Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розробка рекомендаційної системи для підбору фільмів, книг та ігор на основі аналізу користувацьких запитів
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-22-4
Софія БЕДА
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник ас. Максим ПОЛІЇТ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Лариса ЧАЛА
(підпис) (власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Беді Софії Іванівні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка рекомендаційної системи для підбору фільмів, книг та ігор на основі аналізу користувацьких запитів

затверджена наказом університету від 5 червня 20 26 р. № 488Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 8 липня 20 26 р.

3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел та відомих наукових проєктів, документація FastAPI, OpenRouter API, TMDB API, Google Books API та RAWG API, набір даних для тестування системи у вигляді природномовних запитів різних типів.

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та постановка задачі дослідження
2) Методи та технологій побудови рекомендаційних систем
3) Проєктування та програмна реалізація рекомендаційної системи

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.06.2026	виконано
2	Аналіз предметної галузі рекомендаційних	10.06.2026	виконано
	систем мультимедійного контенту		
3	Дослідження методів побудови рекомендаційних	11.06.2026	виконано
	систем та семантичного аналізу тексту		
4	Проєктування архітектури рекомендаційної	13.06.2026	виконано
	системи		
5	Реалізація серверної частини системи	14.06.2026	виконано
6	Реалізація модуля аналізу запитів на основі LLM	16.06.2026	виконано
	та розробка системних промптів		
7	Інтеграція з зовнішніми API: TMDB,	16.06.2026	виконано
	Google Books, RAWG		
8	Реалізація клієнтської частини системи та	16.06.2026	виконано
	користувацького інтерфейсу		
9	Тестування та налагодження системи	17.06.2026	виконано
10	Оформлення пояснювальної записки	20.06.2026	виконано
11	Захист перед ЕК	08.07.2026	

Дата видачі завдання 13 квітня 20 26 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Максим ПОЛІЇТ
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 75 с., 38 рис., 6 табл., 1 дод., 26 джерел.

ВІДЕОІГРИ, КНИГИ, МЕДІАКОНТЕНТ, РЕКОМЕНДАЦІЙНА СИСТЕМА, СЕМАНТИЧНИЙ АНАЛІЗ, ФІЛЬМИ, SENTENCE TRANSFORMERS.

Об’єкт дослідження – процес формування персоналізованих рекомендацій мультимедійного контенту для користувачів цифрових платформ.

Предмет дослідження – методи та засоби побудови інтелектуальної системи семантичних рекомендацій на основі NLP-технологій, текстових ембедингів та аналізу поведінкових даних користувача.

Мета роботи – розробка практичної реалізація інтелектуальної системи семантичних рекомендацій медіаконтенту, що забезпечує персоналізовані рекомендації фільмів, книг та відеоігор на основі аналізу природномовних запитів користувача.

Методи дослідження – теоретичний, експериментальний. Методи розробки базуються на технології текстових ембедингів Sentence Transformers, мірі косинусної схожості для семантичного порівняння, а також методах контентної фільтрації.

У бакалаврській роботі досліджено сучасні підходи до побудови рекомендаційних систем та методи семантичного аналізу природної мови. Проведено аналіз контентної, колаборативної та гібридної фільтрації, розглянуто застосування трансформерних моделей і великих мовних моделей для формування персоналізованих рекомендацій. Розроблено веборієнтовану рекомендаційну систему для підбору фільмів, книг та відеоігор на основі природномовних запитів користувача.

ABSTRACT

Bachelor’s thesis contains: 75 pp., 38 fig., 6 tabl., 1 ann., 26 references.

BOOKS, MEDIA CONTENT, MOVIES, RECOMMENDER SYSTEM, SEMANTIC ANALYSIS, SENTENCE TRANSFORMERS, VIDEO GAMES.

Object of research – the process of generating personalized multimedia content recommendations for users of digital platforms.

Subject of research – methods and tools for developing an intelligent semantic recommendation system based on NLP technologies, text embeddings, and user behavioral data analysis.

Purpose of the study – to develop a practical implementation of an intelligent semantic media recommendation system that provides personalized recommendations for movies, books, and video games based on the analysis of users’ natural language queries.

Research methods – theoretical and experimental. The development methodology is based on Sentence Transformers text embedding technology, cosine similarity for semantic comparison, and content-based filtering methods.

