

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ПІДБОРУ ЛІТЕРАТУРИ
RESEARCH AND SOFTWARE IMPLEMENTATION OF THE
RECOMMENDATION SYSTEM FOR LITERATURE SELECTION

Виконав: студент(ка) 2 курсу, групи КНТ-111м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Сопін В.С.

(прізвище та ініціали)

Керівник Колпакова Т.О.

(прізвище та ініціали)

Рецензент Коцур М.І.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Сопіна Василя Сергійовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація рекомендаційної системи підбору літератури. Research and Software Implementation of the Recommendation System for Literature Selection

керівник проєкту (роботи) Колпакова Тетяна Олексіївна, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “04” листопада 2022 року № 363

2. Строк подання студентом проєкту (роботи) 01 грудня 2022 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Вибір та обґрунтування матеріалів та методів розробки. 3. Розробка архітектури програми. 4. Розробка програми. 5. Експлуатація, тестування програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Колпакова Т.О., доцент		
Нормоконтроль	Липовець М.В., асистент		

7. Дата видачі завдання “01” вересня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз проблеми та постановка завдань дослідження.	2-3 тижні	Розділ 1
3	Вибір та обґрунтування матеріалів та методів розробки.	4-5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7-8 тижні	Розділ 4
6	Експлуатація, тестування програми.	9 тиждень	Розділ 5
7	Оформлення пояснювальної записки та документів до неї.	10-13 тижні	Додатки
8	Нормоконтроль та рецензування.	14-15 тижні	
9	Захист роботи.	16 тиждень	

Студент(ка)

(підпис) Сопін В.С.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Колпакова Т.О.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра: 96 с., 48 рис., 1 табл., 3 дод., 23 джерел.

РЕКОМЕНДАЦІЙНА СИСТЕМА, КОМП'ЮТЕРНА ПРОГРАМА, PYTHON, ФРЕЙМВОРК, DJANGO.

Об'єкт дослідження – рекомендаційні системи.

Предмет дослідження – вебсервіс рекомендаційної системи підбору літератури.

Мета роботи – дослідити та програмно реалізувати рекомендаційну систему підбору літератури.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мова програмування Python, мова тегів HTML, фреймворк Django, база даних Sqlite, персональний комп'ютер з процесором Intel Core i5, під управлінням операційної системи Microsoft Windows 10.

Результати. Створено рекомендаційної системи підбору літератури, що може надавати користувачам рекомендації кількома способами. Рекомендаційну систему було реалізовано на базі популярного фреймворку Django.

Висновки. Розроблено рекомендаційну систему, що дає можливість користувачу отримати рекомендації, на основі різних методів фільтрації.

Галузь використання – електронна комерція комерційних фірм та підприємців.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 96 pages, 48 figures, 1 tables, 3 appendixes, 23 sources.

RECOMMENDATION SYSTEM, COMPUTER PROGRAM, PYTHON, FRAMEWORK, DJANGO.

Object of research is recommendation systems.

The subject of the research is a web service of a literature selection recommendation system.

The purpose of the work is to research and programmatically implement a recommendation system for literature selection.

Materials, methods and technical means: object-oriented programming, Python programming language, Html tag language, Django framework, Sqlite database, personal computer with an Intel Core i5 processor, running the Microsoft Windows 10 operating system.

The results. A literature selection recommendation system was created, which can provide users with recommendations in several ways. The recommendation system was implemented on the basis of the popular Django framework.

Conclusions. A recommendation system has been developed that allows the user to receive recommendations based on various filtering methods.

The field of use is e-commerce of commercial firms and entrepreneurs.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	8
Вступ	9
1 Аналіз проблеми та постановка завдань дослідження	10
1.1 Дослідження характеристик рекомендаційних систем	10
1.2 Аналіз існуючих підходів щодо надання рекомендацій	14
1.3 Особливості предметної області	20
1.4 Огляд програмних аналогів	23
1.5 Постановка задачі	30
1.6 Висновки за розділом	30
2 Вибір та обґрунтування матеріалів та методів розробки	31
2.1 Порівняння методів формування онлайн-рекомендацій	31
2.2 Метрики якості наданої рекомендації	37
2.2.1 Precision at K ($p@K$)	37
2.2.2 Average precision at K	37
2.2.3 Normalized discounted cumulative gain (nDCG)	38
2.3 Вибір матеріалів та методів розробки	39
2.4 Висновки за розділом 2	39
3 Розробка архітектури програми	40
3.1 Вибір фреймворку	40
3.2 Основні вимоги до рекомендаційної системи	43
3.3 Висновки за розділом 3	44
4 Розробка програми	45
4.1 Опис моделей реалізованої програми	45
4.1.1 Модель Books	45
4.1.2 Модель Myrating	46
4.1.3 Модель Mylist	47
4.2 Структура проєкту	48
4.3 Висновки за розділом	51

5 Експлуатація, тестування програми	52
5.1 Опис додаткового функціоналу рекомендаційної системи для користувача.....	52
5.2 Опис основного функціоналу рекомендаційної системи для користувача.....	55
5.2.1 Приклад надання рекомендації на основі методу колаборативної фільтрації	55
5.2.2 Приклад надання рекомендації на основі методу контентної фільтрації	56
5.2.3 Приклад надання рекомендації на основі гібридного методу фільтрації	58
5.3 Висновки за розділом 5	59
Висновки.....	60
Перелік джерел посилання	61
Додаток А Технічне завдання.....	64
Додаток Б Текст програми.....	67
Додаток В Слайди презентації	89

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

CMS – Content Management System;

FS – Full-Stack;

HTML – Hypertext Markup Language;

MVT – Model -View-Templates;

OC – OpenCart;

WP – WordPress;

БД – база даних;

РС – рекомендаційна система.

ВСТУП

Сьогодні уявити своє життя без Інтернету вкрай важко. Інтернет з нами з моменту прокидання і аж до моменту як ми лягаємо спати. Він взяв, та продовжує брати на себе безліч функцій, що спрощують нам життя. Найголовніша функція Інтернету це надання різного роду інформації. Однак кількість цієї інформації збільшується майже що миті. Саме тому для будь-якого вебсайту спочатку з'явилась необхідність надати своєму користувачу можливість фільтрувати інформацію, а потім можливість надавати користувачеві рекомендовану інформацію, для того щоб йому зекономити час, а собі покращити прибутки. Кожного дня кількість сайтів збільшується, і його кожен власник хоче вигідно відрізнитись від інших. В таких випадках саме рекомендаційна система (РС), з простим і зрозумілим інтерфейсом, що може швидко надати вдалі рекомендації, тим самим задовільнивши потреби користувача та зекономивши йому час, вигідно відрізняє цей сайт з-поміж інших.

Однак рекомендації необхідні користувачеві не обов'язково при покупці якогось товару. Багато людей знаходить в Інтернеті для себе відпочинок. Це можуть бути ігри, музика, фільми чи книги. Історично склалося, що саме книги є для багатьох найкращим відпочинком. Не дивлячись на те, що книги дедалі більше переходять в онлайн формат, кількість читачів залишається значною. Лише в Індії, де кількість населення вже більше мільярду, жителі витрачають на книги близько 10 годин на тиждень. Тож не дивно, що сервіси, що продають, чи лише є сховищем безлічі книжок є дуже популярними та весь час розвиваються.

Саме тому робота присвячена створенню рекомендаційної системи підбору літератури. Ця система буде використовувати різні методи надання рекомендацій, аби швидко та вдало задовільнити бажання потенційного користувача. Оскільки зараз рекомендаційні системи швидко розвиваються, цю систему можна вважати актуальною у наш час.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Дослідження характеристик рекомендаційних систем

Спочатку, треба дати визначення рекомендаційної системи. Рекомендаційна система – це комп'ютерна програма, головною метою розробки якої є формування рекомендацій різних пропозицій, будь то товар чи сервіс для користувача на основі різних даних.

За останнє десятиліття місце, яке в нашому житті займає інтернет значно збільшилось. Уявити своє життя без всесвітньої павутини настільки ж неможливо, як і уявляти своє життя без світла чи води. Зі своїм розвитком, який не припинявся з моменту створення, Інтернет перестав бути лише системою для обміну інформацією, в якій в будь-який час можна дізнатись будь-що. На сьогодні Інтернет – це майданчик для спілкування, самовираження, обміну інформацією, а також найбільший комерційний майданчик, на якому діють тисячі компаній, заробляючи мільярдні статки. Так у 2020 році обсяг роздрібної торгівлі через Інтернет склав більш ніж 4 трильйона доларів. Саме через такий розвиток комерційних можливостей зріс попит на розробку рекомендаційних систем, що могли б полегшувати в певній мірі користувачу вибору товару и послуги, що в свою чергу може значно збільшити прибуток компанії.

Сьогодні рекомендаційні системи широко використовуються та дуже поширені. Звичайно, найбільше такі системи використовують в комерції де допомагають кожному конкретному користувачеві відповідні товари. Такі системи збирають та аналізують інформацію, щоб рекомендувати користувачу товари чи послуги, які могли б бути для нього цікавими. Зараз існує велика кількість методів, що використовується для формування рекомендацій, і кожен з них має як переваги так і недоліки. Зважаючи на це, дослідження в області рекомендаційних систем залишаються актуальною задачею.

Одними з найголовніших показників якості роботи рекомендаційної системи є точність, з якою буде прогнозовано оцінку користувача та простору користувачів, і покриття простору об'єктів.

Серед додаткових показників якості слід обов'язково звернути увагу на такі, що дозволяють оцінити наявність/відсутність проблеми бульбашки фільтрів та проблем з інформаційною безпекою. Важливими є такі показники якості роботи рекомендаційної системи, як різноманітність, новизна та неочікуваність, які дозволяють оцінити структуру списку рекомендацій та ймовірність виникнення проблеми бульбашки фільтрів. Суттєвими з погляду інформаційної безпеки є такі показники якості роботи рекомендаційної системи як приватність, ризик та робастність. Вони дозволяють оцінити інформаційну безпеку користувача та інформаційну безпеку системи. Безперечно, для кожного окремого вебресурсу чи додатку можна скласти свій набір додаткових показників якості роботи рекомендаційної системи, але розглянуті вище показники будуть актуальними практично для будь-якого випадку [1]. В той же час в залежності від певних потреб деяких вебсайтів можна виділити й інші показники якості, перелічені нижче.

Впевненість (Confidence). РС може додавати до своїх рекомендацій процент впевненості у них. Так наприклад, система може вказати для першої рекомендації прогнозовану оцінку користувача 5 балів з впевненістю на 95%, а для другої рекомендації – 5 балів з впевненістю на 89%. Користувач може враховувати даний параметр та обирати в першу чергу об'єкти з більшим значенням впевненості системи [1].

Довіра (Trust). Дана властивість характеризує наскільки користувач може довіряти даній рекомендаційній системі. Якщо РС запропонувала користувачу лише невідомі йому об'єкти, то користувач може засумніватися у правильності роботи системи, а якщо серед рекомендацій є певна кількість об'єктів, про які він знає і які йому подобаються, то рівень довіри до системи буде вищим. Тобто, деяка кількість об'єктів, про які відомо користувачу, може збільшити довіру до системи. Ще одним варіантом збільшення довіри

до системи є пояснення до рекомендації. Наприклад, коли елемент зі списку рекомендацій подається у вигляді «Якщо Вам подобається X, то спробуйте Y». Також довіра користувача зростає, якщо рекомендації були йому корисні та відповідали його вподобанням, та знижується в протилежному випадку [1].

Адаптивність (Adaptivity). Реальна система рекомендацій може працювати у ситуації, коли колекції об'єктів, і/або інтереси користувачів швидко змінюються. Прикладом таких систем можуть бути рекомендаційні системи новинних сайтів, коли об'єкти (новини) цікаві тільки на проміжку певного часу. В таких системах з одного боку цікавість до певного об'єкту існує тільки певний час, з іншого боку деякі старі об'єкти можуть знову ставати дуже цікавими, якщо нові об'єкти у системі поновлюють інтерес до старих. В таких системах слід вибирати швидкі алгоритми, навіть якщо доводиться в деякій мірі жертвувати точністю. Інший тип адаптивності – це те, з якою швидкістю система адаптується до вподобань користувача або до змін в його профілі. Якщо дії користувача на сайті занадто повільно змінюють рекомендації, він може відмовитися оцінювати об'єкти, не отримавши зворотного зв'язку. Адаптивність алгоритмів можна оцінювати шляхом визначення різниці між списками рекомендацій до та після додавання нової інформації з профіля користувача, наприклад, за допомогою міри ентропії Шеннона [1].

Масштабованість (Scalability). Зі збільшенням об'ємів даних багато алгоритмів працюють значно повільніше або вимагають додаткових ресурсів, таких як обчислювальні потужності або пам'ять. Тому важливо враховувати просторову та часову складність алгоритмів. В реальних системах може виникнути необхідність згодитися на меншу точність рекомендацій для одержання більш масштабованої системи [1].

Пропускна здатність (Throughput) – це кількість рекомендацій, які система може надати в одиницю часу [1].

Корисність (Utility) – значення, що характеризує, яку вигоду від рекомендації отримує система та/або користувач [1].

У Інтернет-магазинах рекомендації для кожного користувача формуються персонально, зважаючи на його дії в минулому на якомусь конкретному вебсайті чи спираючись на його активність в минулому. Також для формування рекомендацій використовується поведінка інших користувачів.

Для Інтернет-магазинів ця функція вкрай важлива, а для великих комерційних майданчиків, наприклад Amazon, є взагалі майже єдиним рішенням для якісного функціонування. В такому випадку РС спрощує навігацію по сайту. Якщо в каталозі присутня велика кількість позицій продукції, скажімо понад 20 000, то орієнтація для користувача стає надважкою і не підвищує шанси вдалої покупки що вигідна обом сторонам, що ж можна сказати якщо товарів сотні тисяч?

Рекомендаційні системи в своїй роботі можуть використовувати різні дані користувача, та різним способом їх збирати. Саме за методом збору інформації рекомендаційні системи діляться на кілька типів.

Перший тип – явний збір даних. Як можна було здогадатися з назви, користувач сам надає необхідні для роботи матеріали. Наприклад, коли рекомендаційні системи Google або інших пошуковиків просять людину дати оцінки різним елементам, скласти список фаворитів певної сфери або ж відповісти на кілька питань. Якщо ж людина відмовляється дати інформацію самостійно, актуальною буде наступна методика [2].

Другий тип – неявний збір даних. Це шпигунська місія, згідно з якою дії учасника процесу фіксуються програмою для подальшої обробки і застосування. Програма розпізнає покупки, оцінки на сайтах, збирає інформацію про перегляди, коментарі. Звичайно, вибір такої методики веде за собою деякі етичні проблеми, адже захист персональних даних – одна з головних вимог користувачів до пошукових сайтів. Але поки факт

залишається фактом – своєрідне стеження можливо, і відвідувачі сайтів перевірити, чи дійсно ведуться подібні заходи, не можуть.

Існують також види рекомендаційних систем, що визначаються за підходами, які вони застосовують [2].

1.2 Аналіз існуючих підходів щодо надання рекомендацій

Існує два типи товарних рекомендацій: персоналізовані та неперсоналізовані. Прикладом неперсоналізованого є рекомендувати найпопулярніші продукти, або порекомендувати відмовитися від добрих завдань.

Основні підходи персоналізовано рекомендації це так звані content-based, тобто такі що базуються на контенті та так звану collaborative filtering, або колаборативну фільтрацію. Також існує підхід, що поєднує в собі їх обох, це гібридний підхід. На рисунку 1.1 зображено схему існуючих підходів для надання рекомендацій [3].

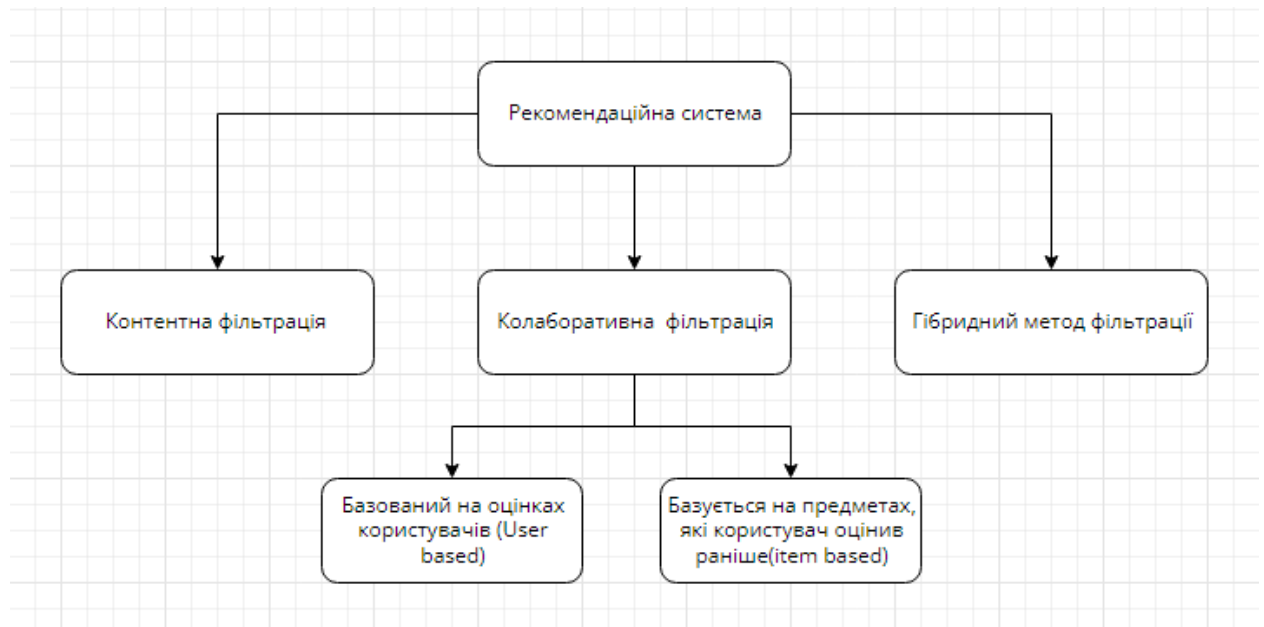


Рисунок 1.1 – Схема існуючих підходів для надання рекомендацій

Основний принцип так званих content-based методів полягає в тому, щоб знайти схожі властивості товарів чи послуг, які попередні користувачі оцінили позитивно, а після цього рекомендувати новому клієнту товари чи послуги, що мають такі ж властивості. Рекомендаційні системи, які в формуванні рекомендацій спираються лише на контент, як правило мають проблеми обмеженого аналізу вмісту та надмірної спеціалізації. Перша проблема трапляється, коли система не має достатньо інформації про товари та клієнтів. Як приклад, вимога клієнта про конфіденційність приватних даних робить неможливим їх використання при аналізі. Також в деяких випадках інформація про товар є недоступною чи дуже складною для аналізу, такі як музика чи картинки. Іноді проблемою може виступати те, що контент товару не завжди є достатнім аби система могла б, проаналізувавши, визначити його якість.

З іншого боку, надмірна спеціалізація є свого роду побічним ефектом того, яким чином рекомендаційні системи на базі content-based методів створюють рекомендації нових товарів, в той час як прогнозована оцінка є високою, та товар є має спільні риси з тими, що вже сподобались користувачу. Наприклад, в додатку для рекомендацій літератури система може рекомендувати користувачу книги такого ж жанру, чи ті що, мають спільного автора з тими книгами, що раніше користувач вже прочитав. Однак, в той же час, система не спроможна порекомендувати книги, які відрізняються від попередніх тим чи іншим параметром, і які також були б цікаві користувачу.

