf-part-i/ (режим доступу до ресурсу 15.09.2023) – Заголовок з екрану.

24. Finding Important Words in Text Using TF-IDF [Електронний ресурс]. Режим доступу: https://stevenloria.com/tf-idf/ (режим доступу до ресурсу 16.09.2023) – Заголовок з екрану.

25. The TF*IDF Algorithm Explained [Електронний ресурс]. Режим доступу: https://www.onely.com/blog/what-is-tf-idf/ (режим доступу до ресурсу 18.09.2023) – Заголовок з екрану.

26. Recommender Systems with Python – Part I: Content-Based Filtering [Електронний ресурс]. Режим доступу: https://heartbeat.fritz.ai/recommender-systems-with-python-part-i-content-based-filtering-5df4940bd831 (режим доступу до ресурсу 21.09.2023) – Заголовок з екрану.

27. How to Build a Content-Based Recommender System For Your Product [Електронний ресурс]. Режим доступу: https://www.offerzen.com/blog/how-to-build-a-content-based-recommender-system-for-your-product (режим доступу до ресурсу 24.09.2023) – Заголовок з екрану.

28. Philip, Simon, Peter Shola, and A. Ovye. "Application of content-based approach in research paper recommendation system for a digital library." International Journal of Advanced Computer Science and Applications 5.10 (2014).

29. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. 42 с

30. Положення про кваліфікаційну роботу на другому (вищому) рівні вищої освіти. Уклад / А.О. Семенов - Вінниця : ВНТУ, 2021. 60 с. СУЯ ВНТУ-03.02.02-П.001.01:21

ДОДАТКИ

107

Додаток А

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

"19 " вересня 2023 р.

Технічне завдання

на магістерську кваліфікаційну роботу «Метод і програмний засіб рекомендацій рекреаційних зон» за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

к.т.н., асистент. каф. ПЗ Тужанський С.Є.

"19" вересня 2023 р.

Виконав:

студент гр.1ПІ-22М І.О. Яровий

"19" вересня 2023 р.

Вінниця – 2023 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Метод і програмний засіб рекомендацій рекреаційних зон».

Галузь застосування - автоматизовані системи обробки даних.

2. Підстава для розробки.

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

3. Мета та призначення розробки.

Метою роботи є підвищення зручності та швидкості пошуку рекреаційних зон користувачам на основі їх критеріїв (вподобань), що досягається за рахунок сформованого персоналізованого списку рекомендацій.

Призначення роботи – підвищення ефективності пошуку зон для відпочинку.

3 Вихідні дані для проведення НДР

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

1. Philip, Simon, Peter Shola, and A. Ovye. "Application of content-based approach in research paper recommendation system for a digital library." International Journal of Advanced Computer Science and Applications 5.10 (2014).
2. Wu, Ching-Seh Mike, Deepti Garg, and Unnathi Bhandary. "Movie recommendation system using collaborative filtering." 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS). IEEE, 2018.

3. Tsolakidis, Anastasios, et al. "Research publication recommendation system based on a hybrid approach." Proceedings of the 20th Pan-Hellenic conference on informatics. 2016.
4. Sunandana, G., et al. "Movie recommendation system using enhanced content-based filtering algorithm based on user demographic data." 2021 6th International Conference on Communication and Electronics Systems (ICCES). IEEE, 2021.

4. Технічні вимоги

Вхідні дані - критерії (вподобання) щодо рекреаційних зон українською мовою; вихідні дані - список рекомендованих рекреаційних з коротким описом, відповідно до вказаних критерії.

5. Конструктивні вимоги.

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

Графічна та текстова документація повинна відповідати діючим стандартам України.

6. Перелік технічної документації, що пред’являється по закінченню робіт:

- пояснювальна записка до МКР;
- технічне завдання;
- лістинги програми.

7. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

8. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Аналіз систем рекомендацій	20.09.2023-09.10.2023
2	Розробка методу навчання моделі	10.10.2023-29.10.2023
3	Розробка методу рекомендацій	30.10.2023-14.10.2023
4	Розробка клієнтської частини системи	15.11.2023-24.11.2023
5	Економічна частина	25.11.2023-05.12.2023

9. Порядок контролю та прийняття.

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

111

Додаток Б

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: **Метод і програмний засіб рекомендацій рекреаційних зон.**

Тип роботи: кваліфікаційна робота
Підрозділ : кафедра програмного забезпечення, ФІТКІ, 1ПІ-22м
Науковий керівник: к.т.н., доц. каф. ПЗ Тужанський С.Є.

Unicheck	
Оригінальність	**97,7 %**
Схожість	2,3 %

Аналіз звіту подібності

■**Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений з повним звітом подібності, який був згенерований Системою щодо роботи «Метод і програмний засіб рекомендацій рекреаційних зон".