На рисунку 1.2 зображено схему content-based методу [4].

Для того, щоб не бути залежним від так званої контентної інформації, колаборативні методи фільтрації спираються в основному на оцінку товарів іншими користувачами в системі. Основною ідеєю в такому випадку є те, що система вважає, що оцінка товару новим користувачем буде схожою з оцінкою інших користувачів, якщо вони подібним чином оцінили й інші товари. Такий базовий метод фільтрації міг би в знаній мірі подолати основні

проблеми content-based методи. Для прикладу товари, що параметри яких відсутні, важко доступні, чи провести їх аналіз важко, все одно будуть рекомендуватись новим користувачам, базуючись на попередніх оцінках попередніх користувачів. Крім того колаборативні методи рекомендації спираються на якість товарів, що були вже оцінені раніше, ніж базуватись лише на параметрах товару що можуть бути невдалим індикатором якості товару. Також при такому методі рекомендацій система має змогу рекомендувати товари з різними параметрами, тому що попередні користувачі виявили інтерес до цих товарів. На рисунку 1.3 зображено схему методу колаборативної фільтрації [5].

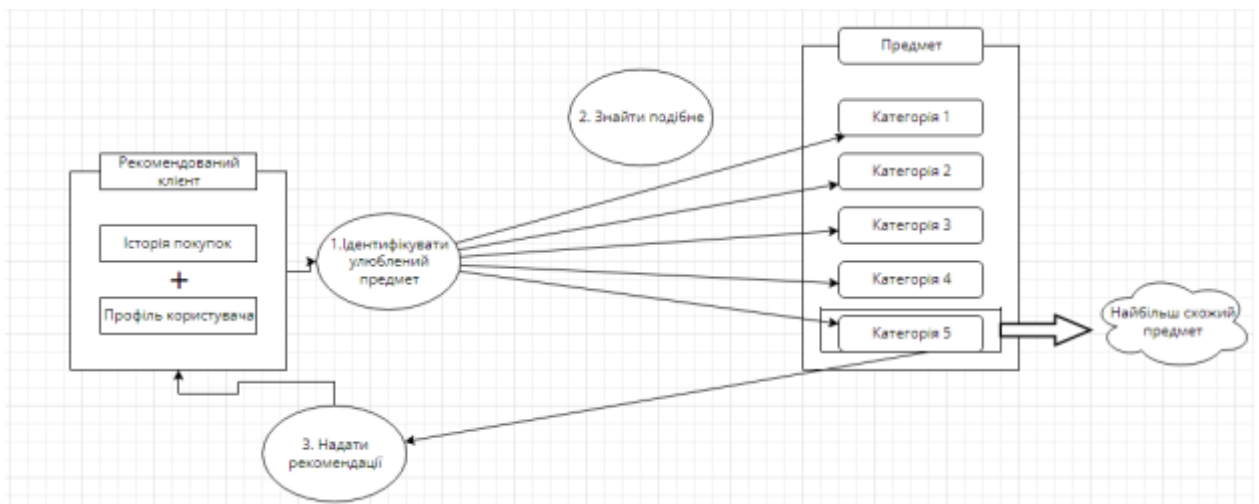


Рисунок 1.2 – Схема Content-based методу

Колаборативні фільтрації можна також умовно поділити за підходами, перший з яких за основу бере алгоритм «пошуку сусідів» а другий – «навчання моделі». В першому підході оцінки, якими користувач оцінив певні товари використовуються безпосередньо під час прогнозування системою вірогідних оцінок для інших, не оцінених товарів. В свою чергу це може бути реалізовано двома способами, а саме user-based та item-based рекомендаціями. Системи що використовують спосіб user-based насамперед оцінюють інтерес певного користувача до певного товару, використовуючи при цьому оцінки так званих «сусідів», тобто користувачів, що мають схожі

моделі поведінки. Для користувача «сусіди» будуть користувачі, оцінки яких найбільше схожі між собою, іншими словами добре корелюються. В той же час підхід item-based, для прогнозування можливої оцінки користувача для певного товару, базується на попередніх оцінках цього ж користувача для інших, схожих товарів. При такому підході декілька товарів вважаються схожими в тому випадку коли певна кількість інших користувачів оцінили їх схожим чином.



Рисунок 1.3 – схема методу колаборативної фільтрації

На рисунку 1.4 зображено алгоритм роботи user-based та item-based рекомендацій [6].

З іншого боку, підхід, який за основу бере метод, що базується на моделях, на відміну від тих що під час безпосереднього прогнозування використовують минулі збережені оцінки, використовують оцінки для того аби навчити систему рекомендувати модель. Множина параметрів моделі поєднує в собі важливі для аналізу характеристики користувачів та товарів, які були отримані безпосередньо за час навчання системи на тренувальних

даних. І саме дані з цієї множини параметрів можуть бути використані під час прогнозування потенційних нових оцінок товарів.

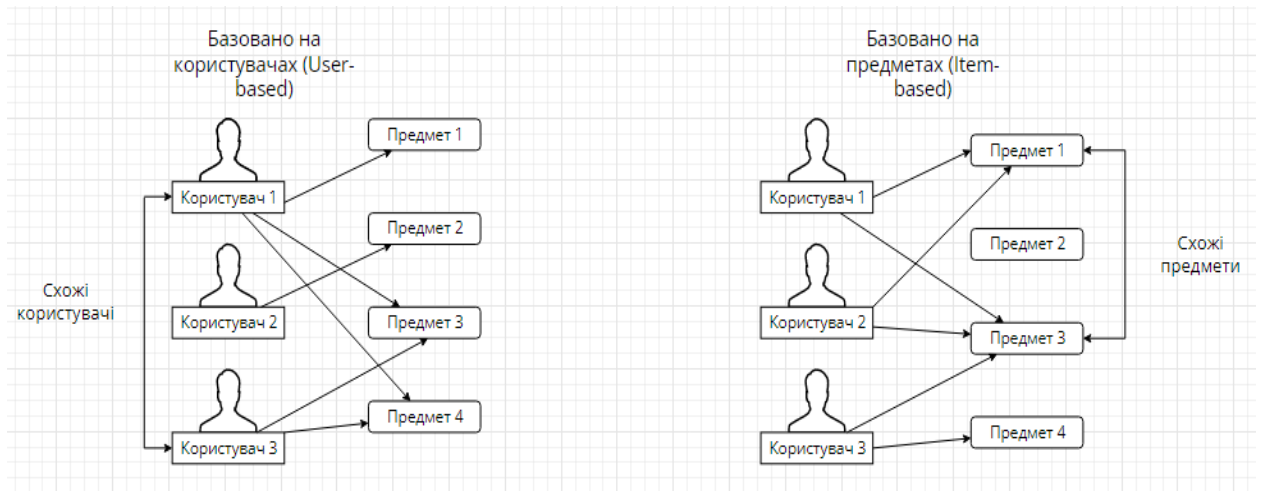


Рисунок 1.4 – Алгоритм роботи user-based та item-based рекомендацій

На останок слід сказати про так звану «гібридну» систему, яка в собі певним чином поєднує методи content-based та колаборативну фільтрацію, дозволяючи тим самим обійти деякі обмеження двох попередніх методів. Існує безліч способів схрестити два методи в один. Наприклад, можна об'єднати деякі списки рекомендацій в один, чи в моделі, що використовує колаборативна система фільтрації додати контентну частину. Проведення досліджень, показало, що впровадження саме «гібридної» системи підвищило ефективність рекомендаційних систем, адже це забезпечило надання точніших рекомендацій ніж при використанні лише content-based та колаборативних методів. Найбільше різницю видно коли в системі наявні данні лише про невелику кількість оцінок. На рисунку 1.5 зображено схему роботи гібридної системи фільтрації [7].

Як уже зазначалося, сегмент інтернету, що пов'язаний з торгівлею (e-commerce), як і інші сегменти стрімко розвивається. Товару на електронних полицях с кожним роком стає дедалі більше, і користувачам вкрай важлива можливість легко відсортовувати товари з їх великої кількості, що найбільше відповідають саме їх бажанням та інтересам. Один з найголовніших способів

вирішити даний «виклик» є рекомендаційні системи, що як і інтернет активно розвиваються. Такі системи видають користувачам персональні рекомендації щодо товарів чи послуг, які б потенційно їм підходили їхнім інтересам. Технології, що використовуються в таких системах базуються на пошуку взаємозв'язку між заздалегідь створеними профайлами користувачів та товарів чи послуг.

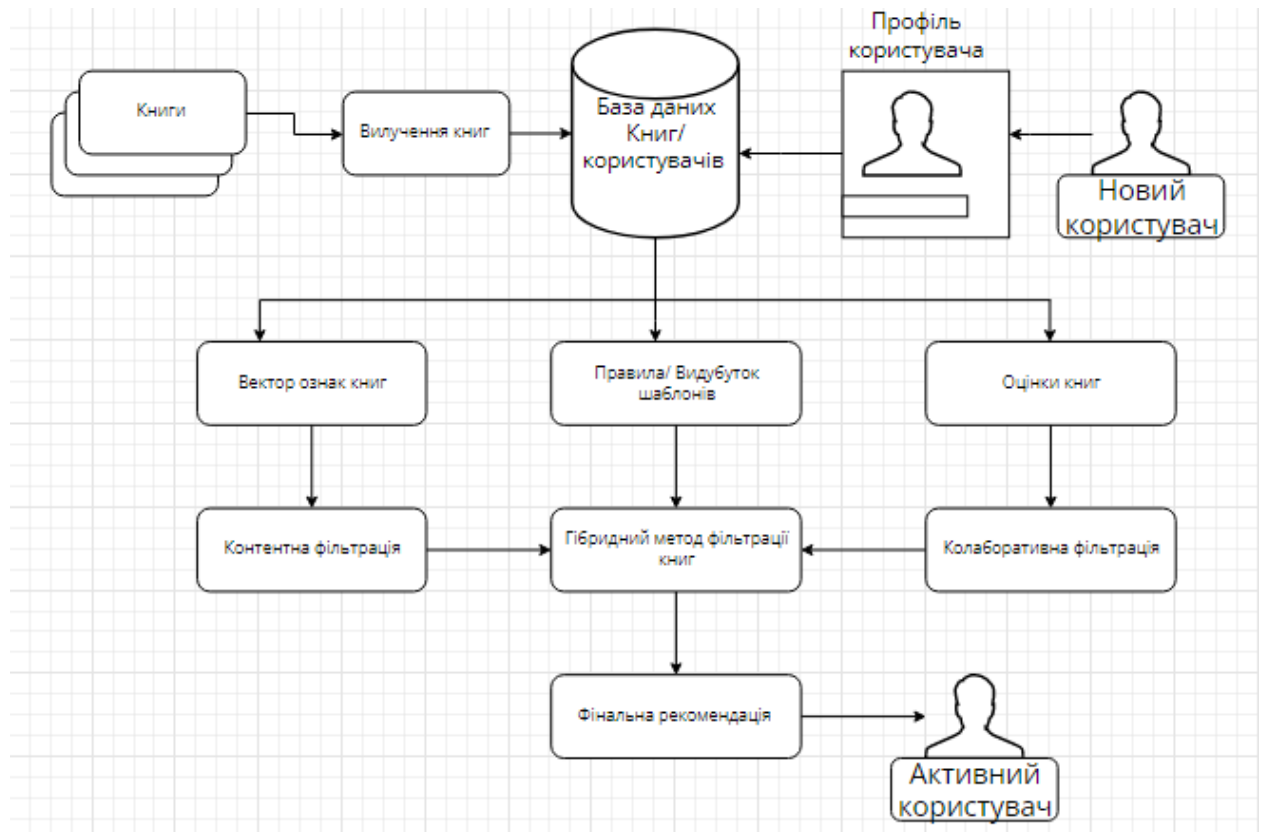


Рисунок 1.5 – Схема роботи гібридної системи фільтрації

Рекомендаційні системи, зазвичай, базуються на двох основних методах, чи методі, що об'єднує ці два методи. Content-based метод в процесі роботи створює окремий профіль з характеристик для кожного окремого користувача та для кожного окремого товару. Наприклад, профіль книги може містити в собі такі характеристики як жанр, автор, рік виходу. В той же час профіль користувача може містити, наприклад, відповіді на відповідні запропоновані запитання. Отриманні дані дозволяють системі, проаналізувавши їх, рекомендувати користувачеві відповідні книги, що

найбільше йому підходять. Проте, такий підхід потребує для результативної роботи збирання достатньої кількості інформації, яку не завжди можна легко отримати.

Альтернативний метод потребує для результативної роботи лише дані про поведінку окремо взятого користувача, без створення відповідного профайлу. Цей підхід називають Collaborative Filtering чи колаборативна фільтрація. Така фільтрація аналізує можливі взаємозв'язки між користувачем та товаром, аби знайти потенційні асоціації виду «користувач-товар». Системи на базі таких методів шукають саме групи товарів, які були однаково оцінені різними користувачами зі схожою моделлю поведінки.

1.3 Особливості предметної області

Оцінками, якими користувачі оцінюють ті чи інші товари можуть бути представлені числовими значеннями. Такі значення частіше називають рейтингами, наприклад, звичні нам зірочки, які позначають оцінку від 1 до 5. Також оцінки можуть бути представлені у вигляді бінарних значень, наприклад сподобалось або не сподобалось. Також варто зазначити, що відгуки від користувача, як правило, отримуються двома основними способами: в явному вигляді (*explicit*), тобто коли користувач залишив свій відгук про товар в системі, в неявному вигляді (*implicit*), як правило, такі відгуки отримують переважно з даних про попередні історії покупок, переглядів, тощо.

Звичайно, оцінки які можна отримати шляхом *explicit*, тобто явним, ціняться значно вище, оскільки користувач сам залишив оцінку до товару, що йому подобається. Водночас, переважна кількість користувачів не вважає за потрібне витратити час на написання відгуків. Такий підхід користувачів часто призводить до втрати важливої інформації, що в свою чергу не дає змоги покращити рекомендаційну систему. Через це в переважній кількості випадків інформацію система отримує неявним (*implicit*) шляхом.

Першим і одним з найголовніших чинників, на який слід звернути увагу, є область застосування, оскільки саме цей чинник має безпосередній вплив на алгоритмічний підхід, який слід застосувати при розробці рекомендаційної системи для того, щоб збільшити її ефективність.

В інтернеті зараз надзвичайно багато різної інформації, різних послуг та товарів, тож існує декілька класів рекомендаційних систем [8]:

- розваги – рекомендації для фільмів, музики, ігор, тощо;
- контент – персоналізовані газети, новини, рекомендації документів, веб сторінок в пошукових запитах, рекомендації онлайн курсів, а також фільтрація по категоріям для електронної пошти;
- електронна комерція – рекомендації товарів для покупок, таких як книжки, телефони, ноутбуки, аксесуари, тощо;
- послуги – рекомендації туристичних послуг, рекомендації експертів для консультації, рекомендації будинків для оренди або послуги знайомств;
- соціальні мережі – рекомендація людей у соціальних мережах та рекомендації контенту в соціальних медіа, такі як твіти, канали Facebook, оновлення LinkedIn та інші.

Нижче перелічено головні функції рекомендаційної системи.

Збільшити відсотку конверсії. Для комерційних майданчиків це перш за все підвищення проценту продаж товарів чи послуг в порівнянні з тим, коли для продажу не використовувалась РС. Цю мету все ж можна досягти, адже вбудовані рекомендаційні системи та товари, які вони рекомендують користувачу, часто можуть задовільнити потреби користувача. Вірогідно, користувач все ж зможе знайти цікавий для нього товар після того, як перегляне список створених для нього рекомендацій. Некомерційні майданчики мають схожу мету, не дивлячись на те, що користувач напямуч не купує товари чи послуги. Наприклад, вебпортал, що публікує новини, може бути зацікавленим щоб користувачі середньому прочитали більше новин. Загалом, основною метою використання рекомендаційної системи на сайті, є збільшення відсотку конверсії. У цьому можна легко впевнитись,

якщо порівняти кількість користувачів, які прийняли рекомендацію системи та знайшли підходящий для них товар з кількістю користувачів що не використовували рекомендаційну систему та просто шукали товари серед всієї наявної інформації.

Рекомендувати більш різноманітні товари. Ще однією важливою метою, беззаперечно є те, щоб РС надавала користувачу різні товари, які в свою чергу без рекомендаційної системи знайти б було вкрай важко. Наприклад, туристичні компанії зацікавлені в тому, щоб РС рекомендувала відвідувачам всі можливі туристичні місця, а не лише найпопулярніші. Це може бути досить складно без рекомендаційних систем, оскільки компанія не може ризикувати рекламувати ті місця, які можуть не сподобатися користувачу. Саме тому, рекомендаційні системи пропонують або рекламують не дуже популярні місця саме для тих користувачів, яким потенційно вони можуть сподобатись [8].

Збільшити задоволення користувача. Добре розроблена РС, може вплинути на враження, яке користувач отримає від користування сайтом. Вдалі рекомендації, привабливий вигляд, зручний функціонал, усе це позитивно впливає на враження користувача. Поєднання вдалих рекомендацій та привабливого користувацького інтерфейсу, значно покращить особисте відношення користувача до системи.

Збільшити лояльність користувачів. Користувачі будуть більш відданим до веб сервісу, якщо під час відвідування, система розпізнає їх як минулих покупців та буде поводитись з ними як з цінними відвідувачами. Це є стандартна особливість рекомендаційних систем, оскільки вони здійснюють рекомендації, на основі попередніх взаємодій користувача, такі як переглянуті, куплені товари, та залишені до них оцінки. Відповідно, чим довше користувач взаємодіє з веб сайтом, тим більш модель стає адаптованою до потреб користувача, отримує краще розуміння його уподобань, рекомендації стають більш персоналізовані [8].

Краще розуміння потреб користувача. Іншою важливою функцією рекомендаційної системи, яка може бути використана в багатьох інших програмах - це опис уподобань користувача, який збирається від користувачів у явному вигляді або прогнозується системою. Постачальник послуг може потім вирішити перевикористати ці знання для багатьох інших цілей, таких як покращити управління запасами товарів та виробництва. Наприклад, в контексті туристичного бізнесу, управління напрямками в організації може вирішити рекламувати специфічні регіони для певних груп користувачів базуючись на аналізі інформації зібраної рекомендаційною системою [9].

1.4 Огляд програмних аналогів

В Інтернеті існує багато сайтів де можна отримати рекомендації щодо літератури. Це в першу чергу комерційні майданчики, що спеціалізуються на продажу книг. Один з найбільших комерційних майданчиків – це Amazon. Він не спеціалізується саме на продажі літератури, проте така категорія товарів наявна у списку категорій і беручи до уваги що Amazon має велику кількість клієнтів, які щодня шукають для себе щось цікаве чи необхідне, слід розглянути, як такий великий комерційний майданчик справляється з задачею рекомендації.

На рисунку 1.6 зображена вкладка «книги» інтернет-магазину «Amazon».

На ній ми бачимо книги, що мають найвищі продажі, а отже можуть бути для нас цікавими, а також значну бокову панель, на якій розміщені безліч категорій товарів.



Рисунок 1.6 – Вкладка книги інтернет – магазину «Amazon»

При переході на якусь категорію книг, наприклад «історію», відкривається схоже вікно, проте усі книги вже відповідної категорії та рекомендації щодо можливих категорій. А також більш приємне та значно менше бокове меню, що дозволяє більш точно відкалібрувати фільтр. На рисунку 1.7 зображена вкладка історичних книг інтернет-магазину «Amazon».