Особа, відповідальна за перевірку _____ Черноволик Г. О

Опис прийнятого рішення: **допустити до захисту**

Автор роботи _____ Яровий І. О.

Керівник роботи _____ Тужанський С.Є.

Додаток В

Лістинг коду модуля прийому та відправки повідомлень

```
from Database import database_service
from config import app
from flask import request
from flask import redirect
from flask import jsonify
from flask import make_response


class CriteriaAPI:
    @app.route("/criterias", methods = ['GET'])
    def getCriterias():
        criterias = database_service.DBService.criterias()
        return make_response(jsonify({ "data": criterias }), 200)


class RecommendationAPI:
    @app.route("/recommendation/<id>", methods = ['GET'])
    def getRecommendation(id):
        criteria = database_service.DBService.criteriaBy(id)
                                                        recommendations       =
recommendation.Recommendation.getRecommendations(criteria.title)
        return make_response(jsonify({ "data": list(recommendations) }), 200)

    @app.route("/recommendations", methods = ['POST'])
    def getRecommendations():
        data = request.get_json(force=True)
        result = list()

        for id in data['id']:
            criteria = database_service.DBService.criteriaBy(id)
                                                          recommendations     =
recommendation.Recommendation.getRecommendations(criteria.title)
```

```
    result.append(list(recommendations))

return make_response(jsonify({ "data": list(result) }), 200)
```

Додаток Г

Лістинг коду модуля оновлення датасету

```
import os
import pandas as pd

class Store:
    @staticmethod
    def addOrUpdate(entity):
        currentDir = os.path.dirname(os.path.abspath(__file__))
        currentDir = os.path.join(currentDir, "dataset.csv")
        pd.write(currentDir, entity)

    @staticmethod
    def remove(id):
        currentDir = os.path.dirname(os.path.abspath(__file__))
        currentDir = os.path.join(currentDir, "dataset.csv")
        pd.removeBy(id, currentDir)
```

Додаток Д

Лістинг коду модуля навчання моделі

```
import os
import pandas as pd
```

```
class ModelLearning:


    @staticmethod
    def learn():
        currentDir = os.path.dirname(os.path.abspath(__file__))
        currentDir = os.path.join(currentDir, "dataset.csv")
        ds = pd.read_csv(currentDir)


        tf = TfidfVectorizer(analyzer='word', ngram_range=(1, 3), min_df=0,
stop_words=get_stop_words('ukrainian'))
        tfidf_matrix = tf.fit_transform(ds['description'])
        cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)
        mapping = pd.Series(ds.index, index = ds['criteria'])
        return mapping
```

Додаток Е

Лістинг коду модуля прогнозування рекомендацій

```
class Recommendation:


    @staticmethod
    def getRecommendations(criteria):
        placeIndex = ModelLearning.learn()
        for index in placeIndex:
            similarityScore = list(enumerate(cosine_similarities[index]))
              similarityScore = sorted(similarityScore, key=lambda x: x[1],
reverse=True)
            similarityScore = similarityScore[:5]
            places += [similarityScore[0][0]]


        return ds['description'].iloc[places]
```

Додаток Ж

Лістинг коду відправки запиту до серверу для отримання списку рекомендацій

```
final class RecommendationService: RecommendationUseCase {
    let network: Network

    init(network: Network) {
        self.network = network
    }

    func criterias() -> AsyncTask<[Criteria]> {
        network.reactive
        .request(API.Criterias.getCriterias)
        .decode(APIResponse<[Criteria.Response]>.self)
        .map { $0.data.map { Criteria(from: $0) }}
        .observe(on: UIScheduler())
    }
     // ...
}
```

Додаток И

Лістинг коду відправки запиту до серверу для отримання списку рекомендацій

```
final class RecommendationService: RecommendationUseCase {
     // ...
      func recommendations(by criteria: Criteria) ->
AsyncTask<Recommendations> {
              network.reactive
                 .request(API.Recommendation.getRecommendations(criteriaID:
criteria.id))
```

```
                .decode(APIResponse<[String]>.self)
                .map { Recommendations(data: $0.data) }
                .observe(on: UIScheduler())
            }

                func recommendations(for criterias: [Criteria]) ->
AsyncTask<[[String]]> {
                network.reactive
                        .request(API.Recommendation.recommendations(params:
["id": criterias.map { $0.id }]))
                .decode(APIResponse<[[String]]>.self)
                .map(\.data)
                .observe(on: UIScheduler())
            }
        }
```