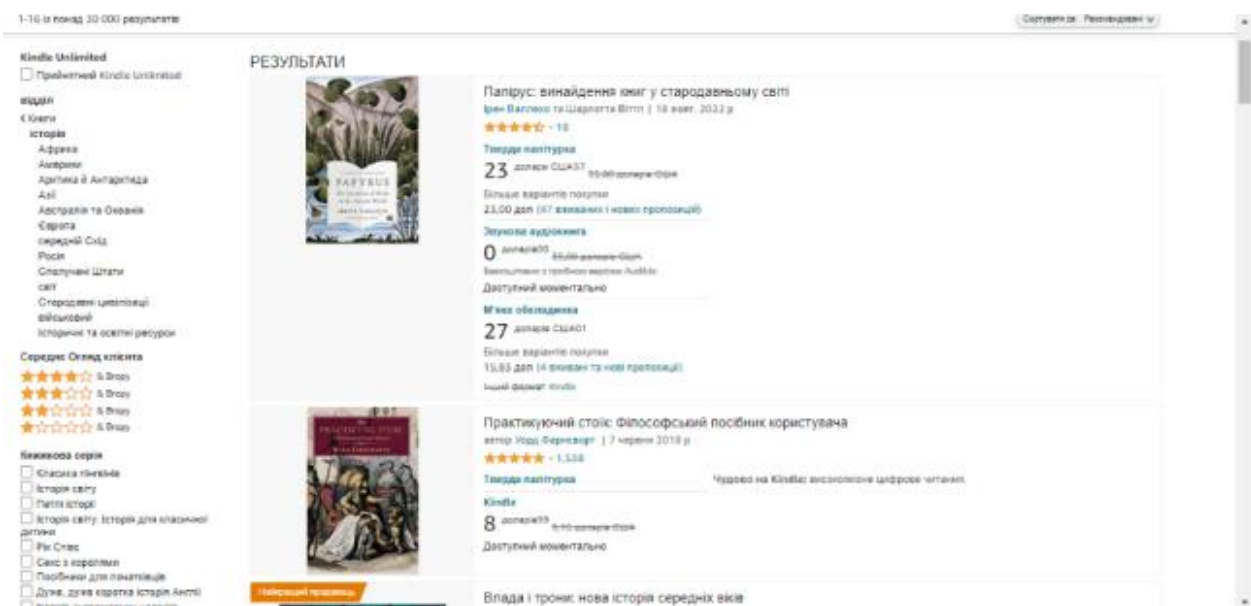


Рисунок 1.7 – Вкладка історичних книг інтернет-магазину «Amazon»

Підводячи підсумок, на комерційному майданчику «Amazon» все ж присутня РС з підбору літератури, вона реалізована доволі просто, тобто книги фільтруються на основі того які користувач обрав фільтри. Для користувача такий спосіб знаходження цікавої літератури є не зручним. По-перше, це велика кількість фільтрів в яких можна «загубитись», по-друге, серед наданих рекомендацій, доступних без втрачання користувача, є лише товари, що мають найвищі продажі.

Наступний, популярний сайт, на якому можна отримати рекомендації з літератури, це «TtasteDive». Цей сайт спеціалізується не на продажі товарів, а на тому, щоб допомогти користувачам відкрити для себе щось нове. Відривати нове користувачі можуть в таких категоріях, як музика, фільми, тв-шоу, книги, ігри. Нас звичайно цікавить, як реалізована рекомендаційна система з підбору літератури на сайті. Спочатку сайт пропонує зареєструватись, зроблено це для того, щоб збирати інформацію про конкретного користувача. На вкладці «книги» нас зустрічають книги, поділені на 4 категорії. На рисунку 1.8 зображена вкладка «книги» сайту «TtasteDive».

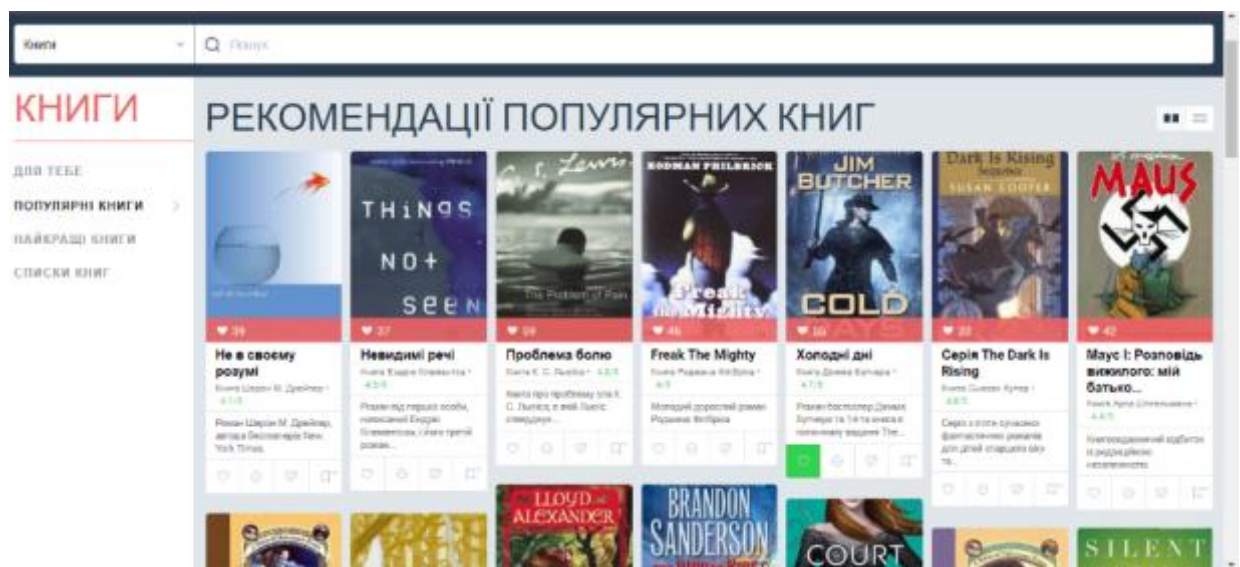


Рисунок 1.8 – Вкладка книги сайту «TtasteDive»

В даних категоріях книгам, які ви вже читали раніше, можна поставили оцінку «Подобається» чи «Не подобається». Цей вибір зберігається і відображається в вкладці профілю, де можна його переглянути чи змінити. Відсутність фільтрів значно ускладнює процес пошуку книг, які б можна було б оцінити. На рисунку 1.9 зображена сторінка профілю сайту «TtasteDive».

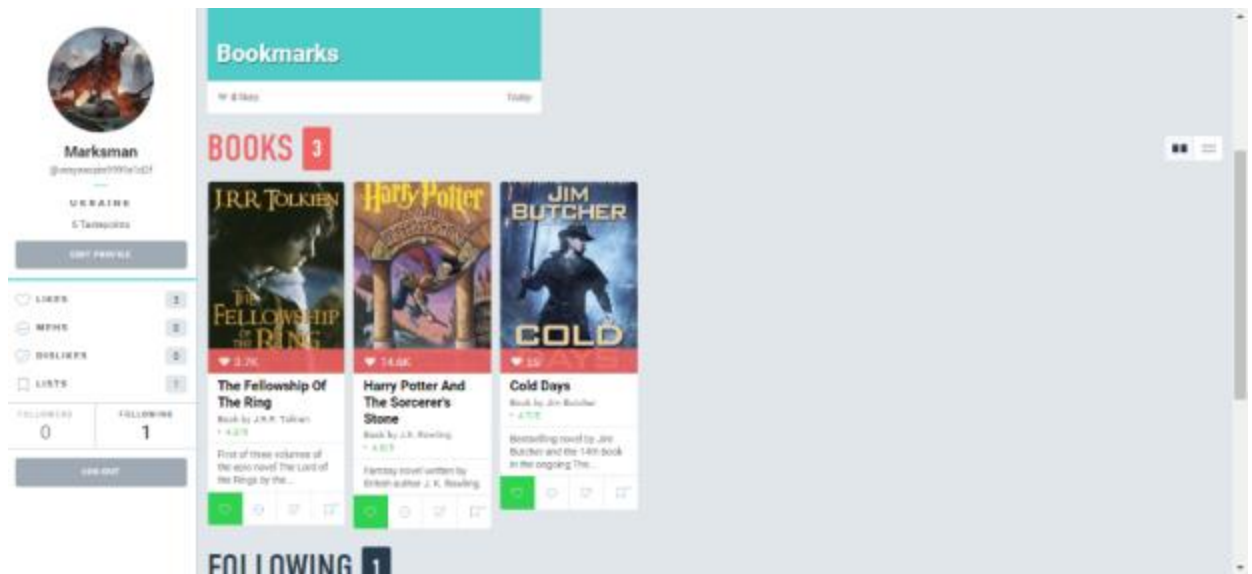


Рисунок 1.9 – Сторінка профілю сайту «TtasteDive»

Після того, як ви залишили хоча б декілька відгуків, система готова вам рекомендувати книги у вкладці «Для вас». Тут слід зауважити, що якщо ви залиште оцінку лише на одній книзі, то система буде вам рекомендувати книги тільки того ж автора та жанру. Повернемось до рекомендацій, що надала нам система. На рисунку 1.10 зображена вкладка «Для вас» «TtasteDive».

Система видає рекомендації на основі даних про книги, яким користувач віддав перевагу. Зокрема це рекомендації книг, які мають схожий жанр, але рекомендації включають в себе у тому числі й книги інших авторів.



Рисунок 1.10 – Вкладка «Для вас» «TtasteDive»

В цілому, дана рекомендаційна система також має для користувача багато незручностей. Якщо користувач не читав попередньо жодної книги з тих, що пропонують йому оцінити, то в купі з відсутністю можливості отримати рекомендації без реєстрації, це робить дану рекомендаційну систему незручною для користувача.

Ну і на останок розглянемо приклад використання рекомендаційної системи з підбору літератури на прикладі сайту «Whichbook». В попередніх аналогах рекомендаційна система базувалась на якомусь одному способі рекомендацій. На сайті «Whichbook», який до речі, спеціалізується саме лише на рекомендаціях літератури, представлено одразу декілька способів надання рекомендацій.

Перший можливий спосіб наведений у вкладці «Настрій та емоції». Тут слід відмітити привабливий та простий інтерфейс, який вірогідно подобається великому відсотку користувачів. На рисунку 1.11 зображена вкладка «Настрій та емоції» сайту «Whichbook». Як ми бачимо, тут присутня доволі специфічна фільтрація, на основі якої система надає свої рекомендації.



Рисунок 1.11 – Вкладка «Настрій та емоції» сайту «Whichbook»

Наступна вкладка, що має назву «Карта світу», говорить сама за себе. На рисунку 1.12 зображено вкладку «Карта світу» «Whichbook». На цій вкладці можна оригінальним способом обрати країну, і система запропонує книги саме цієї країни. На рисунку 1.13 зображено рекомендації книг при обиранні країни «Бразилія».

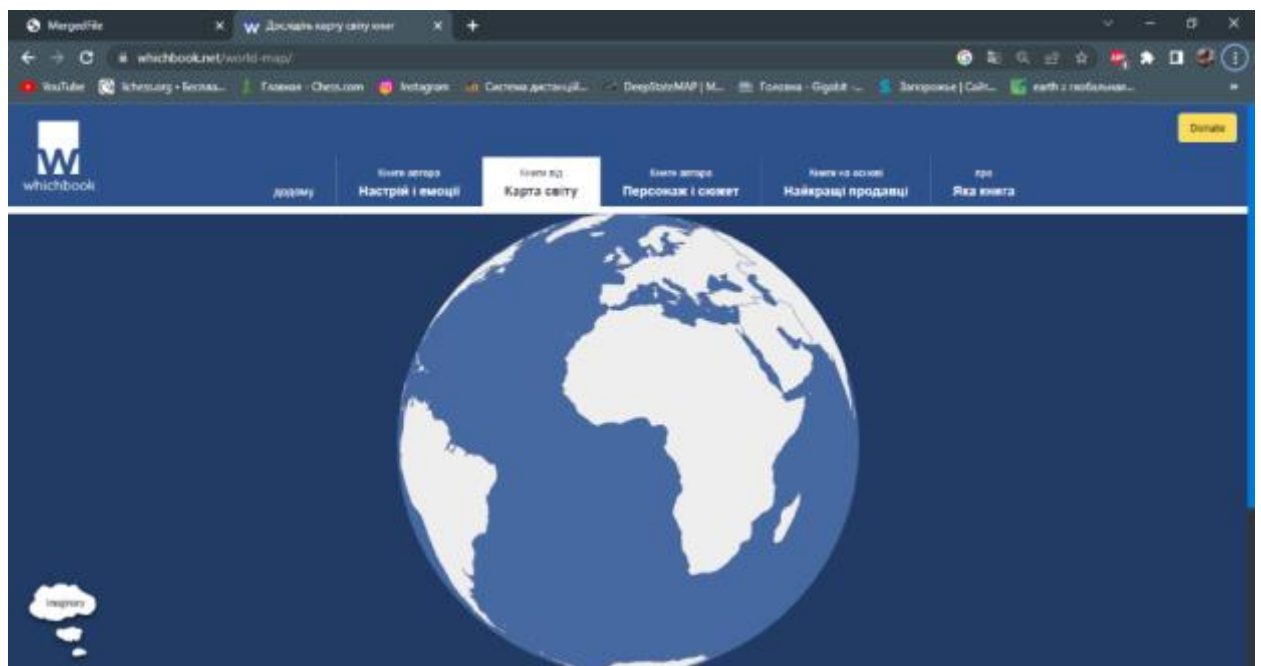


Рисунок 1.12 – Вкладка «Карта світу» «Whichbook»

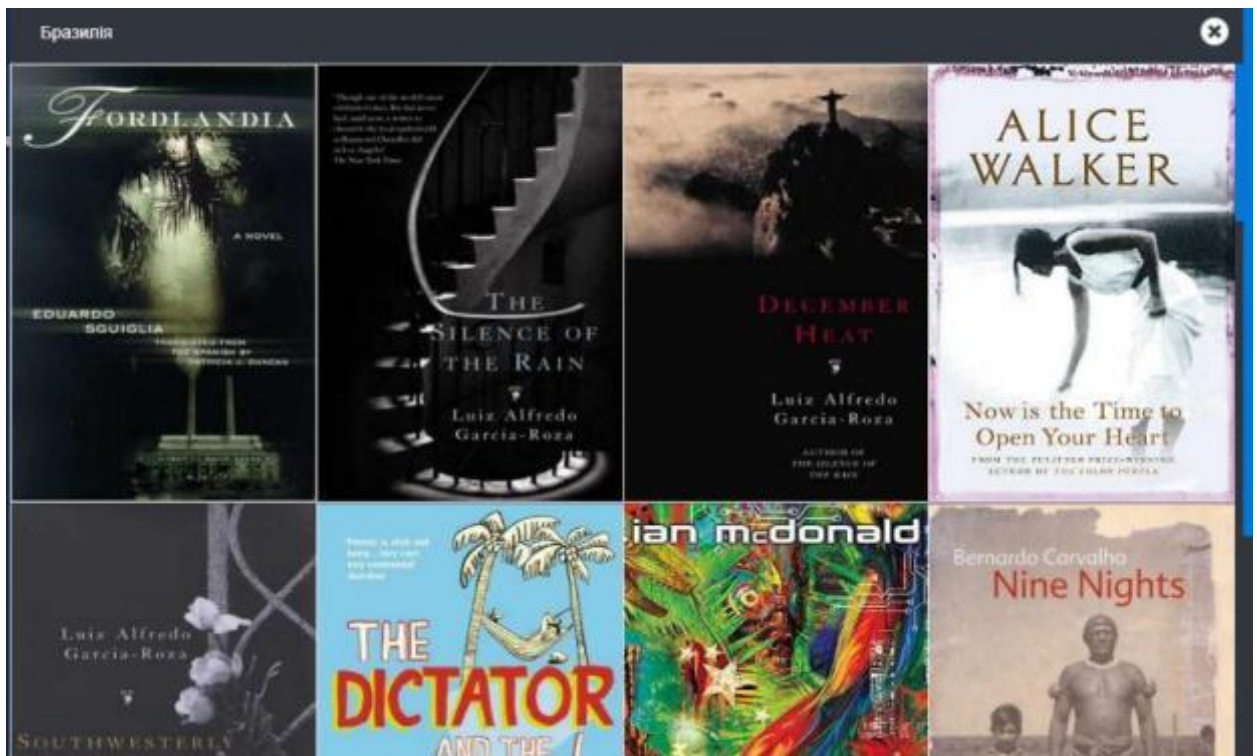


Рисунок 1.13 – Рекомендації книг при обиранні країни «Бразилія»

Третя, та остання вкладка для отримання рекомендацій має назву «Персонаж та сюжет». В ній можна звичним способом відфільтрувати книги за запропонованими параметрами. На рисунку 1.14 зображено вкладку «Персонаж та сюжет» «Whichbook».



Рисунок 1.14 – Вкладка «Персонаж та сюжет» «Whichbook»

Сайт «Whichbook» надає користувачу простий інтерфейс, та декілька незвичних варіантів отримати рекомендації системи. Користувачу легко користуватись сайтом, тому подібний підхід використання рекомендаційної системи можна вважати найбільш вдалим серед розглянутих.

1.5 Постановка задачі

В результаті аналізу предметної області та існуючих рішень було виділено наступні задачі:

- провести дослідження існуючих методів формування рекомендацій;
- розробити рекомендаційну систему підбору літератури;
- провести оцінку якості роботи системи.

1.6 Висновки за розділом

В першому розділі було розглянуто основні характеристики рекомендаційних систем. Було розглянуто метод контентно фільтрації, метод колаборативної фільтрації, гібридний метод фільтрації, а також схеми їх роботи. Було розглянуто декілька найпопулярніших програмних аналогів.

2 ВИБІР ТА ОБҐРУНТУВАННЯ МАТЕРІАЛІВ ТА МЕТОДІВ РОЗРОБКИ

2.1 Порівняння методів формування онлайн-рекомендацій

Раніше розглядалось два основних, дуже не схожих один на одного методів надання онлайн-рекомендацій: колаборативну фільтрацію та фільтрацію контенту. Обидва розглянуті підходи мають як певний набір переваг, так і набір недоліків, якщо вживати їх в певному контексті. Необхідно порівняти ефективність запропонованих методів, а також методу, який базується на обох вище приведених методах.

Колаборативна фільтрація добре себе показує у тих випадках, коли в системі вже складено профіль користувача, в наслідок чого є можливість, на основі даних про користувачів, активність яких можна вважати схожими, з великою точністю передбачити потенційні його вподобання. При використанні такого методу система також може рекомендувати користувачу нові вподобання, адже система спирається не лише на активність окремо взятого користувача, а й на активність користувачів, що є схожими на активного користувача.

У цьому випадку системі не потрібно аналізувати громісткий масив інформації про користувача та параметрів предметів. Системі у даному випадку необхідні дані про те, які користувачі мали взаємодію з якими товарами.

Однак, у колаборативної фільтрації є і свої недоліки. Серед таких можна зазначити проблему «холодного старту» – знижену продуктивність системи щодо нових предметів та нових користувачів. Холодний старт можна розглядати як додаткову проблему охоплення, оскільки він вимірює охоплення системи певним набором предметів та користувачів. На додаток до вимірювання, наскільки великий обсяг предметів холодного запуску або користувачів, також може бути важливим виміряти точність системи для цих користувачів та предметів [10]. Якщо зосередитись на предметах категорії

холодного старту, можна встановити критерії, по яким можна буде сформувати список холодних предметів. Наприклад, ми можемо встановити умову, що холодними предметами будуть лише ті предмети, які ще не були оцінені користувачами та не мають інших доказів використання, або предмети, які були нещодавно додані на сайт, через що побули в системі недостатньо довго, менше певного проміжку часу, чи предмети, які мають менше певної кількості оцінок. Найбільш частими критеріями холодних предметів виступають критерій перебування предмету в системі, чи кількість даних, які система для нього збирала. В такому випадку можна налаштувати систему таким чином, щоб вона частіше рекомендувала холодні предмети та менше гарячі, ті що користуються значною популярністю. В таких умовах система краще рекомендує холодні предмети, розплачуючись при цьому зниженням точності для гарячих предметів. Таке рішення може бути вмотивованим через різні міркування, такі як випадковість чи новизна.

У протипагу цьому контентна фільтрація не має такої проблеми, оскільки схожість між двома користувачами в методах, що базуються на користувачах, що визначає сусідів користувача, зазвичай отримується шляхом порівняння оцінок, зроблених цими користувачами для тих самих предметів [10]. Для прикладу розглянемо таку систему, в якій наявні 100 000 оцінок, які зробили 1000 різних користувачів, і оцінки рівномірно розділились по предметах. В такому випадку кількість так званих «сусідів», тобто користувачів, яких система позиціонує як схожих, становить 650. Проте в такому випадку кількість рейтингів, які система використовує для того, щоб обчислити подібність лише 1. В той же час метод фільтрації контенту для обчислення схожості предметів використовує алгоритм порівняння їхніх характеристик. В такому випадку знайти рекомендовані предмети значно легше.

Також ще однією перевагою цього методу є висока швидкість виконання, ця перевага особливо важлива для комерційних майданчиків, оскільки час відгуку сторінки не має перевищувати однієї секунди.

Звичайно, як і у будь-якого методу, у контентної фільтрації є і свої недоліки. Серед них можна відзначити те, що їхня точність дуже залежить від наповненості даних про предмети. Наприклад, якщо для усіх товарів у системі електронної комерції буде вказаний лише виробник, тоді система буде рекомендувати користувачу лише товари цього бренду, що може бути дуже поганим, якщо цей виробник представлений у цьому магазині лише однією шафою, яку переглядає користувач, та декількома десятками приборів для освітлення. У такому випадку, користувачу, який хоче придбати шафу будуть рекомендовані лише освітлюючі прибори, і загальне враження від магазину, а разом із цим вірогідність того, що користувач обере саме його, значно знизиться [11].

Як було зазначено вище, гібридний метод, який досліджується, поєднує у собі відразу два методи, на основі яких формуються онлайн-рекомендації: фільтрацію контенту та колаборативну фільтрацію. В такому випадку, на відміну від основного підходу він використовує негативний зворотній зв'язок.

Це можна вважати основною перевагою гібридного методу в порівнянні з загальноприйнятою колаборативною фільтрацією на основі неявного зворотнього зв'язку. Системи такого типу не використовують негативні відгуки. Серед причин, по яким це відбувається можна виділити неможливість їх отримати, як наприклад, у системах формування онлайн рекомендацій для систем, що надають своїм користувачам контент певного роду, як, наприклад, стримінговий сервіс Netflix не може отримати негативний неявний зворотній зв'язок від користувача тому, що їх система не може знати напевно, чому користувач не дивився той чи інший фільм або серіал: через те, що від дивився його десь раніше, чув про нього, і він йому не подобається, або ж просто не знає про його існування. У рамках предметної області системи електронної комерції такі дані отримати можливо. Серед таких можуть бути видалення товару з кошика, або повернення товару магазину після покупки. Таким чином ці дані дозволять отримати більш

повну картину про вподобання користувачів, тим самим трохи наблизивши за точністю рекомендацій удосконалений метод до методів колаборативної фільтрації з явним зв'язком [12].

Також метод, що досліджується, частково вирішує проблему «холодного старту» для нових користувачів, оскільки за недостатньої заповненості профілю користувача система просто буде рекомендувати предмети, засновуючись на їх схожості між собою, та не беручи до уваги інші дані [13].

З цього пункту витікає головний недолік гібридного методу фільтрації. Суть цього недоліку полягає в зниженні точності рекомендацій у випадках, коли в профілі користувача недостатньо інформації. У цьому випадку точність великою мірою залежить від інформації про предмети, з якими раніше взаємодіяли.

Ще одним недоліком можна вважати те, що для ефективної роботи системі необхідна техніка з великою обчислювальною здатністю, бо в такому випадку системі треба опрацювати великий масив даних.

В таблиці 2.1 наведене порівняння різних методів формування онлайн-рекомендацій.

Таблиця 2.1 – Порівняння різних методів формування онлайн-рекомендацій

Метод	Переваги	Недоліки
1	2	3
Колаборативна фільтрація (англ. collaborative filtering) Не потрібно аналізувати дані про користувачів та про предмети.	Випадковість – система надає рекомендації предметами, про які користувач міг і не знати.	Якщо багато користувачів взаємодіють з певним товаром, то система буде менше рекомендувати його альтернатив.

Продовження таблиці 2.1

1	2	3
Якість надання рекомендацій поступово збільшується відповідно до того як збільшується кількість взаємодій користувачів з предметами	Системі не треба обробляти наявну інформацію про користувачів.	Система має працювати з недостатньою кількістю даних. Проблема холодного старту для нових товарів та предметів.
Фільтрація контенту	Системі потрібно мати заповнений профіль цього користувача. Системі не треба аналізувати будь-які дані про користувача. Відсутня проблема холодного старту для нових користувачів.	Оскільки представлення предметів певною мірою розроблено вручну, цей метод вимагає знання знань домену. Отже, модель може бути настільки ж хорошою, як і вручну розроблені функції [12]. Даний метод може давати вдалі рекомендації лише на основі даних про існуючі інтереси користувача.

Продовження таблиці 2.1

1	2	3
Гібридний метод фільтрації	Якість надання рекомендацій поступово збільшується відповідно до того як збільшується кількість взаємодій користувачів з предметами. Випадковість – система надає рекомендації предметами, про які користувач міг і не знати. Системі не потрібно мати заповнений профіль цього користувача. Цей метод частково вирішує проблему холодного старту для нових користувачів в системі.	Модель, не має достатніх можливостей, щоб розширити сформовані раніше інтереси користувачів. Точність надання рекомендації великою мірою залежить від ступеня наявності інформації про предмет. Точність надання рекомендацій має велику залежність від заповненості даних про предмет. Для ефективної роботи необхідна техніка з великою обчислювальною потужністю.

2.2 Метрики якості наданої рекомендації

Для визначення того, наскільки добре проранжовані потенційні рекомендації, введемо декілька метрик, які б оцінювали якість цієї рекомендації. Ціль знаходження метрики ранжування – визначити, наскільки отримані алгоритмом оцінки релевантності $r(e)$ і відповідна їм перестановка π відповідають істинним значенням релевантності $r(e)$ [14].

2.2.1 Precision at K (p@K)

Precision at K (p@K) – точність K елементів, є основним показником якості ранжирування для окремого об'єкта. Припустимо, що наш алгоритм ранжування дає оцінки релевантності для кожного $\{r(e)\}_{e \in E}$. Вибравши серед них перші елементи $K \leq M$ з найбільшим $r(e)$ ми можемо обчислити частку релевантних. Саме це реалізує метрика Precision at K, представлена формулою (2.1) [15].

$$p@K = \frac{\sum_{k=1}^K r^{true}(\pi^{-1}(k))}{K} = \frac{\text{кількість релевантних}}{K}. \quad (2.1)$$

В даній формулі $\pi^{-1}(k)$ – це елемент $e \in E$, який потрапив на позицію k , при перестановці π . Отже, $\pi^{-1}(1)$ – це елемент із найбільшою $r(e)$, $\pi^{-1}(2)$ – це елемент із другою за величиною формулою $r(e)$, і т.д.

2.2.2 Average precision at K

Precision at K проста для розуміння і реалізації метрика, але вона має важливий недолік – вона не враховує порядок «зверху». Отже, якщо алгоритм вгадав лише один з десяти елементів, то не має значення, де він

був: перший чи останній, у будь-якому випадку $p@10 = 0.1$. Очевидно, що перший варіант набагато кращий.

Метрика Average precision at K ($ap@K$), не має такого недоліку, яка рівна сумі $p@k$ по індексам k от 1 до K , тільки для релевантних елементів, поділених на K [15].

Метрика Average precision at K вираховується за формулою (2.2):

$$ap@K = \frac{1}{K} \sum_{k=1}^K r^{true}(\pi^{-1}(k)) * p@K. \quad (2.2)$$

2.2.3 Normalized discounted cumulative gain (nDCG)

Розглянемо об'єкт та елементи K з найбільшою $r(e)$. Cumulative gain at K ($CG@K$) – це одна з базових метрик ранжування, що використовує просту ідею: елементи, які більш релевантні, вважаються кращими. Формула (2.3) описує метрику Cumulative gain at K ($CG@K$):

$$CG@K = \sum_{k=1}^K r^{true}(\pi^{-1}(k)). \quad (2.3)$$

Ця метрика має очевидні недоліки: вона не нормалізована і не враховує позицію релевантних елементів [15].

Discounted cumulative gain at K ($DCG@K$) – модифікація cumulative gain at K, що враховує порядок елементів у списку шляхом примноження релевантності елемента на вагу, що дорівнює зворотному логарифму номера позиції формула (2.4):

$$DCG@K = \sum_{k=1}^K \frac{2^{r^{true}(\pi^{-1}(k))} - 1}{\log_2(k+1)}. \quad (2.4)$$

У цьому випадку, якщо r^{true} приймає лише значення 0 чи 1, то формула (2.4) значно спрощується формула (2.5):

$$DCG@K = \sum_{k=1}^K \frac{r^{true}(\pi^{-1}(k))}{\log_2(k+1)}. \quad (2.5)$$

Використання логарифму як функції дисконтування можна пояснити такими інтуїтивними міркуваннями: з точки зору ранжирування позиції на початку списку відрізняються набагато сильніше, ніж позиції його кінці.

2.3 Вибір матеріалів та методів розробки

Не дивлячись на те, що в наш час дуже активно розвивається мобільна розробка, все ж найбільш поширеною Інтернет-платформою є вебсайт. Тож доцільно розробляти рекомендаційну систему на базі вебсайту.

Вище ми розглянули та порівняли між собою різні методи та способи надання рекомендацій, їх переваги та недоліки. Враховуючи що жоден з методів має як свої переваги, так і свої слабкі сторони, необхідно при розробці рекомендаційної системи, застосувати усі ці методи, щоб надати користувачеві не один, а декілька варіантів отримання рекомендацій, що в свою чергу збільшить привабливість рекомендаційної системи для користувача.

2.4 Висновки за розділом 2

Було виконано порівняння а також визначено недоліки та переваги методів надання рекомендацій, такі як колаборативна фільтрація, фільтрація контенту та гібридний метод фільтрації. Було розглянуто метрики оцінки якості наданої рекомендації та обрано матеріали та методи для розробки.

3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

3.1 Вибір фреймворку

Сьогодні, для розробки вебсайтів використовують різні мови програмування та різні підходи розробки. Найпоширеніші мови програмування для розробки вебсайтів це в першу чергу Java, Python, PHP.

Кожна цих мов має своє місце у вебпрограмуванні та займає свою нішу. Для створення торгового майданчика була обрана мова програмування Python. Python – це потужна мова програмування, якою легко оволодіти. Вона має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять його ідеальним для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ [16].

Якщо порівнювати Python з Java чи PHP, то перший демонструє значну ефективність, а також зрозумілу структуру коду завдяки високій «читаемості». Також доволі простий синтаксис Python, приваблює все більше і більше розробників, тому ця мова є популярною та актуальною у наш час.

Для створення вебсайтів використовують два найпоширеніші способи, на базі Content Management System (CMS) або на базі фреймворку. Кожне з цих рішень має свої переваги та недоліки.

CMS – це середовище інструментів для створення та адміністрування сайту без знання мов програмування (Content Management System – система управління вмістом). Система управління сайтом, або CMS-движок, містить готові шаблони структур, набір функцій для проектування та наповнення даними. CMS сайту дозволяє зручно підтримувати ресурс – контролювати доступ до вмісту, змінювати дані на сторінках, надсилати пошту. Сучасний ринок побудови сайтів пропонує користувачам різні типи CMS, орієнтовані на створення сайтів будь-якої складності – від простої візитної картки

компанії до високонавантажених порталів [17]. Найпоширеніші CMS це Wordpress, OpenCart, Joomla.

Фреймворк – це інфраструктура програмних рішень, що полегшує розробку складних систем. Спрощено дану інфраструктуру можна вважати своєрідною комплексною бібліотекою, але при цьому вона має ряд обмежень, що задають правила створення структури проєкту та написання коду [18].

Порівнюючи CMS та фреймворк, можна сказати, що CMS – це шаблоні рішення, в той час як фреймворк може забезпечити індивідуальні рішення для тієї чи іншої задачі. Тому РС більш доречніше розробляти саме на фреймворку.

На мові програмування Python є декілька поширених фреймворків. Вони діляться на Full-Stack (FS) фреймворки та мікрофреймворки.

FS фреймворки – це фреймворки, що мають велику кількість вбудованого функціоналу, в той час як мікрофреймворки мають лише мінімальний функціонал.

Для роботи було обрано фреймворк Django, бо на відміну від Flask, Django має такі переваги:

- дуже швидку роботу;
- безпека проєкту;
- надійність;
- легкість масштабування;
- гнучкість.

При створенні сайту, як правило, завжди необхідні деякі однакові компоненти, такі як панель керування сайтом, реєстрація, авторизація та форми. Люди з часом помітили це, тому вони об'єднали їх в фреймворки. Django – чудовий приклад фреймворка. Django використовує структуру MVT, яка розшифровується як Model View Templates. В MVT структурі: Model – шар, доступний для даних, View – шар бізнес-логіки, Templates – шар представлення даних [19]:

– Model – модель, шар, що буде виступати в якості інтерфейсу ваших даних. Він відповідає за ведення даних. Це логічна структура даних, що лежить в основі всього програми, яка представлена базою даних (зазвичай це реляційні бази даних (БД), такі як MySQL, Postgres) [20];

– View – це представлення даних в певному форматі, ініційоване рішенням контролера представити дані. Це системи шаблонів на основі сценаріїв, такі як JSP, ASP, PHP, і їх дуже легко інтегрувати з технологією AJAX [21];

– Templates – як зрозуміло з назви, містить статичний HTML шаблон, а також конкретний синтаксис, який описує спосіб вставлення динамічного змісту.

Більш наочно, роботу цієї схеми наведено на рисунку 3.1 [22].

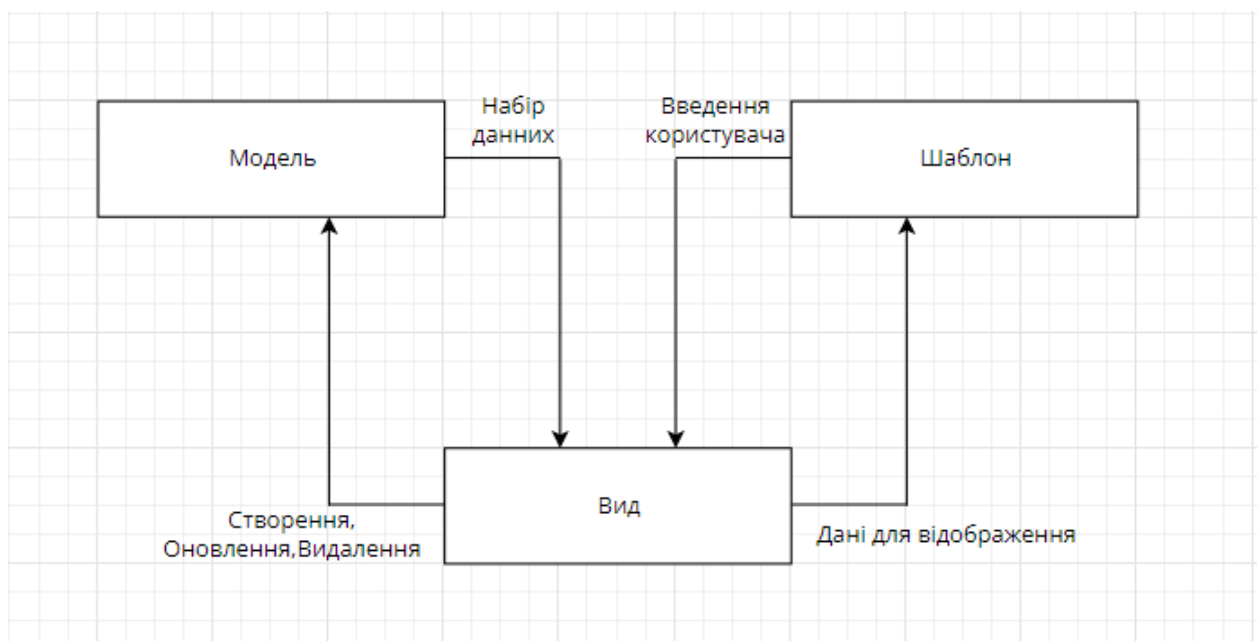


Рисунок 3.1– Схема роботи Model View Template

Django-проект зазвичай містить такі файли:

– Manage.py – користувач може через цей файл може взаємодіяти з Django-проектом. Запустити сервер, взаємодіяти з базами даних, створити суперкористувача, усе це можна легко виконати з допомогою цього файлу;

- View.py – надзвичайно важливий файл. У ньому можна програмно прописати взаємодію Django з HTML-шаблоном. Наприклад форми, віджети, використання в шаблоні тих чи інших моделей прописується саме у цьому файлі;
- Models.py – як видно з назви, цей файл вміщує в собі моделі Django-проєкта. Моделі є основою проєкта;
- Setting.py – назва говорить сама за себе, у цьому файлі зберігаються основні налаштування проєкту. Тут реєструються усі створені додатки, зберігається конфігурація БД, вказується розташування статичних файлів, вказується мова проєкту, часовий пояс та ін.;
- urls.py – цей файл відповідає за посилання. Аби перейти з однієї HTML- сторінки на іншу, треба спочатку прописати це у даному файлі;
- _init_.py - файл містить в собі додаткові пакети Python, які могли знадобитись під час розробки.

3.2 Основні вимоги до рекомендаційної системи

Нижче наведено основні вимоги користувача до розроблюваної рекомендаційної системи:

- можливість отримувати рекомендації на основі оцінок інших користувачів;
- можливість отримувати рекомендації на основі книг, що сподобались;
- можливість пошуку книг;
- функція аутентифікації;
- зручне використання вебсайту;
- стабільна робота вебсайту.

3.3 Висновки за розділом 3

У цьому розділі було розглянуто та обрано для розробки мову програмування та фреймворк, який буде використовуватись для розробки РС, попередньо порівнявши можливі рішення. Було розглянуто схему роботи фреймворку та основні його файли. Також було висунуто основні вимоги РС з боку користувача.

4 РОЗРОБКА ПРОГРАМИ

4.1 Опис моделей реалізованої програми

Перед початком розробки РС, було використано деякі додаткові бібліотеки, а також пакети, що полегшують розробку. Серед них такі бібліотеки як Django 3.10, Bootstrap4, Slugify.