Додаток І

Лістинг коду контролера представлення

```
 final class ChooseCriteriaVC: BaseVC, ViewModelContainer {

    @IBOutlet private(set) weak var proceedButton: UIButton!
    @IBOutlet private(set) weak var tableView: UITableView!

    @IBAction private func makeRecommendations(_ sender: UIButton) {
        viewModel.fetchRecommendations()
    }

    func didSetViewModel(_ viewModel: ChooseCriteriaVM, lifetime: Lifetime)
{
        reactive.activity <~ viewModel.actions.isExecuting
        reactive.errors <~ viewModel.actions.errors
        reactive.makeBindingTarget { viewController, criterias in
            viewController.viewModel.updateDatasource(with: criterias)
```

```
            } <~ viewModel.criterias.signal.take(during: lifetime)
                                                tableView.reactive.reloadData     <~
viewModel.datasource.signal.mapToVoid()
                                            proceedButton.reactive.isEnabled     <~
viewModel.selectedCriteria.producer.map { !$0.isEmpty }

            reactive.makeBindingTarget { viewController, _ in
                viewController.viewModel.showListOfRecommendations()
            } <~ viewModel.recommendations.signal.take(during: lifetime)
        }

        override func viewDidLoad() {
            super.viewDidLoad()
            setup()
            viewModel.fetchCriterias()
        }

        private func setup() {
            title = viewModel.title
            setupTableView()
            setupProceedButton()
        }

        private func setupProceedButton() {
            proceedButton.layer.cornerRadius = proceedButton.bounds.height / 4

proceedButton.setTitle(R.string.localizable.chooseCriteriaButtonProceedTitle(),
for: .normal)
        }

        private func setupTableView() {
            tableView.registerNib(for: CriteriaTVC.self)
            let headerView = TableHeader(frame: .zero)
            headerView.headerDescription = viewModel.description
            tableView.tableHeaderView = headerView
```

```
                tableView.tableHeaderView?.frame.size = CGSize(width:
tableView.bounds.width,
                                                                height:
tableView.bounds.height * 0.5)
                tableView.dataSource = self
            }
        }

        extension ChooseCriteriaVC: UITableViewDataSource {
            func tableView(_ tableView: UITableView, numberOfRowsInSection section:
Int) -> Int {
                viewModel.datasource.value.count
            }

            func tableView(_ tableView: UITableView, cellForRowAt indexPath:
IndexPath) -> UITableViewCell {
                let cell = tableView.dequeueReusableCell(cellClass:
CriteriaTVC.self, for: indexPath)
                cell.configure(with: viewModel.datasource.value[indexPath.row])
                return cell
            }
        }
```

Додаток Й

Лістинг коду моделі представлення

```
        protocol ChooseCriteriaVMDelegate: class {
                func chooseCriteriaVM(_ viewModel: ChooseCriteriaVM,
didGetRecommendations recommendations: [[String]])
        }

        final class ChooseCriteriaVM: UseCasesConsumer {
            typealias UseCases = HasRecommendationUseCase
```

```
            let actions = ActionGroup()
            let criterias = MutableProperty<[Criteria]>([])
            var title: String { R.string.localizable.chooseCriteriaTitle() }
                                        var       description:       String       {
R.string.localizable.chooseCriteriaDescription() }
                                    private(set)       var       datasource       =
MutableProperty<[UserInterestView.Model]>([])
            private(set) var selectedCriteria = MutableProperty<[Criteria]>([])
            private(set) var recommendations = MutableProperty<[[String]]>([])
               private(set)  lazy  var  fetchRecommendationAction  =  Action(execute:
useCases.recommendation.recommendations(for:))
                   private   lazy   var   fetchCriteriasAction   =   Action(execute:
useCases.recommendation.criterias)
            private weak var delegate: ChooseCriteriaVMDelegate?

            init(useCases: UseCases, delegate: ChooseCriteriaVMDelegate) {
                self.useCases = useCases
                self.delegate = delegate
                setupObservers()
            }

            func fetchCriterias() {
                fetchCriteriasAction.apply().start()
            }

            func fetchRecommendations() {
                fetchRecommendationAction.apply(selectedCriteria.value).start()
            }

            func updateDatasource(with criterias: [Criteria]) {
                criterias.forEach {
                    let model = UserInterestView.Model(
                        criteria: $0,
                        tapHandler: { [weak self] criteria, isOn in
```

```
                        isOn ? self?.addCriteria(criteria)
                             : self?.removeCriteria(criteria)
                    })
                datasource.value.append(model)
            }
        }

        func showListOfRecommendations() {
                    delegate?.chooseCriteriaVM(self,  didGetRecommendations:
recommendations.value)
        }

        private func addCriteria(_ criteria: Criteria) {
            selectedCriteria.value.append(criteria)
        }

        private func removeCriteria(_ criteria: Criteria) {
             selectedCriteria.value = selectedCriteria.value.filter { $0.id !=
criteria.id }
        }

        private func setupObservers() {
            actions.append(fetchCriteriasAction)
            actions.append(fetchRecommendationAction)
               criterias <~ fetchCriteriasAction.values.take(duringLifetimeOf:
self)
                                                    recommendations      <~
fetchRecommendationAction.values.take(duringLifetimeOf: self)
        }
    }
```