При розробці проєкту було розроблено кілька моделей, які використовуються в усіх частинах проєкту, серед них : Books, Myrating, Mylist.

Модель є єдиним джерелом інформації про дані. Вона містить основні поля і поведінку даних, які зберігає програма. Як правило, кожна модель відображається в одну таблицю бази даних [23].

Кожна модель має певну кількість полів, що описують структуру даних цієї моделі. Розглянемо детально ці моделі.

4.1.1 Модель Books

Ця модель розроблена, щоб додавати книги в систему. Нижче на рисунку 4.1 наведено структуру цієї моделі.

```
class Books(models.Model):  
    title = models.CharField(max_length=200)  
    genre = models.CharField(max_length=100)  
    author = models.CharField(max_length=140, default='SOME STRING')  
    tags = models.CharField(max_length=140, default='#')  
    tagsSize = models.CharField(max_length=140, default='#')  
    genreSecond = models.CharField(max_length=50, default='SOME STRING')  
    genreThird = models.CharField(max_length=50, default='SOME STRING')  
    Books_logo = models.FileField()
```

Рисунок 4.1 – Модель Books

У даній моделі представлені наступні поля:

- title – відповідає назві книги. Тип поля CharField, який дає змогу зберігати рядок тексту чи цифр;
- genre – відповідає жанру книги. Тип поля CharField, який дає змогу зберігати рядок тексту чи цифр;
- author – відповідає автору, що написав книгу. Тип поля CharField, який дає змогу зберігати рядок тексту чи цифр;
- description – відповідає опису книги. Тип поля CharField, який дає змогу зберігати рядок тексту чи цифр;
- tags – відповідає за основну після жанру характеристику книги, такі як орієнтованість на сюжет, орієнтованість на дитячу аудиторію чи дорослу аудиторію;
- tagsSize – відповідає за розмір книги, указує чи книга є частиною серії книг, чи можливо це коротка книга на один вечір;
- genreSecond – відповідає другорядному жанру книги, що збільшить відсоток вдалих рекомендацій. Тип поля CharField, який дає змогу зберігати рядок тексту чи цифр;
- genreThird – відповідає додатковому жанру книги. Це збільшить відсоток вдалих рекомендацій. Тип поля CharField, який дає змогу зберігати рядок тексту чи цифр;
- Books_logo – відповідає за відображення обкладинки книги на сайті.

4.1.2 Модель Myrating

Ця модель необхідна для зберігання оцінок, зроблених користувачами. Нижче на рис. 4.2 наведено структуру цієї моделі:

```
class Myrating(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    books = models.ForeignKey(Books, on_delete=models.CASCADE)
    rating = models.IntegerField(default=0, validators=[MaxValueValidator(5), MinValueValidator(0)])
```

Рисунок 4.2 – Модель Myrating

У даній моделі представлені наступні поля :

- user – відповідає за зв'язок цієї моделі з певним користувачем, що залишив оцінку. Тип поля ForeignKey, це поле даних таблиці, посиляється на первинний ключ в іншій таблиці, в даному випадку на таблицю з даними користувача;
- books – відповідає за зв'язок цієї моделі головною моделлю Books, аби мати змогу прив'язати конкретну оцінку до конкретної книги. Тип поля ForeignKey, це поле даних таблиці, посиляється на первинний ключ в іншій таблиці, в даному випадку на таблицю з даними про книгу;
- rating – відповідає оцінці, яку залишив користувач. Може бути від 0 до 5. Тип поля IntegerField, який дає змогу зберігати рядок цифр.

4.1.3 Модель Mylist

Ця модель необхідна для зберігання та додавання до списку книг по бажанню користувача . На рис. 4.3 наведено структуру цієї моделі.

```
class MyList(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    books_list = models.ForeignKey(Books, on_delete=models.CASCADE)
    watch = models.BooleanField(default=False)
```

Рисунок 4.3 – Модель Mylist

У даній моделі представлені наступні поля :

- user – відповідає за зв'язок цієї моделі з певним користувачем, що залишив оцінку. Тип поля ForeignKey, це поле даних таблиці, посиляється на

первинний ключ в іншій таблиці, в даному випадку на таблицю з даними користувача;

- `books_list` – відповідає за зв'язок цієї моделі головною моделлю `Books`, аби мати змогу прив'язати конкретну оцінку до конкретної книги. Тип поля `ForeignKey`, це поле даних таблиці, посилається на первинний ключ в іншій таблиці, в даному випадку на таблицю з даними про книгу;

- `watch` – відповідає за наявність чи відсутність книги в списку користувача. Тип поля `BooleanField`, який дає змогу зберігати або `True`, або `False`.

4.2 Структура проєкту

Під час реалізації проєкту, була визначена наступна структура, наведена на рис. 4.4.

Ім'я	Дата змінення	Тип	Розмір
book_recommender	29.11.2022 14:48	Папка файлів	
media	29.11.2022 14:48	Папка файлів	
recommend	07.12.2022 13:05	Папка файлів	
db.sqlite3	29.11.2022 13:17	Файл SQLITE3	192 КБ
manage	18.09.2020 14:05	Файл PY	1 КБ

Рисунок 4.4 – Структура проєкту

Файл `db.sqlite3` містить базу даних проєкту.

Файл `Manage.py` – файл, через який можна взаємодіяти з Django-проєктом.

Папка `book_recommender` – це головна папка проєкту (рис. 4.5), в ній зберігаються такі файли як:

- `settings`, які містять основні налаштування проєкту;
- `urls`, які є головним `url`-файлом, що відповідає за перехід між HTML-сторінками, в рамках усього проєкту;

- asgi/wsgi – клієнт-серверний протокол, що забезпечує взаємодію між Django-проєкту та вебсервером;
- __init__ – пустий файл для взаємодії Python та Django.
- додаткові папки _pycache_, aws, що містять допоміжні файли.

Ім'я	Дата змінення	Тип	Розмір
__pycache__	29.11.2022 14:48	Папка файлів	
aws	29.11.2022 14:48	Папка файлів	
__init__	18.09.2020 14:05	Файл PY	0 КБ
asgi	18.09.2020 14:05	Файл PY	1 КБ
settings	26.10.2022 19:06	Файл PY	4 КБ
urls	13.11.2022 15:04	Файл PY	2 КБ
wsgi	18.09.2020 14:05	Файл PY	1 КБ

Рисунок 4.5 – Папка book_recommender

Папка media, в якій зберігаються зображення, що відповідають обкладинкам книг.

Папка recommend, папка головного додатка проєкту, в ній зберігаються файли, що описують та реалізують логіку роботи Django-проєкту (рис. 4.6). В цій папці знаходяться такі файли:

- __init__ - пустий файл для взаємодії Python та Django;
- admin - у цьому файлі підключаються моделі, які використовуються в проєкті, та з якими можна взаємодіяти через панель адміністратора;
- apps – у цьому файлі прописуються додатки проєкту;
- form – в даному файлі прописується функціонал форм, що будуть використовуватись в проєкті, наприклад форма реєстрації чи авторизації;
- models – як було сказано в попередньому розділі, тут зберігаються моделі, що використовуються в проєкті;
- test – як зрозуміло з назви, тут програмуються тести для перевірки проєкту;

– urls – url файл додатка, який взаємодіє з головним url-файлом з папки book_recommender, та відповідає за url-посилання, що використовуються в даному додатку;

– views – один з основних файлів проєкту, який програмно зв’язує моделі та HTML-файли;

– папка __pycache__ – містить допоміжні файли;

– папка migrations – містить файли, що програмують базу даних, згідно з моделями проєкту;

– папка static – містить допоміжні css та js файли;

– папка templates – містить HTML-шаблони (рис. 4.7).

Ім'я	Дата змінення	Тип	Розмір
__pycache__	29.11.2022 14:48	Папка файлів	
migrations	29.11.2022 14:48	Папка файлів	
static	29.11.2022 14:48	Папка файлів	
templates	29.11.2022 14:48	Папка файлів	
__init__.py	18.09.2020 14:05	Файл PY	0 КБ
admin.py	08.11.2022 17:57	Файл PY	1 КБ
apps.py	18.09.2020 14:05	Файл PY	1 КБ
forms.py	18.09.2020 14:05	Файл PY	1 КБ
models.py	07.12.2022 13:05	Файл PY	2 КБ
tests.py	18.09.2020 14:05	Файл PY	1 КБ
urls.py	21.11.2022 20:03	Файл PY	1 КБ
views.py	29.11.2022 13:03	Файл PY	7 КБ

Рисунок 4.6 – Папка recommend

Ім'я	Дата змінення	Тип	Розмір
 base	29.11.2022 13:15	Файл HTML	5 КБ
 detail	09.11.2022 14:24	Файл HTML	5 КБ
 form_template	18.09.2020 14:05	Файл HTML	1 КБ
 genres	29.11.2022 13:21	Файл HTML	3 КБ
 list	21.11.2022 20:03	Файл HTML	2 КБ
 login	18.09.2020 14:05	Файл HTML	3 КБ
 recommend	21.11.2022 17:34	Файл HTML	1 КБ
 signUp	18.09.2020 14:05	Файл HTML	2 КБ
 tags	21.11.2022 19:24	Файл HTML	2 КБ
 watch	29.11.2022 14:41	Файл HTML	2 КБ

Рисунок 4.7 – Папка templates

4.3 Висновки за розділом

Було розглянуто моделі, що були розроблені під час програмної реалізації проекту, а саме Books, Myrating, Mylist. Було розглянуто структуру проекту. Було детально розглянуто, якими полями представлені моделі, та з який файлів складається проєкт.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ПРОГРАМИ

5.1 Опис додаткового функціоналу рекомендаційної системи для користувача

Розроблена РС має змогу надати користувачу рекомендації трьома різними способами, а також має додатковий функціонал для більш зручного користування нею. Перш ніж розпочати опис, слід зазначити, що для того, аби повною мірою користуватись РС, слід зареєструватись в системі.

Користувач має можливість переглянути увесь список наявних в системі книг, та знайти ту, що його цікавить вже на першій сторінці системи. На рисунку 5.1 зображено фрагмент головної сторінки РС.

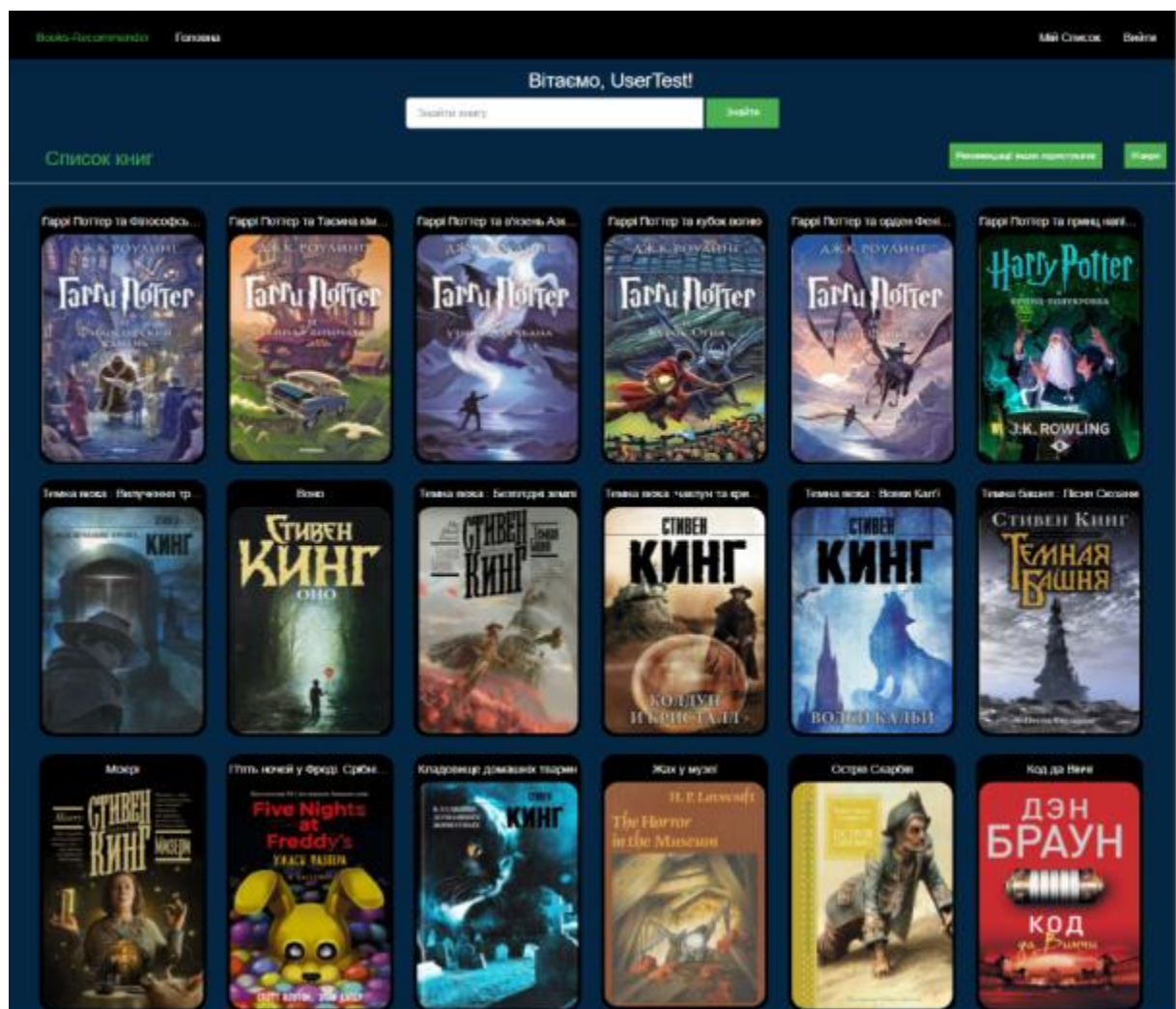


Рисунок 5.1 – Фрагмент головної сторінки РС

На рисунку 5.2 зображено результат пошуку .

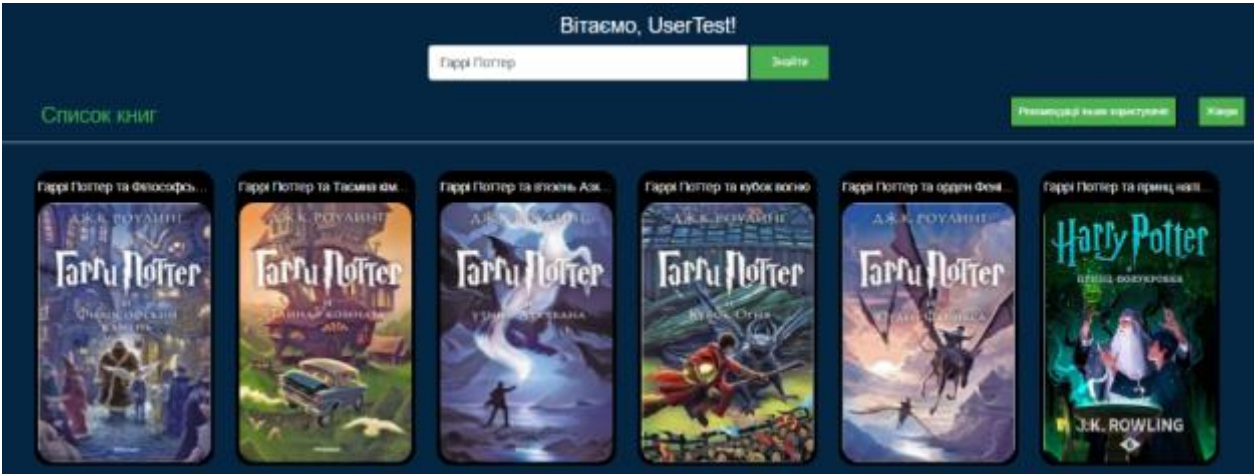


Рисунок 5.2 – Результат виконання пошуку

Користувач має змогу переглядати детальну інформацію про будь-яку книгу. На рисунку 5.3 зображено детальну сторінку книги «Код да Вінчі».



Рисунок 5.3 – Детальна сторінка книги

В цій вкладці, користувач може оцінити книгу, поставивши у відповідне поле оцінку від 1 до 5, або додати до списку цікавих книг чи видалити з нього. На рисунку 5.4 зображено вкладку «Ваш список».

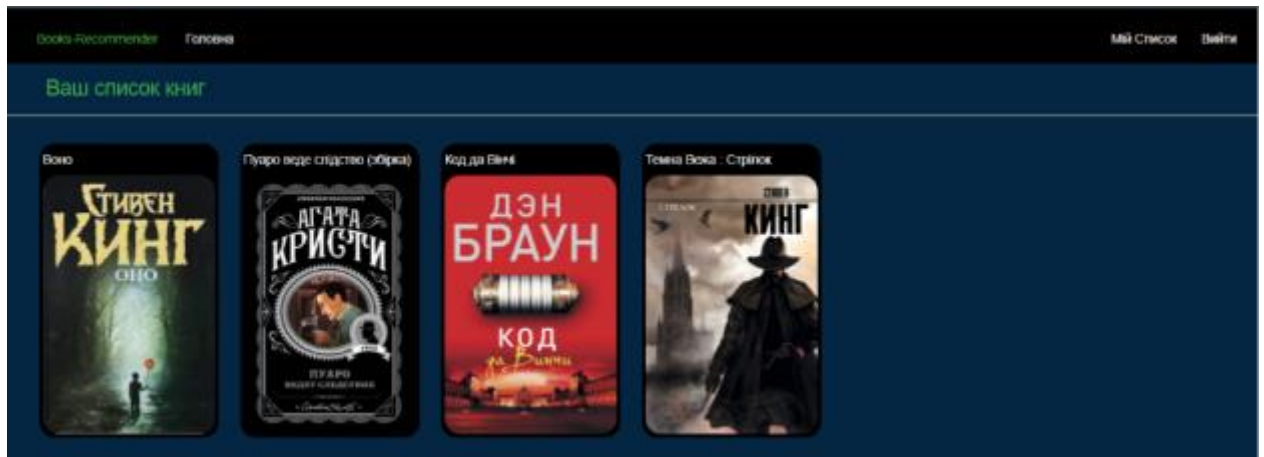


Рисунок 5.4 – Вкладка «Ваш список»

Також в системі присутня функція реєстрації та авторизації. На рисунках 5.5 та 5.6 зображено сторінки реєстрації та авторизації.

Рисунок 5.5 – Форма реєстрації

Рисунок 5.6 – Форма авторизації

5.2 Опис основного функціоналу рекомендаційної системи для користувача

5.2.1 Приклад надання рекомендації на основі методу колаборативної фільтрації

Суть надання рекомендації на основі методу колаборативної фільтрації, як зазначалось вище, використовує оцінки інших користувачів. В присутній відповідній вкладці «Рекомендації інших користувачів» система автоматично рекомендує кожному користувачу книги, які отримали найбільше позитивних оцінок від різних користувачів, на момент реєстрації чи авторизації в системі, не враховуючи такі характеристики, як жанр, автор, додаткові параметри. Тому для різних користувачів, що авторизувались у різний час, пропонуються різні рекомендації, які є актуальними на момент авторизації. Зроблено це для зменшення навантаження на систему. На рисунку 5.7, 5.8 зображено надання рекомендації на основі методу колаборативної фільтрації для різних користувачів.

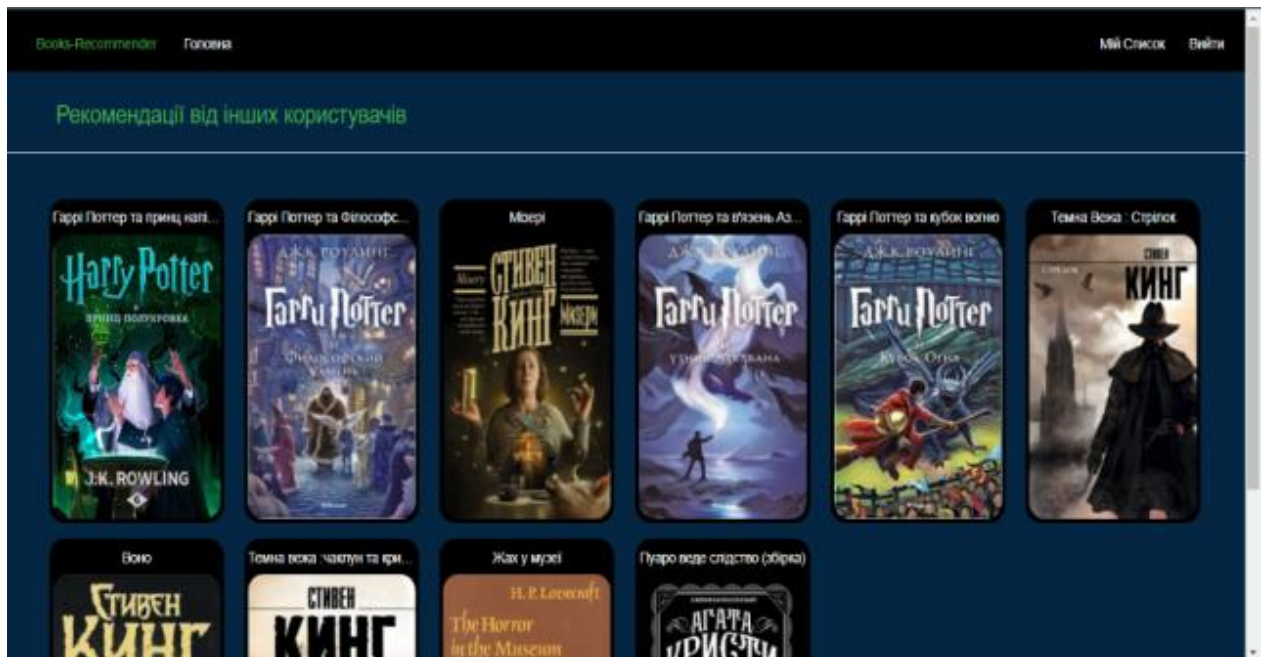


Рисунок 5.7 – Надання рекомендації на основі методу колаборативної фільтрації

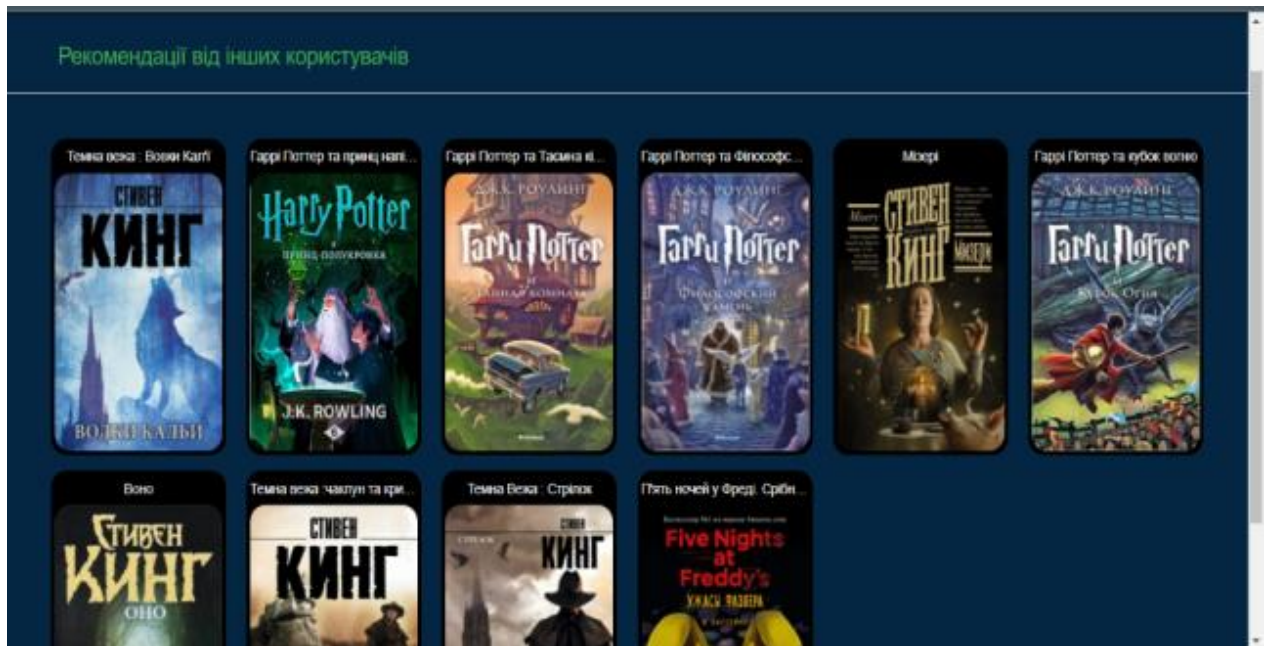


Рисунок 5.8 – Надання рекомендації на основі методу колаборативної фільтрації

5.2.2 Приклад надання рекомендації на основі методу контентної фільтрації

Надання рекомендації на основі методу контентної фільтрації, як зазначалось вище, використовує схожість параметрів. Якщо користувачу сподобався якийсь предмет, система буде відбирати для нього схожі предмети, що мають аналогічні параметри. Розроблена РС має змогу надавати і такі рекомендації. Для цього система має зрозуміти, які книги цікаві користувачу, для цього система використовує список книг, з якими взаємодіяв користувач, що знаходяться у відповідній вкладці. У вкладці «Мій список», нижче від списку книг, знаходяться рекомендації схожих книг, за такими параметрами як : жанр, автор, вікові обмеження, додаткові жанри. На рисунках 5.9, 5.10, зображено рекомендації, надані на основі методу контентної фільтрації для різних користувачів.

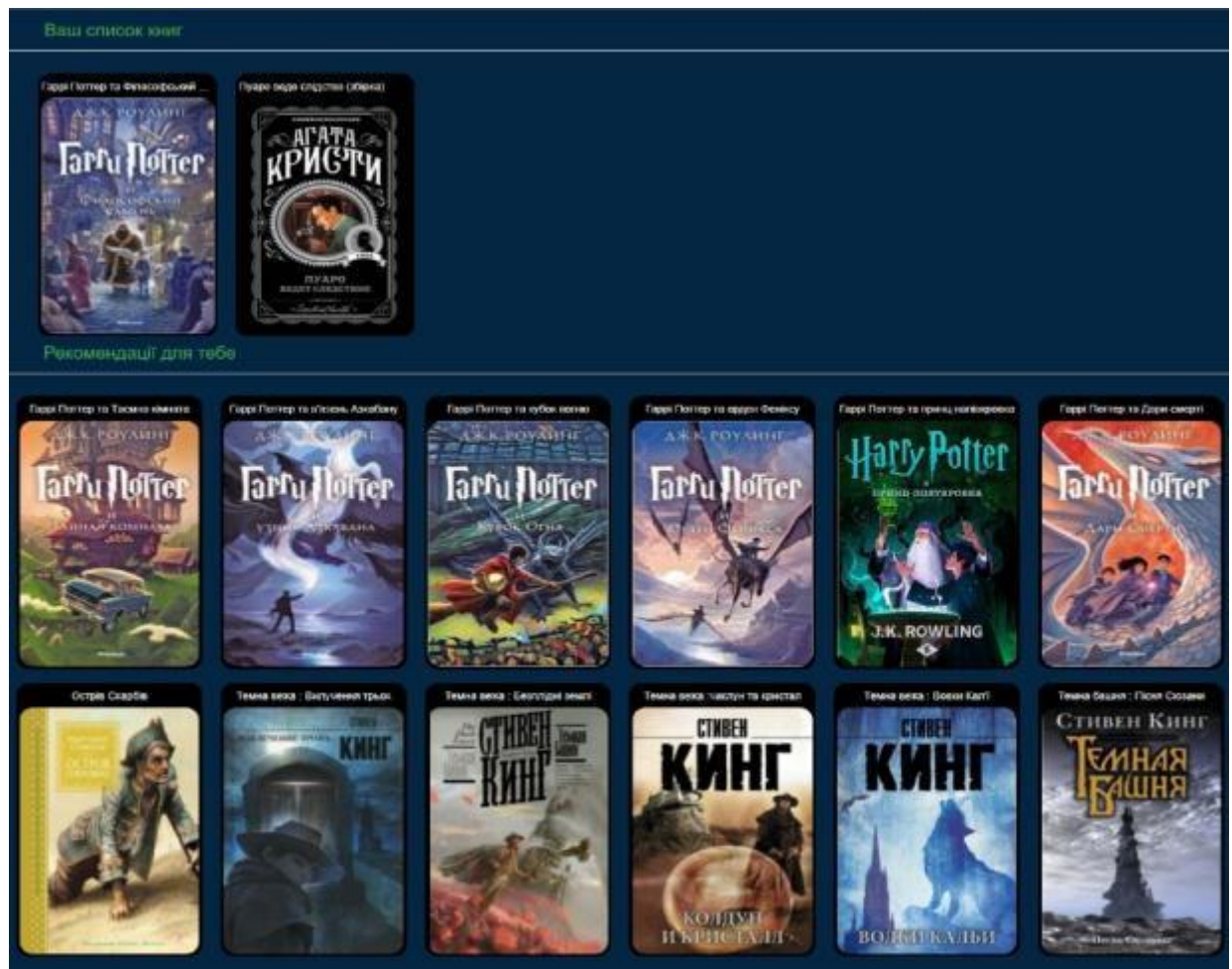


Рисунок 5.9 – Надання рекомендації на основі методу контентної фільтрації

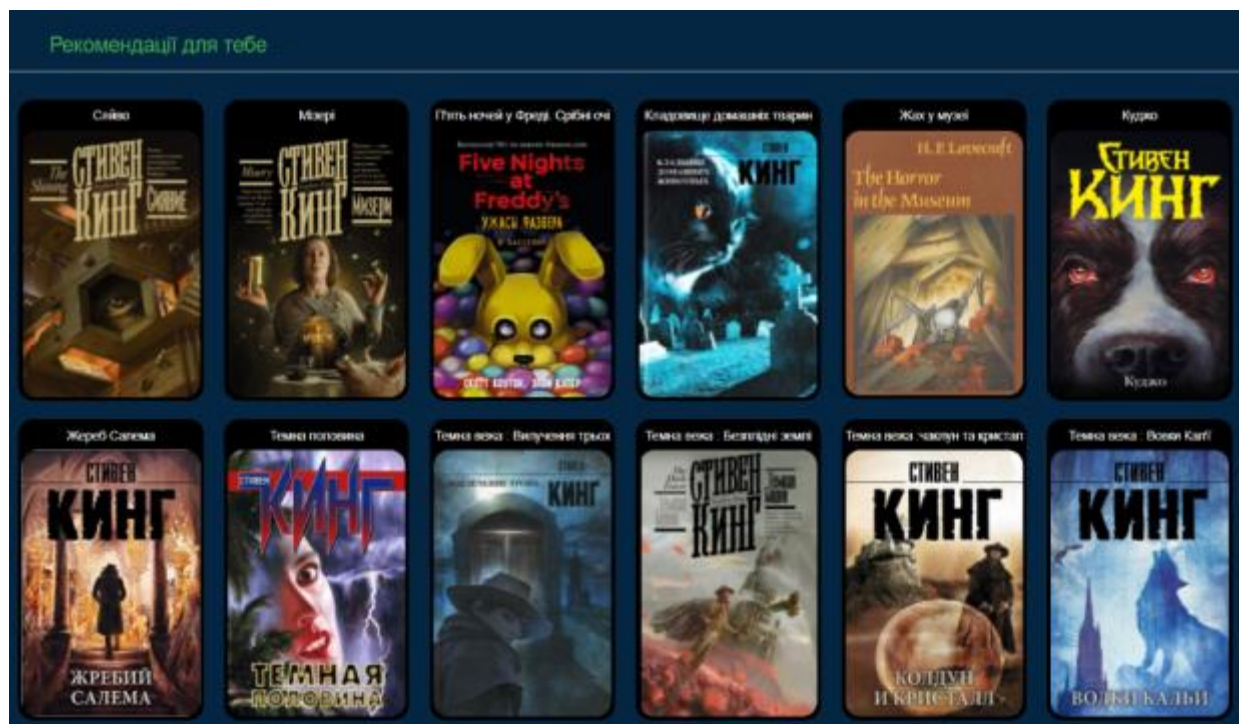


Рисунок 5.10 – Надання рекомендації на основі методу контентної фільтрації

5.2.3 Приклад надання рекомендації на основі гібридного методу фільтрації

Методи надання рекомендації на основі гібридного методу фільтрації, як зазначалось вище, базуються на поєднанні двох методів надання рекомендацій, на основі контенту та на основі оцінок інших користувачів. Гібридний метод фільтрації, розроблений для даної РС, присутній на сторінці детального перегляду книги. Надані рекомендації базуються на параметрах книги, що переглядається, та оцінках інших користувачів, що оцінили книги із схожими параметрами. На рисунках 5.11, 5.12, зображено рекомендації, надані на основі гібридного методу фільтрації., для різних книг.

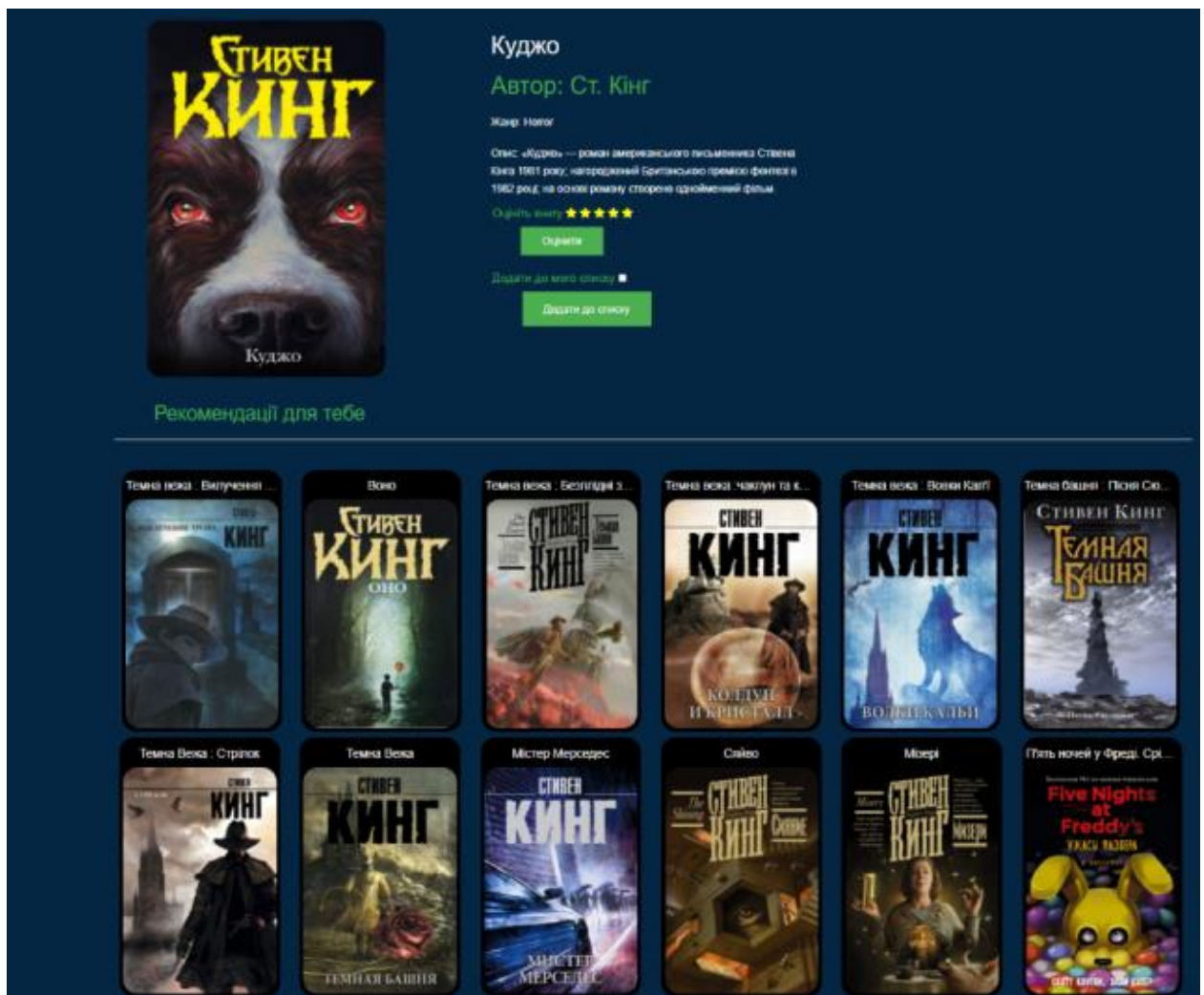


Рисунок 5.11 – Рекомендації, надані на основі гібридного методу фільтрації

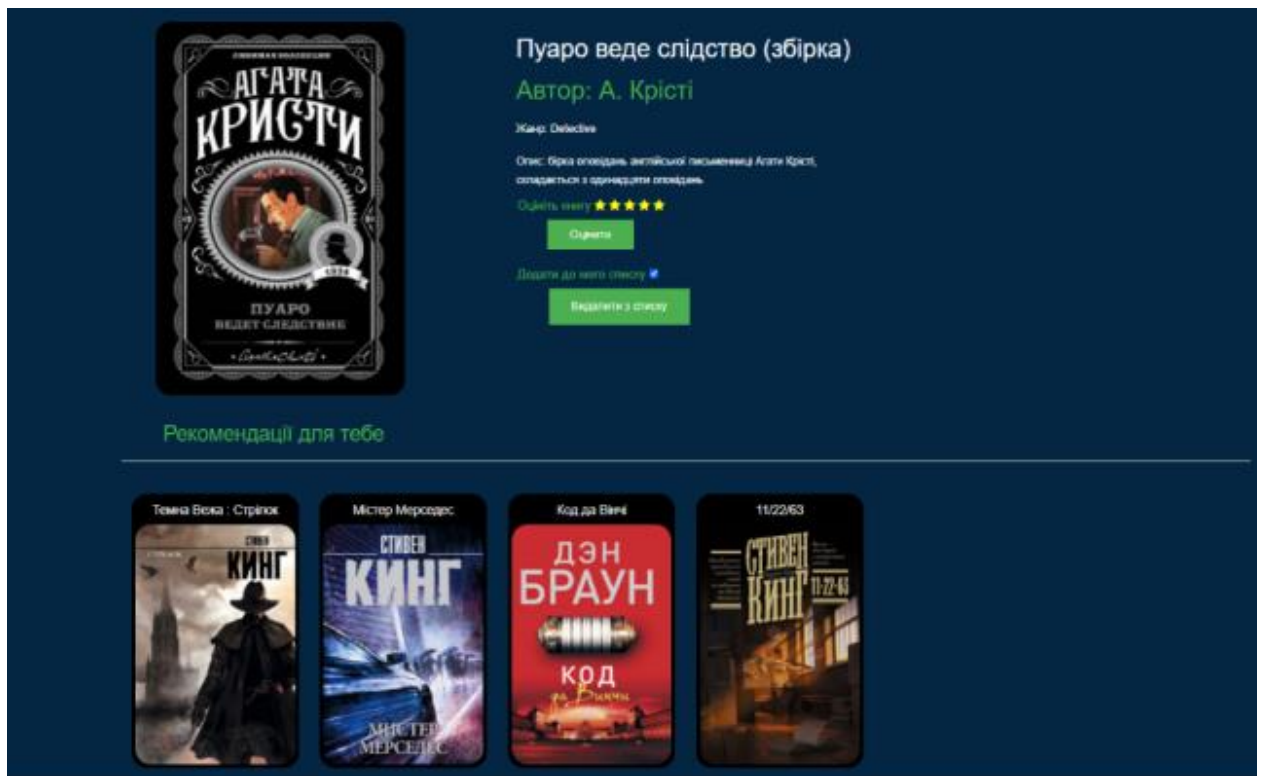


Рисунок 5.12 – Рекомендації, надані на основі гібридного методу фільтрації

5.3 Висновки за розділом 5

Було розглянуто інтерфейс розроблюваної рекомендаційної системи, а саме головну сторінку, сторінку реєстрації та авторизації, детальну сторінку книги. Було протестовано функції надання рекомендацій, на основі методу колаборативної фільтрації, на основі методу контентної фільтрації, на основі гібридного методу фільтрації.