Додаток К

ІЛЮСТРАТИВНА ЧАСТИНА

Метод і програмний засіб рекомендацій рекреаційних зон

Рисунок Г.1 - Слайд презентації №1

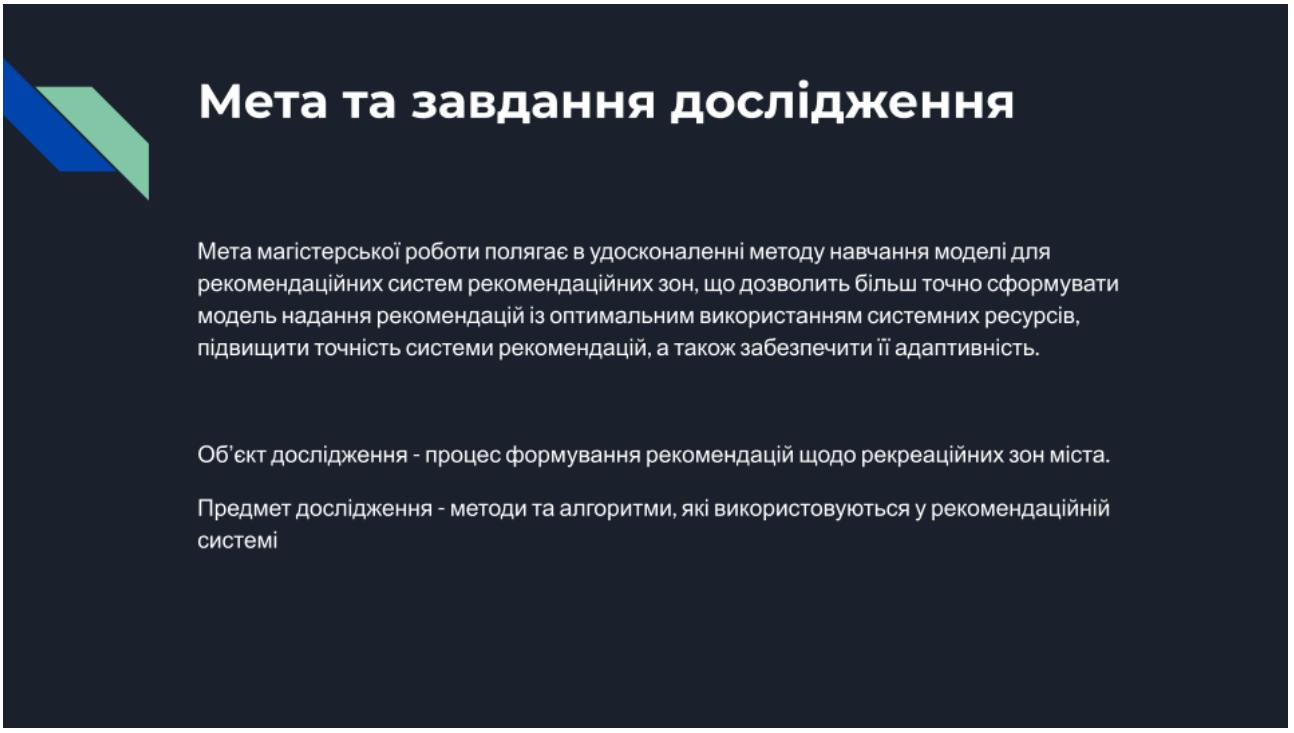

Рисунок Г.2 - Слайд презентації №2

Мета та завдання дослідження

Основними завданнями дослідження є:

- провести аналіз існуючих методів і засобів рекомендаційних систем для визначення напрямків підвищення їх продуктивності та точності;
- удосконалити метод прогнозування рекомендацій та метод навчання моделі;
- визначити перелік критеріїв для формування рекомендацій;
- розробити веб-сервіс для прогнозування рекомендацій;
- розробити мобільний клієнт для роботи з модулем прогнозування рекомендацій;
- провести тестування програмного додатку та експериментальні дослідження розробленого програмного засобу.