ВИСНОВКИ

В ході роботи проведено аналіз предметної області, у результаті якого було сформовано ряд вимог до проєктування власної рекомендаційної системи.

Також було проаналізовано технології розробки і було обрано фреймворк Django. Це сучасний та прогресивний фреймворк, що користується популярністю серед розробників вебсайтів та має ряд переваг над іншими фреймворками. У роботі було досягнуто усіх вимог, що були висунуті, а саме надання рекомендацій різними способами, зручний інтерфейс, приємний дизайн.

Кінцевий проєкт є якісним продуктом, що не містить зайвої інформації, анімації, та функціоналу. Він швидко завантажується при різній якості Інтернет – з'єднання, завдяки прогресивній мові програмування Python, та простого дизайну та сучасним технологіям веброботи .

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методи оцінки якості роботи рекомендаційних систем [Електрон. ресурс]. – Режим доступу: <http://journals.nupp.edu.ua/sunz/article/view/1271/1071/>.
2. Дослідження методів оцінки рекомендаційних систем систем [Електрон. ресурс]. – Режим доступу: https://openarchive.nure.ua/bitstream/document/10953/1/2019_M_IUS_Sherbakov_O_E.pdf.
3. Попереднє вивчення системи рекомендацій [Електрон. ресурс]. – Режим доступу: <https://ithelp.ithome.com.tw/articles/10219033/>.
4. Content based approach algorithm [Електрон. ресурс]. – Режим доступу: [https://www.researchgate.net/figure/Content-based-approach-algorithm_fig1_326241366 /](https://www.researchgate.net/figure/Content-based-approach-algorithm_fig1_326241366/).
5. Колаборативна фільтрація [Електрон. ресурс]. – Режим доступу: [https://svitppt.com.ua/nauka/kolaborativna-filtraciya.html /](https://svitppt.com.ua/nauka/kolaborativna-filtraciya.html).
6. The Mathematics of Recommendation Systems [Електрон. ресурс]. – Режим доступу: <https://levelup.gitconnected.com/the-mathematics-of-recommendation-systems-e8922a50bdea/>.
7. A Hybrid Book Recommender System Based on Table of Contents (ToC) and Association Rule Mining [Електрон. ресурс]. – Режим доступу: [https://www.semanticscholar.org/paper/A-Hybrid-Book-Recommender-System-Based-on-Table-of-Ali-Khusro/d1015b01b8056331739ffba3c1be8c6eebb5795 /](https://www.semanticscholar.org/paper/A-Hybrid-Book-Recommender-System-Based-on-Table-of-Ali-Khusro/d1015b01b8056331739ffba3c1be8c6eebb5795/).
8. Рекомендаційна система товарів [Електрон. ресурс]. – Режим доступу: [http://ela.kpi.ua/bitstream/123456789/29220/1/Tarnavskyi_bakalavr.pdf?](http://ela.kpi.ua/bitstream/123456789/29220/1/Tarnavskyi_bakalavr.pdf)
9. Artificial Intelligence in Retail – 10 Present and Future Use Cases [Електрон. ресурс]. – Режим доступу: <https://emerj.com/ai-sector-overviews/artificial-intelligence-retail/>.
10. Chalyi S. Доповнення вхідних даних рекомендаційної системи в ситуації циклічного холодного старту з використанням темпоральних

обмежень типу «next» / S. Chalyi, V. Leshchynskyi, I. Leshchynska // Системи управління, навігації та зв'язку. Збірник наукових праць. – Полтава: ПНТУ, 2019. – Т. 4 (56). – С. 105-109.

11. Lu Y. Improvement of Text Feature Extraction with Genetic Algorithm / Y. Lu, and M. Liang. – New Technology of Library & Information Service, 2014. – p. 523-525.

12. Дослідження методів формування онлайн-рекомендацій в системах електронної комерції [Електрон. ресурс]. – Режим доступу: https://openarchive.nure.ua/bitstream/document/15002/1/2020_M_IUS_Kubota_D_D.pdf.

13. Wang L. H. An Improved Method of Short Text Feature Extraction Based on Words Co-Occurrence / L. H. Wang. – Applied Mechanics & Materials, 2014. – p. 842-845.

14. Методи аналізу геоданих для надання рекомендацій щодо розміщення громадських закладів у Києві [Електрон. ресурс]. – Режим доступу: http://ela.kpi.ua/bitstream/123456789/29264/1/Trokhymovych_bakalavr.pdf?.

15. Метрики якості ранжування [Електрон. ресурс]. – Режим доступу: <https://habr.com/ru/company/econtenta/blog/303458/>.

16. Підручник мови Python [Електрон. ресурс]. – Режим доступу: <https://uk.wikibooks.org/>.

17. Що таке CMS и Які види CMS для сайтів бувають [Електрон. ресурс]. – Режим доступу: <https://cityhost.ua/uk/blog/chto-takoe-cms-i-kakie-vidy-cms-dlya-saytov-byvayut.html/>.

18. Django [Електрон. ресурс]. – Режим доступу: <https://uk.wikipedia.org/>.

19. Django документація [Електрон. ресурс]. – Режим доступу: <https://djbook.ru/>.

20. DJANGO PROJECT MVT СТРУКТУРА [Електрон. ресурс]. – Режим доступу: <http://espressocode.top/django-project-mvt-structure/>.

21. Basic MVC Architecture [Електрон. ресурс]. – Режим доступу: https://www.tutorialspoint.com/struts_2/basic_mvc_architecture.htm.
22. Чому Django найкращий фреймворк для розробки сайтів [Електрон. ресурс]. – Режим доступу: [https://ru.hexlet.io/blog/posts/pochemu-django-luchshiy-freymvork-dlya-razrabotki-saytov /](https://ru.hexlet.io/blog/posts/pochemu-django-luchshiy-freymvork-dlya-razrabotki-saytov/).
23. Документація по фреймворку Django [Електрон. ресурс]. – Режим доступу: <https://django.fun/docs/django/ru/3.1/topics/db/models/>.

ДОДАТОК А
Технічне завдання

Вступ

Програма призначена для надання рекомендацій на основі різних методів фільтрації.

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Дослідження та програмна реалізація рекомендаційної системи підбору літератури», затверджене наказом № 363 від 04 листопада 2022 року по Національному університету «Запорізька політехніка».

A.2 Призначення розробки

Програма призначена для використання людьми, що потребують отримати рекомендації книг в Інтернеті.

A.3 Вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Функції, які мають виконуватись програмним продуктом:

- надавати рекомендації на основі методу колаборативної фільтрації.
- надавати рекомендації на основі методу контентної фільтрації.
- надавати рекомендації на основі гібридної методу фільтрації.

A.3.2 Вимоги до надійності

Програма має забезпечувати коректну роботу своїх модулів.

A.3.3 Умови експлуатації

Для роботи з програмою користувач має володіти достатнім рівнем користування персональним комп'ютером.

A.3.4 Вимоги до складу й параметрів технічних засобів

Для роботи необхідний персональний комп'ютер з версією операційної системи Microsoft Windows 10.

A.4 Вимоги до програмної документації

Програмна документація включає в себе технічне завдання і повинна бути написана українською мовою.

A.5 Порядок контролю й приймання

Робота допускається до захисту керівником, нормоконтролером, затверджується завідувачем кафедри і захищається в присутності членів екзаменаційної комісії.

ДОДАТОК Б
Текст програми

Б.1 Файл settings.py

```

import os
# from movie_recommender.aws.conf import *
# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '40_$n*hs61m(shf^y0q6&th7i^b!t1rw6k-458_0)ed34ji73v'
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
ALLOWED_HOSTS = ['awesome-books-recommender.herokuapp.com', '127.0.0.1']
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'storages',
    'recommend'
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'whitenoise.middleware.WhiteNoiseMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'books_recommender.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

```

```

]

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/3.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles/')
STATIC_URL = '/static/'
STATICFILES_DIRS = ([
    os.path.join(BASE_DIR, "static"),
])
# STATICFILES_DIRS = ()

MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'

```

Б.2 Файл (main) urls.py

```

from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path("", include('recommend.urls')),
]

if settings.DEBUG:
    urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

Б.3 Файл views.py

```

from django.contrib.auth import authenticate, login
from django.contrib.auth import logout
from django.shortcuts import render, get_object_or_404, redirect
from .forms import *
from django.http import Http404
from .models import Movie, Myrating, MyList, HashTag
from django.db.models import Q
from django.contrib import messages
from django.http import HttpResponseRedirect
from django.db.models import Case, When
from django.views.generic.list import ListView
import pandas as pd

# Create your views here.

def index(request):
    movies = Movie.objects.all()

    query = request.GET.get('q')

    if query:
        movies = Movie.objects.filter(Q(title__icontains=query)).distinct()

    return render(request, 'recommend/list.html', {'movies': movies})

    return render(request, 'recommend/list.html', {'movies': movies})

def filtre(request):

```

```

# return HttpResponse('Hello World')
books = Movie.objects.all()
tags = HashTag.objects.all()

return render(request, 'recommend/genres.html',
              {'books': books, 'tags': tags })
# Show details of the movie
def detail(request, movie_id):
    if not request.user.is_authenticated:
        return redirect("login")
    if not request.user.is_active:
        raise Http404
    movies = get_object_or_404(Movie, id=movie_id)
    movie = Movie.objects.get(id=movie_id)

    temp = list(MyList.objects.all().values().filter(movie_id=movie_id,user=request.user))
    if temp:
        update = temp[0]['watch']
    else:
        update = False
    if request.method == "POST":

        # For my list
        if 'watch' in request.POST:
            watch_flag = request.POST['watch']
            if watch_flag == 'on':
                update = True
            else:
                update = False
            if MyList.objects.all().values().filter(movie_id=movie_id,user=request.user):
MyList.objects.all().values().filter(movie_id=movie_id,user=request.user).update(watch=update)
            else:
                q=MyList(user=request.user,movie=movie,watch=update)
                q.save()
            if update:
                messages.success(request, "Movie added to your list!")
            else:
                messages.success(request, "Movie removed from your list!")

        # For rating
        else:
            rate = request.POST['rating']
            if Myrating.objects.all().values().filter(movie_id=movie_id,user=request.user):
Myrating.objects.all().values().filter(movie_id=movie_id,user=request.user).update(rating=rate)
            else:
                q=Myrating(user=request.user,movie=movie,rating=rate)
                q.save()

```

```

        messages.success(request, "Rating has been submitted!")

    return HttpResponseRedirect(request.META.get('HTTP_REFERER'))
    out = list(Myrating.objects.filter(user=request.user.id).values())

# To display ratings in the movie detail page
movie_rating = 0
rate_flag = False
for each in out:
    if each['movie_id'] == movie_id:
        movie_rating = each['rating']
        rate_flag = True
        break

context = {'movies': movies, 'movie_rating': movie_rating, 'rate_flag': rate_flag, 'update': update}
return render(request, 'recommend/detail.html', context)

# MyList functionality
def watch(request):

    if not request.user.is_authenticated:
        return redirect("login")
    if not request.user.is_active:
        raise Http404

    movies = Movie.objects.filter(mylist__watch=True, mylist__user=request.user)
    query = request.GET.get('q')
    genres = Movie.objects.all()

    if query:
        movies = Movie.objects.filter(Q(title__icontains=query)).distinct()
        return render(request, 'recommend/watch.html', {'movies': movies})

    return render(request, 'recommend/watch.html', {'movies': movies, 'genres': genres})

# To get similar movies based on user rating
def get_similar(movie_name, rating, corrMatrix):
    similar_ratings = corrMatrix[movie_name]*(rating-2.5)
    similar_ratings = similar_ratings.sort_values(ascending=False)
    return similar_ratings

# Recommendation Algorithm
def recommend(request):

    if not request.user.is_authenticated:
        return redirect("login")
    if not request.user.is_active:
        raise Http404

```

```

movie_rating=pd.DataFrame(list(Myrating.objects.all().values()))

new_user=movie_rating.user_id.unique().shape[0]
current_user_id= request.user.id
    # if new user not rated any movie
if current_user_id>new_user:
    movie=Movie.objects.get(id=19)
    q=Myrating(user=request.user,movie=movie,rating=0)
    q.save()

userRatings =
movie_rating.pivot_table(index=['user_id'],columns=['movie_id'],values='rating')
userRatings = userRatings.fillna(0,axis=1)
corrMatrix = userRatings.corr(method='pearson')

user =
pd.DataFrame(list(Myrating.objects.filter(user=request.user).values())).drop(['user_id','id'],axis=
1)
user_filtered = [tuple(x) for x in user.values]
movie_id_watched = [each[0] for each in user_filtered]

similar_movies = pd.DataFrame()
for movie,rating in user_filtered:
    similar_movies =
similar_movies.append(get_similar(movie,rating,corrMatrix),ignore_index = True)

movies_id = list(similar_movies.sum().sort_values(ascending=False).index)
movies_id_recommend = [each for each in movies_id if each not in movie_id_watched]
preserved = Case(*[When(pk=pk, then=pos) for pos, pk in
enumerate(movies_id_recommend)])
movie_list=list(Movie.objects.filter(id__in =
movies_id_recommend).order_by(preserved)[:10])

context = {'movie_list': movie_list}
return render(request, 'recommend/recommend.html', context)

# Register user
def signUp(request):
    form = UserForm(request.POST or None)

    if form.is_valid():
        user = form.save(commit=False)
        username = form.cleaned_data['username']
        password = form.cleaned_data['password']
        user.set_password(password)
        user.save()
        user = authenticate(username=username, password=password)

    if user is not None:
        if user.is_active:

```

```

        login(request, user)
        return redirect("index")

context = {'form': form}

return render(request, 'recommend/signUp.html', context)

# Login User
def Login(request):
    if request.method == "POST":
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(username=username, password=password)

        if user is not None:
            if user.is_active:
                login(request, user)
                return redirect("index")
            else:
                return render(request, 'recommend/login.html', {'error_message': 'Your account
disable'})
        else:
            return render(request, 'recommend/login.html', {'error_message': 'Invalid Login'})

    return render(request, 'recommend/login.html')

# Logout user
def Logout(request):
    logout(request)
    return redirect("login")

```

Б.4 Файл models.py

```

from django.db import models
from django.core.validators import MaxValueValidator, MinValueValidator
from django.contrib.auth.models import User

class Books(models.Model):
    title = models.CharField(max_length=200)
    genre = models.CharField(max_length=100)
    author = models.CharField(max_length=140, default='SOME STRING')
    tags = models.CharField(max_length=140, default='#')
    tagsSize = models.CharField(max_length=140, default='#')
    genreSecond = models.CharField(max_length=50, default='SOME STRING')
    genreThird = models.CharField(max_length=50, default='SOME STRING')
    Books_logo = models.FileField()

```

```
class Myrating(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    books = models.ForeignKey(Books, on_delete=models.CASCADE)
    rating = models.IntegerField(default=0, validators=[MaxValueValidator(5),
MinValueValidator(0)])
```

```
class MyList(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    books_list = models.ForeignKey(Books, on_delete=models.CASCADE)
    watch = models.BooleanField(default=False)
```

Б.5 Файл urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path("", views.index, name='index'),
    path('signup/', views.signUp, name='signup'),
    path('login/', views.Login, name='login'),
    path('logout/', views.Logout, name='logout'),
    path('<int:movie_id>/', views.detail, name='detail'),
    path('watch/', views.watch, name='watch'),
    path('recommend/', views.recommend, name='recommend'),
    path('genres/', views.filtre, name='genres'),
]
```

Б.6 Файл forms.py

```
from django.contrib.auth.models import User
from django import forms
from .models import *

class UserForm(forms.ModelForm):
    password = forms.CharField(widget=forms.PasswordInput)

    class Meta:
        model = User
        fields = ['username', 'email', 'password']
```

Б.7 Файл admin.py

```
from django.contrib import admin
from .models import Movie, Myrating, MyList

# Register your models here.
admin.site.register(Movie)
admin.site.register(Myrating)
admin.site.register(MyList)
```

Б.8 Файл base.html

```
<!DOCTYPE html>
<html>
<head>

    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

    <title>{% block title %}Movies{% endblock title %}</title>

    {% load static %}
    <link rel="stylesheet" type="text/css" href="{% static
'recommend/css/bootstrap.min.css"%}">
    <link rel='stylesheet' href='https://maxcdn.bootstrapcdn.com/font-
awesome/4.5.0/css/font-awesome.min.css' >
    <link href='http://fonts.googleapis.com/css?family=Open+Sans:400,300,700' rel='stylesheet'
type='text/css'>
    <link rel='stylesheet' href='{% static "recommend/css/base.css" %}' />

    <style type="text/css">
        element.style{
            background-color:#072d4e;
        }
        .thumbnail p, .thumbnail h4 {
            white-space: nowrap;
            text-overflow: ellipsis;
            overflow: hidden;
        }
        .star-rating {
            line-height:32px;
            font-size:1.25em;
            margin-left:-33px;
        }
        .star-rating .fa-star{
            color: yellow;
```

```

    }
    .heart-rating {
        line-height:32px;
        font-size:1.25em;
        margin-left:-33px;
    }
    .heart-rating .fa-heart{
    color:yellow;
    }
    .button {
        background-color: #4CAF50;
        border: none;
        color: white;
        padding: 10px 32px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 16px;
        margin: 4px 4px;
        cursor: pointer;
        margin-left:45px;
    }
    .button1 {
        background-color: #4CAF50;
        border: none;
        color: white;
        padding: 12px 30px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 16px;
        margin: -4px -4px;
        cursor: pointer;
        margin-left:4px;
    }

</style>
</head>

<body style="background-color:#052642">
    <nav class="navbar navbar-inverse">
    <div class="container-fluid">
        <div class="collapse navbar-collapse" id="topNavBar">

            <ul class="nav navbar-nav ">
                <li><a style="color:#4CAF50" href="/">Books-Recommender</a></li>
            </ul>
            <ul class="nav navbar-nav ">
                <li><a href="{% url 'index' %}">&nbsp; Home</a></li>
            </ul>

            <ul class="nav navbar-nav navbar-right">