Рисунок Г.3 - Слайд презентації №3

Наукова новизна

Наукова новизна отриманих результатів:

- Удосконалено метод навчання моделі для рекомендаційних систем шляхом впровадження обчислення косинусної схожості векторів ознак рекреаційних зон після аналізу частотності термінів (TF) та зворотної частотності документів (IDF). Це підвищило точність моделі рекомендацій при ефективному використанні системних ресурсів.
- Отримав подальшого розвитку метод прогнозування рекомендацій за допомогою попередньої ініціалізації удосконаленої моделі навчання із зазначенням критеріїв аналізу та формування рекомендацій. Це забезпечило вищий рівень персоналізації рекомендацій, підвищило точність системи та покращило її адаптивність та інтеграційні можливості.

Рисунок Г.4 - Слайд презентації №4

Практична цінність отриманих результатів

Практичне значення магістерської кваліфікаційної роботи полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби для отримання рекомендацій щодо вибору рекреаційних зон.

Рисунок Г.5 - Слайд презентації №5

Апробації результатів та публікації

- Результати магістерської кваліфікаційної роботи доповідалися та обговорювалися на міжнародній науково-практичній інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ» (Вінниця, 2023).
- Основні результати дослідженні опубліковані у тезах доповіді на міжнародній науково-практичній інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ».

Рисунок Г.6 - Слайд презентації №6

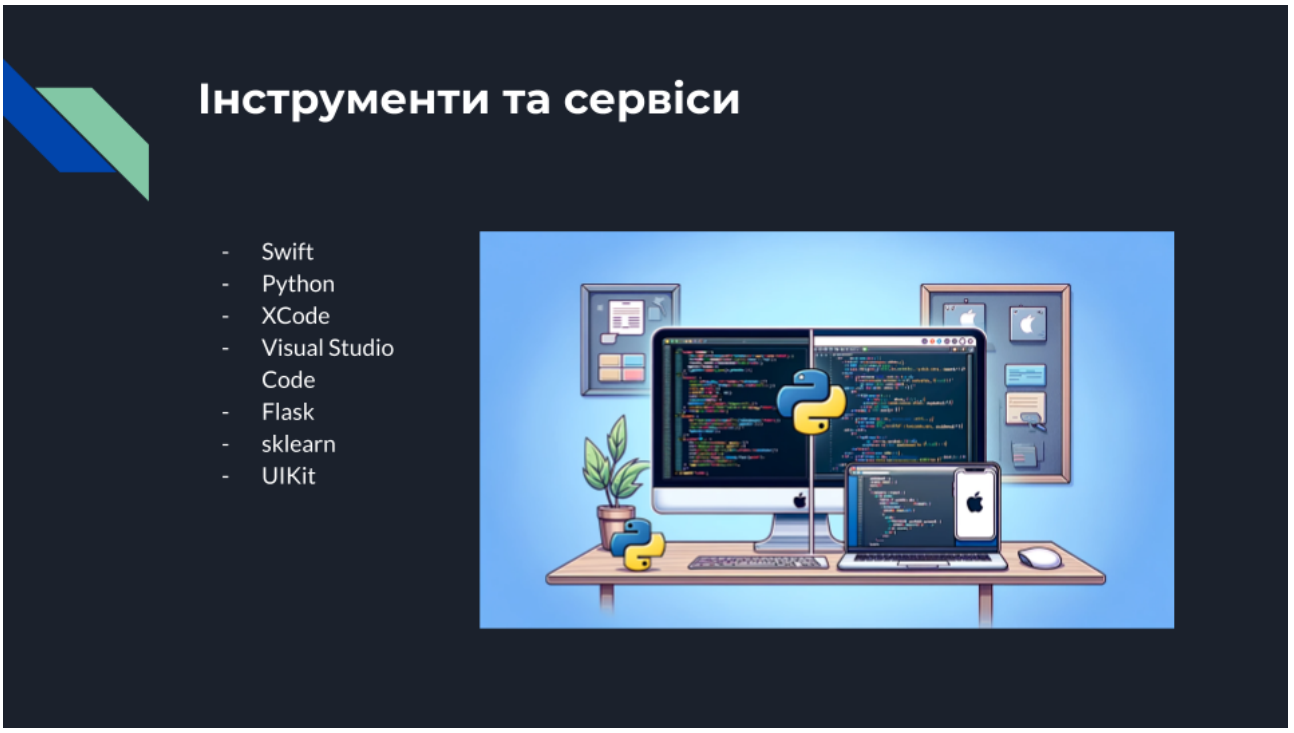

Рисунок Г.7 - Слайд презентації №7

Рисунок Г.8 - Слайд презентації №8

Перелік критеріїв для формування рекомендацій

Пасивний Активний	Бесідки	Медитація	Йога
Футбол	Відпочинок на свіжому повітрі	Відпочинок на пляжі	Риболовля
Велоспорт	Баскетбол	Відпочинок на свіжому повітрі	Настільні ігри
Прогулянки	Волейбол	Плавання	Відпочинок на свіжому повітрі

Рисунок Г.9 - Слайд презентації №9

Метод навчання моделі

- Описує процес навчання моделі, що впливає на точність рекомендаційних систем.
- Процес Навчання включає підготовку даних, вибір алгоритму, налаштування моделі, тренування та оцінку ефективності.
- Модель адаптується до нових даних та трендів для забезпечення актуальності рекомендацій.
- Аналізує частоту та унікальність термінів для визначення ключових елементів тексту.
- Розрахунок TF-IDF - інтеграція частотності терміну TF та зворотної частотності документів IDF для оцінки важливості термінів.
- Косинусна Схожість - визначає схожість між рекреаційними зонами для точних рекомендацій.
- Використання обчислень TF-IDF для релевантності зон та оптимізація ресурсів на завершальній стадії.

Рисунок Г.10 - Слайд презентації №10

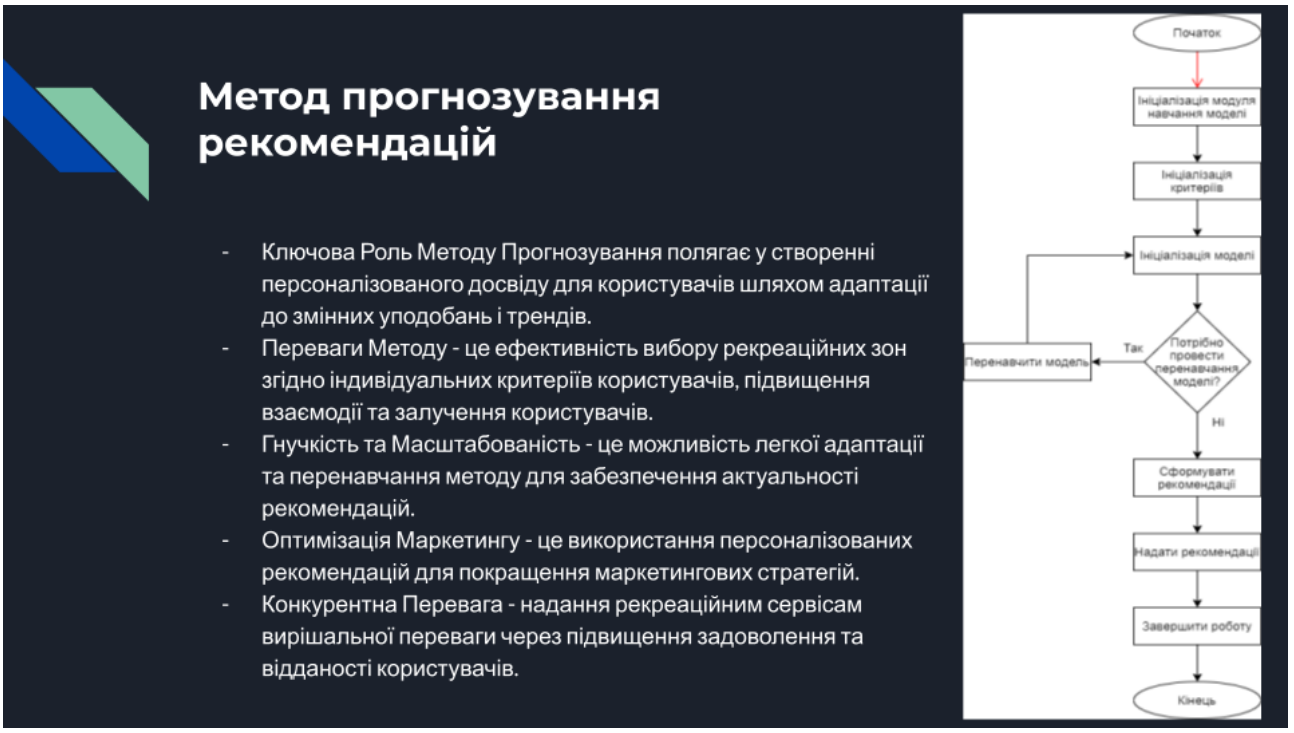

Рисунок Г.11 - Слайд презентації №11

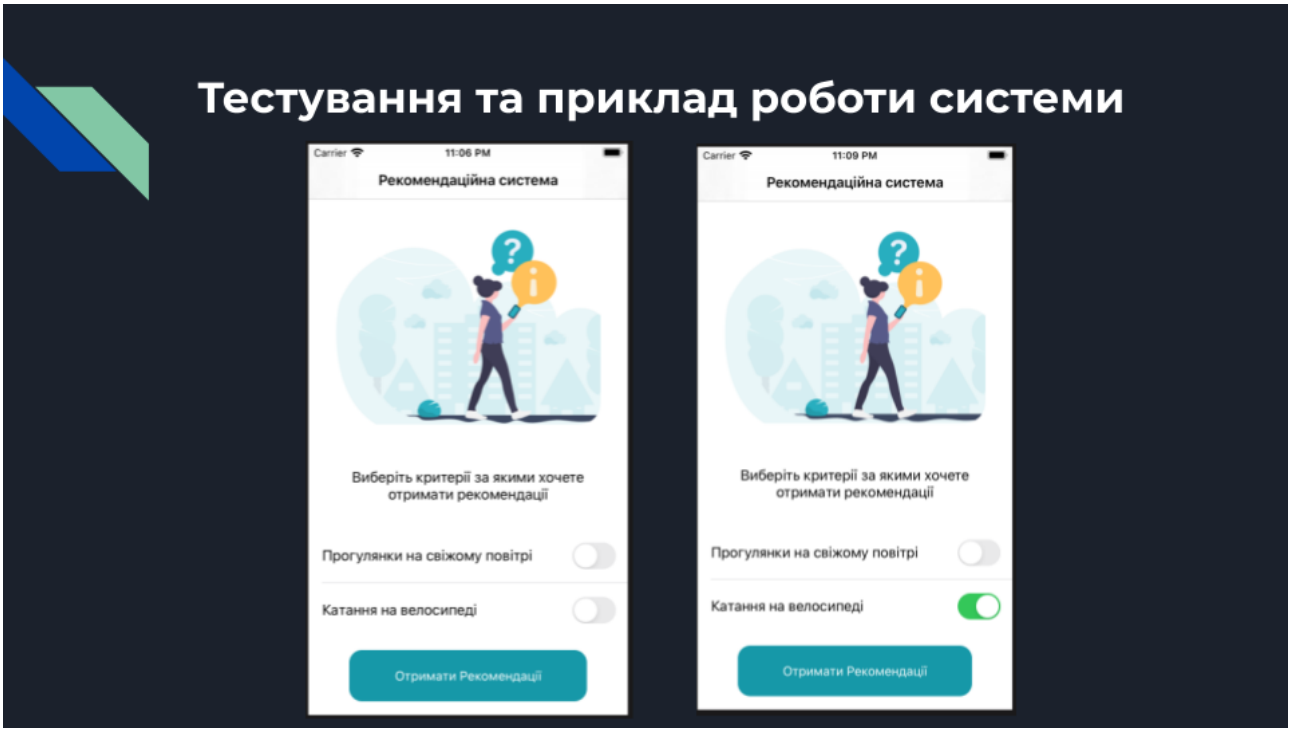

Рисунок Г.12 - Слайд презентації №12

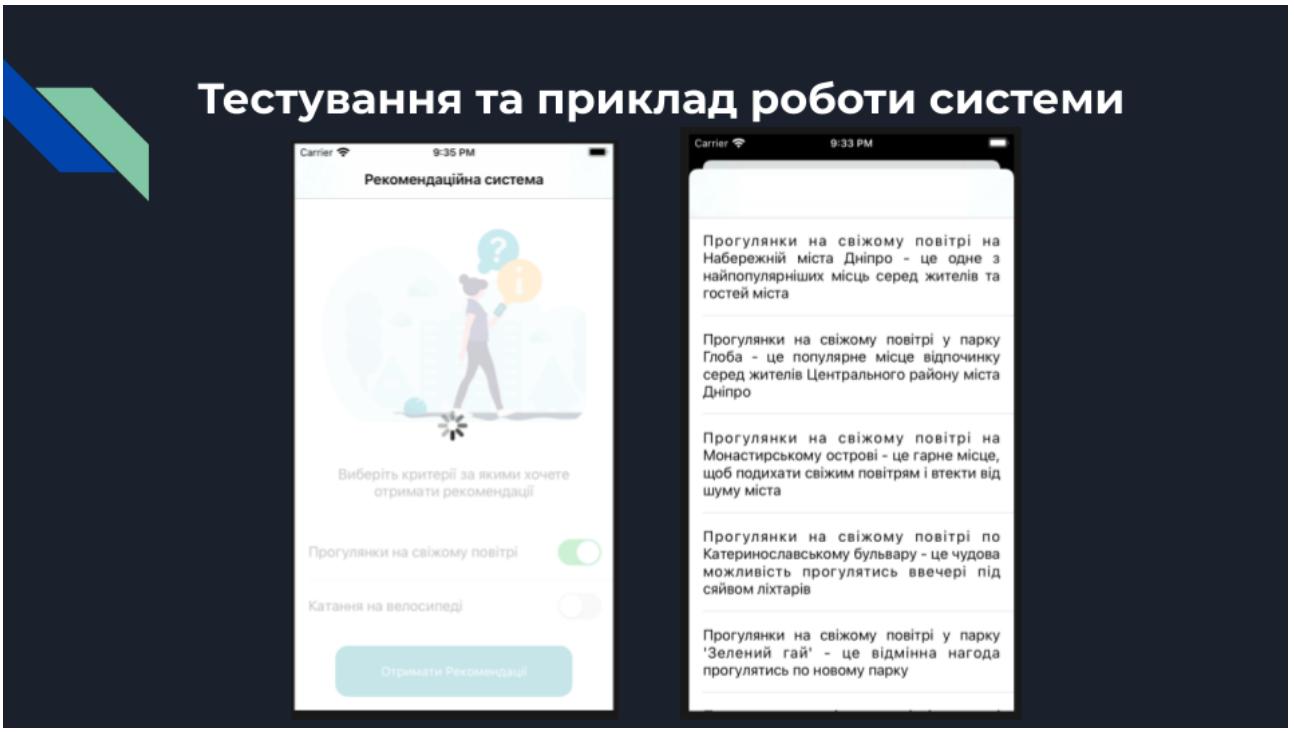

Рисунок Г.13 - Слайд презентації №13

Рисунок Г.14 - Слайд презентації №14

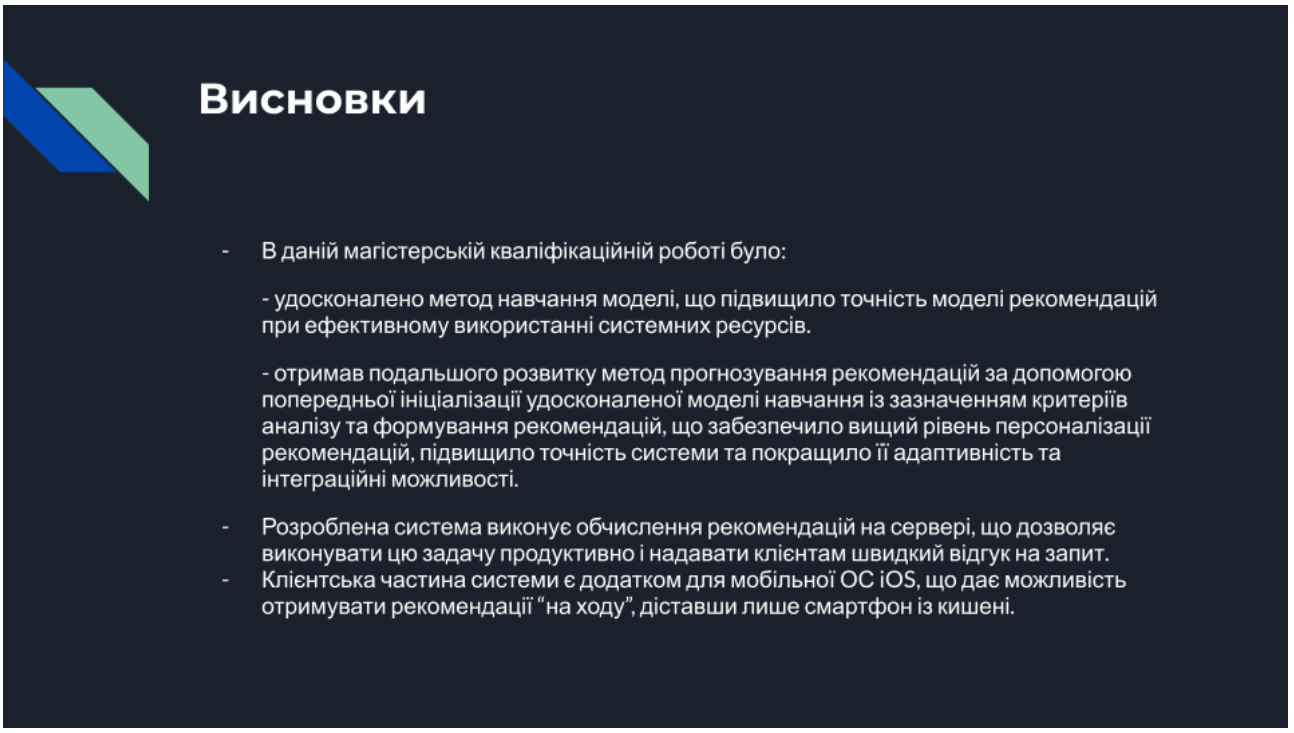

Рисунок Г.15 - Слайд презентації №15