```

```

        <li>
            <a href="{% url 'watch' %}">
                <span aria-hidden="true"></span>MyList
            </a>
        </li>
    {% if not request.user.is_authenticated %}
        <li>
            <a href="{% url 'signup' %}">
                <span aria-hidden="true"></span>SignUp
            </a>
        </li>
        <li>
            <a href="{% url 'login' %}">
                <span aria-hidden="true"></span>Login
            </a>
        </li>
    {% else %}
        <li>
            <a href="{% url 'logout' %}">
                <span aria-hidden="true"></span>&nbsp;Logout
            </a>
        </li>
    {% endif %}
</ul>

</div>
</div>
</nav>
</div>
{% block body %}
{% endblock %}

<script type="text/javascript" src="{%static 'recommend/js/jquery.min.js'%}">
</script>

<script type="text/javascript" src="{%static 'recommend/js/bootstrap.min.js'%}">
</script>
<script type="text/javascript">
var $star_rating = $('star-rating .fa');

var SetRatingStar = function() {
    return $star_rating.each(function() {
        if (parseInt($star_rating.siblings('input.rating-value').val()) >=
parseInt($(this).data('rating'))) {
            return $(this).removeClass('fa-star-o').addClass('fa-star');
        } else {
            return $(this).removeClass('fa-star').addClass('fa-star-o');
        }
    });
};

```

```

$star_rating.on('click', function() {
$star_rating.siblings('input.rating-value').val($(this).data('rating'));
return SetRatingStar();
});

SetRatingStar();
$(document).ready(function() {

});

</script>
<script type="text/javascript">
function validateForm(){
var x = document.forms["ratingForm"]["rating"].value;
if(x=="0"){
alert("Invalid Input");
return false;
}
}
}
</script>

</body>
</html>

```

Б.9 Файл detail.html

```

{% extends 'recommend/base.html'%}
{% block title %}{{ book.title }}{% endblock %}

{% block body %}
{% if messages %}
<ul class="messages">
    {% for message in messages %}
    <div class="alert alert-success">
        <strong>Success!</strong> {{ message }}
    </div>
    {% endfor %}
</ul>
{% endif %}

<div class="container-fluid col-sm-offset-1">

    <div class="row">
        <!-- Left Movie Info -->
        <div class="col-sm-2 col-md-3">
<!--         <div class="panel panel-default">-->
<!--         <div class="panel-body" style="border-radius: 22px 22px 22px 22px;">-->

```



```

    <div class="star-rating">
      <span>Add to my list </span>
      {% if update %}
      <input type="checkbox" name="watch" checked>
      <input type="submit" class="button" value="Remove" name="watch">
      {% else %}
      <input type="checkbox" name="watch">
      <input type="submit" class="button" value="Add">

      {% endif %}
    </div>
  </div>
</div>
<!-- <input type="submit" class="button" value="Add">-->
</div>
</div>
</form>

</div>
</div>
</div>
</div>

{% endblock %}

```

Б.10 Файл watch.html

```

{% extends 'recommend/base.html'%}
{% block body %}

<h2>Ваш список книг</h2>
<hr>
<br>
<!-- Movie list -->
<div class="row">
  {% if movies %}
  {% for movie in movies %}
    <div class="col-sm-2 col-md-2 ">
      <div class="thumbnail" style="border-radius: 22px 22px 22px 22px;">
        <h4>{{ movie.title }}</h4>
        <a href="{% url 'detail' movie.id %}">
          
        </a>
        <!-- <h5>{{ movie.genre }}</h5> -->
      </div>
    </div>

  {% endfor %}

```

```

{% else %}
  <div class="alert alert-danger">
    <strong>Sorry!</strong> No movies in your list.
  </div>
{% endif %}

</div>
<h2>Рекомендації для тебе</h2>
<hr>
<br>
{% if movies %}
  {% for movie in movies %}

    {% for genre in genres %}
      {% if genre.genre == movie.genre and genre.title != movie.title %}

        <div class="col-sm-2 col-md-2">
          <div class="thumbnail" style="border-radius: 22px 22px 22px 22px;">
            <center><h4>{{ genre.title }}</h4></center>
            <a href="{% url 'detail' genre.id %}">
              
            </a>
            <!-- <h5>{{ movie.genre }}</h5> -->
          </div>
        </div>
      {% endif %}
    {% endfor %}
  {% endfor %}
{% endif %}

{% endblock %}

```

Б.11 Файл recommend.html

```

{% extends 'recommend/base.html'%}
{% block title %}Recommendations{% endblock %}
{% block body %}

<div class="container-fluid">
  <h2>Recommendations for you</h2>
</div>
<hr>
<br>

  <div class="container-fluid">

    <!-- Movie list -->
    <div class="row">
      {% if movie_list %}

```

```

        {% for movie in movie_list %}
        <div class="col-sm-2 col-md-2 ">
            <div class="thumbnail" style="border-radius: 22px 22px 22px 22px;">
                <center><h4>{{ movie.title }}</h4></center>
                <a href="{% url 'detail' movie.id %}">
                    
                </a>
            <!--
                <center><h5>{{ movie.genre }}</h5></center>-->
            </div>
        </div>
        {% endfor %}
    {% endif %}
</div>

</div>

{% endblock %}

```

Б.12 Файл form_template.html

```

{% for field in form %}
    <div class="form-group">
        <div class="col-sm-offset-2 col-sm-12">
            <span class="text-danger small">{{ field.errors }}</span>
        </div>
        <label class="control-label col-sm-2" for="movie_title">{{ field.label_tag }}</label>
        <div class="col-sm-10">{{ field }}</div>
    </div>
{% endfor %}

```

Б.13 Файл list.html

```

{% extends 'recommend/base.html'%}
{% block body %}

    {% if not request.user.is_authenticated %}
        <br>
    {% else %}
        <center style="color:#fff"><h2>Welcome {{ user.username }}!!</h2></center>
    {% endif %}

    <div class="container" >
        <div class="col-xs-6 col-xs-offset-3">
            <form class="form-horizontal" role="search" method="get" action="">
                <div class="form-group">
                    <div class="input-group">

```

```

        <input type="text" class="form-control input-lg " name="q" value="{{
request.GET.q }}" placeholder= "Search Book "/>
        <span class='input-group-btn'>
            <button class='button1' type='submit'>Search</button>
        </span>
    </div>
</div>
</form>
</div>
</div>

<p style="text-align:left;">
    <span style="float:right;">
        <a href="{% url 'recommend' %}" class="btn"><button style="color:#fff;background-
color:#4CAF50;padding:10px">Get Books Recommendation</button></a>
        <a href="{% url 'genres' %}" class="btn"><button style="color:#fff;background-
color:#4CAF50;padding:10px">Genres</button></a>
        <h2>Books List</h2>
    </span>
</p>
<hr>
<br>

<!-- Movie list -->
<div class="row" >
    {% if movies %}
    {% for movie in movies %}

        <div class="col-sm-2 col-md-2">
            <div class="thumbnail" style="border-radius: 22px 22px 22px 22px;">
                <center><h4>{{ movie.title }}</h4></center>
                <a href="{% url 'detail' movie.id %}">
                    
                </a>
                <!-- <h5>{{ movie.genre }}</h5> -->
            </div>
        </div>

        {% endfor %}
    {% endif %}
</div>

</div>

{% endblock %}

```



```

        </a>
        <!-- <h5>{{ movie.genre }}</h5> -->
    </div>
</div>
{% endif %}
{% endfor %}
</div>

<div class="row" >
    <p style="text-align:left;">
    <span style="float:right;">
    <h2 id="Adventure">Adventure</h2>
    </span>
</p>
<hr>
<br>
    {% for book in books %}

        {% if book.genre == "Fantasy, Adventure"%}
        <div class="col-sm-2 col-md-2">
            <div class="thumbnail" style="border-radius: 22px 22px 22px 22px;">
                <center><h4>{{ book.title }}</h4></center>
                <a href="{% url 'detail' book.id %}">
                    
                </a>
                <!-- <h5>{{ movie.genre }}</h5> -->
            </div>
        </div>
        {% endif %}
    {% endfor %}

</div>

</div>

{% endblock %}

```

Б.15 Файл login.html

```

{% extends 'recommend/base.html'%}
{% block title %}Log In{% endblock %}

{% block body %}
<div class="container-fluid">

    <div class="row">
        <div class="col-sm-12 col-md-6 col-md-offset-3">
            <div class="panel panel-default">

```

```

<div class="panel-body">
  <h3>Log In</h3>
  {% if error_message %}
    <p><strong>{{ error_message }}</strong></p>
  {% endif %}
  <form class="form-horizontal" role="form" action="#" method="post"
enctype="multipart/form-data">
    {% csrf_token %}
    <div class="form-group">
      <label class="control-label col-sm-2" for="id_username">
        Username:
      </label>
      <div class="col-sm-10">
        <input id="id_username" maxlength="30" name="username" type="text">
      </div>
    </div>
    <div class="form-group">
      <label class="control-label col-sm-2" for="id_password">
        Password:
      </label>
      <div class="col-sm-10">
        <input id="id_password" maxlength="30" name="password"
type="password">
      </div>
    </div>
    <div class="form-group">
      <div class="col-sm-offset-2 col-sm-10">
        <button type="submit" class="btn btn-success">Submit</button>
      </div>
    </div>
  </form>
</div>
<div class="panel-footer">
  Don't have an account? <a href="{% url 'signup' %}">Click here</a> to register.
</div>
</div>
</div>

{% endblock %}

```

Б.16 Файл singUp.html

```

{% extends 'recommend/base.html'%}
{% block title %}Register{% endblock %}
{% block body %}
<div class="container-fluid">

```

```

<div class="row">
  <div class="col-sm-10 col-md-6 col-md-offset-3">
    <div class="panel panel-default">
      <div class="panel-body">
        <h3>Register for an Account</h3>
        {% if error_message %}
          <p><strong>{{ error_message }}</strong></p>
        {% endif %}
        <form class="form-horizontal" role="form" action="" method="post"
enctype="multipart/form-data">
          {% csrf_token %}
          {% include 'recommend/form_template.html' %}
          <div class="form-group">
            <div class="col-sm-offset-2 col-sm-10">
              <button type="submit" class="btn btn-success">Submit</button>
            </div>
          </div>
        </form>
      </div>
      <div class="panel-footer">
        Already have an account? <a href="{% url 'login' %}">Click here</a> to log in.
      </div>
    </div>
  </div>
</div>

{% endblock %}

```

ДОДАТОК В
Слайди презентації

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«ЗАПОРІЗЬКА ПОЛІТЕХНІКА»
КАФЕДРА ПРОГРАМНИХ ЗАСОБІВ

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ РЕКОМЕНДАЦІЙНОЇ
СИСТЕМИ ПІДБОРУ ЛІТЕРАТУРИ

Виконав
ст. гр. КНТ-111М

В.С.Сопін

► Керівник :
► доцент

Т.О.Колпакова

1

Рисунок В.1 – Слайд 1

Об'єкт , предмет, мета роботи

- Об'єкт дослідження – рекомендаційні системи.
- Предмет дослідження – вебсервіс рекомендаційної системи підбору літератури.
- Мета роботи – дослідити та програмно реалізувати рекомендаційну систему підбору літератури.

2

Рисунок В.2 – Слайд 2

Завдання роботи

Для досягнення поставленої мети необхідно було розв'язати такі завдання :

- ▶ Виконати аналіз предметної області.
- ▶ Проаналізувати та порівняти актуальні методи фільтрації даних.
- ▶ Розробити рекомендаційну систему з можливістю надавати рекомендації на основі кількох методів фільтрації даних.
- ▶ Виконати тестування розробленої рекомендаційної системи .

3

Рисунок В.3 – Слайд 3

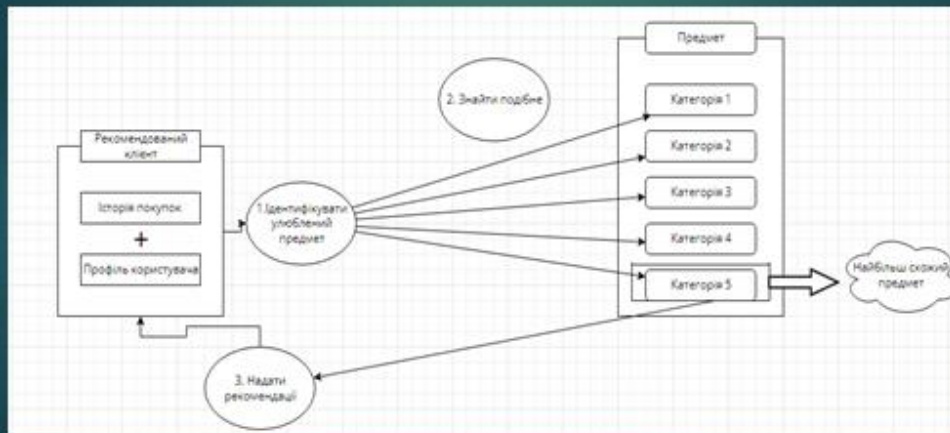
Схема існуючих підходів для надання рекомендацій



4

Рисунок В.4 – Слайд 4

Схема методу контентної фільтрації (Content-based)



5

Рисунок В.5 – Слайд 5

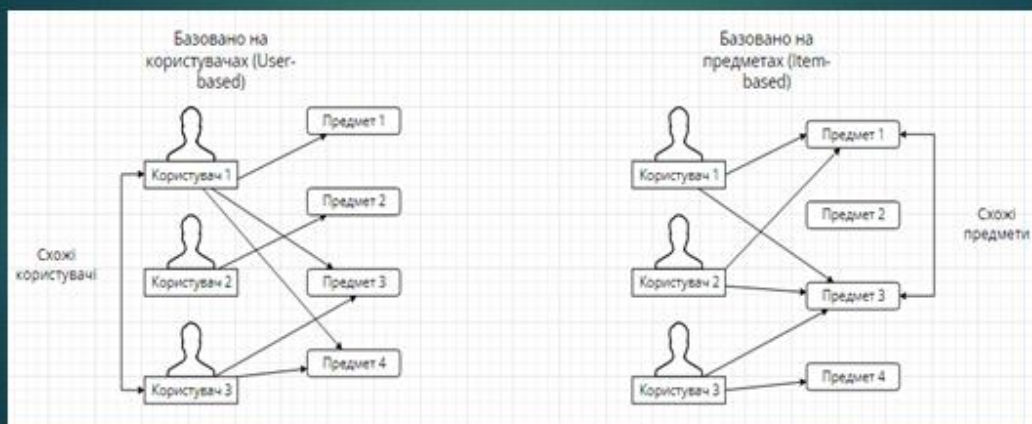
Схема методу колаборативної фільтрації



6

Рисунок В.6 – Слайд 6

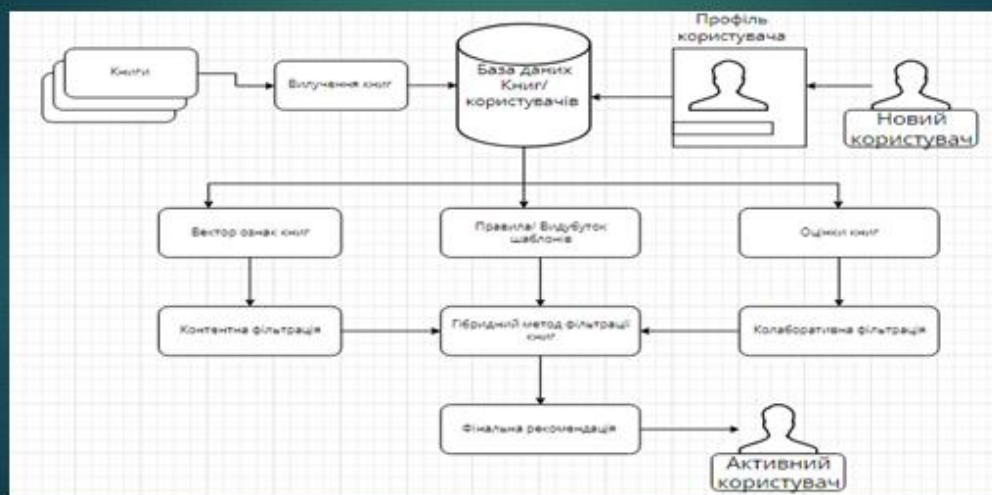
Алгоритм роботи user-based та item-based рекомендацій



7

Рисунок В.7 – Слайд 7

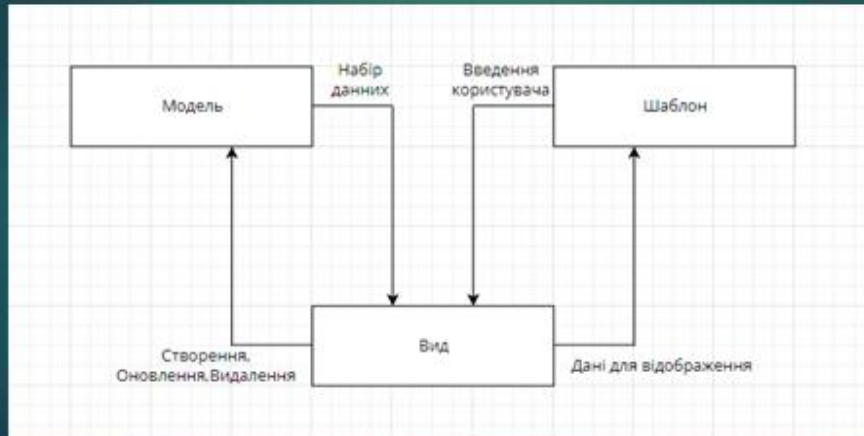
Схема роботи гібридної системи фільтрації



8

Рисунок В.8 – Слайд 8

Схема роботи Model View Template



9

Рисунок В.9 – Слайд 9

Структури даних проєкту

```

class Books(models.Model):
    title = models.CharField(max_length=200)
    genre = models.CharField(max_length=100)
    author = models.CharField(max_length=140, default='SOME STRING')
    tags = models.CharField(max_length=140, default='x')
    tagsSize = models.CharField(max_length=140, default='x')
    genreSecond = models.CharField(max_length=50, default='SOME STRING')
    genreThird = models.CharField(max_length=50, default='SOME STRING')
    Books_logo = models.FileField()

class Myrating(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    books = models.ForeignKey(Books, on_delete=models.CASCADE)
    rating = models.IntegerField(default=0, validators=[MaxValueValidator(5), MinValueValidator(0)])

class Mylist(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    books_list = models.ForeignKey(Books, on_delete=models.CASCADE)
    watch = models.BooleanField(default=False)
  
```

10

Рисунок В.10 – Слайд 10

Рекомендації, надані на основі гібридного методу фільтрації.



13

Рисунок В.13 – Слайд 13

Висновок

- ▶ В ході роботи проведено аналіз предметної області, у результаті якого було сформовано ряд вимог до проєктування власної рекомендаційної системи.
- ▶ Також було проаналізовано технології розробки і було обрано фреймворк Django. Це сучасний та прогресивний фреймворк, що користується популярністю серед розробників вебсайтів та має ряд переваг над іншими фреймворками. У роботі було досягнуто усіх вимог, що були висунуті, а саме надання рекомендацій різними способами, зручний інтерфейс, приємний дизайн.
- ▶ Кінцевий проєкт є якісним продуктом, що не містить зайвої інформації, анімації, та функціоналу. Він швидко завантажується при різній якості Інтернет – з'єднання, завдяки прогресивній мові програмування Python, та простого дизайну та сучасним технологіям веброзробки.

14

Рисунок В.14 – Слайд 14