This bachelor's thesis investigates modern approaches to the development of recommender systems and methods of semantic natural language analysis. Content-based, collaborative, and hybrid filtering approaches are analyzed, and the application of transformer models and large language models for generating personalized recommendations is examined. A web-based recommendation system for selecting movies, books, and video games based on users’ natural language queries has been developed.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 7
Вступ 9
1 Аналіз предметної галузі та постановка задачі дослідження 11
1.1 Особливості рекомендаційних систем мультимедійного контенту ... 11
1.2 Огляд аналогів та існуючих рішень 14
1.3 Постановка задачі 20
2 Методи та технології побудови рекомендаційних систем 24
2.1 Методи побудови рекомендаційних систем 24
2.2 Методи семантичного аналізу тексту 26
2.3 Трансформерні моделі та текстові ембедінги 29
2.4 Методи персоналізації рекомендацій 31
2.5 Методи оцінювання якості рекомендаційних систем 34
3 Проєктування та програмна реалізація рекомендаційної системи 38
3.1 Архітектура рекомендаційної системи 38
3.2 Реалізація аналізу користувацьких запитів за допомогою LLM 41
3.3 Реалізація серверної частини системи 45
3.4 Реалізація клієнтської частини системи 50
3.5 Збереження даних та персоналізація рекомендацій 56
3.6 Демонстрація роботи рекомендаційної системи 60
Висновки 69
Перелік джерел посилання 71
Додаток А відомість кваліфікаційної роботи 75

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – штучний інтелект;

AI – Artificial Intelligence – штучний інтелект;

ALS – Alternating Least Squares – метод почергових найменших квадратів;

API – Application Programming Interface – інтерфейс програмування застосунків;

BERT – Bidirectional Encoder Representations from Transformers – двонаправлені енкодерні представлення на основі трансформерів;

CORS – Cross-Origin Resource Sharing – спільне використання ресурсів між різними джерелами;

CSS – Cascading Style Sheets – каскадні таблиці стилів;

CTR – Click-Through Rate – коефіцієнт кліків;

GPT – Generative Pre-trained Transformer – генеративний попередньо навчений трансформер;

HTML – HyperText Markup Language – мова гіпертекстової розмітки;

HTTP – HyperText Transfer Protocol – протокол передачі гіпертексту;

JSON – JavaScript Object Notation – текстовий формат обміну даними на основі JavaScript;

LLM – Large Language Model – велика мовна модель;

LSA – Latent Semantic Analysis – латентно-семантичний аналіз;

MAP – Mean Average Precision – середня усереднена точність;

MRR – Mean Reciprocal Rank – середній зворотний ранг;

nDCG – Normalized Discounted Cumulative Gain – нормалізований дисконтований кумулятивний прибуток;

NLP – Natural Language Processing – обробка природної мови;

RAWG – RAWG Video Games Database – база даних відеоігор RAWG;

REST – Representational State Transfer – архітектурний стиль передачі репрезентативного стану;

RS – Recommender Systems – рекомендаційні системи;

SBERT – Sentence-BERT – речення-BERT;

SVD – Singular Value Decomposition – розклад за сингулярними числами;

TMDB – The Movie Database – база даних фільмів;

UI – User Interface – користувацький інтерфейс;

URL – Uniform Resource Locator – уніфікований локатор ресурсів.

ВСТУП

Стрімке зростання обсягів цифрового контенту створює нові виклики для користувачів онлайн-платформ. Кількість доступних фільмів, книг та відеоігор постійно збільшується, що ускладнює процес пошуку матеріалів, які відповідають індивідуальним інтересам користувача. Традиційні пошукові системи здебільшого працюють на основі збігу ключових слів та не завжди здатні враховувати семантичний зміст запиту або особисті вподобання користувача. У зв'язку з цим особливої актуальності набувають рекомендаційні системи, які автоматизують процес підбору контенту та допомагають користувачам знаходити релевантні об'єкти серед великого інформаційного потоку.

Сучасний розвиток технологій штучного інтелекту та обробки природної мови відкрив нові можливості для побудови рекомендаційних систем. Трансформерні моделі та великі мовні моделі здатні аналізувати природномовні запити користувачів, визначати їхній зміст та формувати рекомендації на основі семантичної схожості, а не лише формального збігу окремих слів. Це дозволяє підвищити якість рекомендацій та зробити взаємодію із системою більш природною та зрозумілою для користувача.

Актуальність роботи обумовлена необхідністю створення інтелектуальних рекомендаційних систем, здатних працювати з природномовними запитами та забезпечувати персоналізований підбір мультимедійного контенту. Особливого значення набувають рішення, що поєднують можливості великих мовних моделей із зовнішніми джерелами даних та враховують попередню активність користувача для покращення результатів рекомендації.

Метою роботи є розробка рекомендаційної системи для підбору фільмів, книг та відеоігор на основі аналізу користувацьких запитів природною мовою. Для досягнення поставленої мети було досліджено сучасні підходи до побудови рекомендаційних систем, методи

семантичного аналізу тексту, принципи використання трансформерних моделей та великих мовних моделей, а також реалізовано програмний застосунок, що забезпечує пошук і персоналізований підбір контенту.

Практичним результатом роботи є веборієнтована рекомендаційна система, реалізована з використанням мови програмування Python, фреймворку FastAPI та зовнішніх API сервісів TMDB, Google Books і RAWG. Система дозволяє користувачу формулювати запити у довільній формі, отримувати рекомендації для різних типів медіаконтенту, переглядати історію взаємодій та використовувати елементи персоналізації на основі попередніх пошуків і оцінок.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Особливості рекомендаційних систем мультимедійного контенту

У сучасному цифровому середовищі можна спостерігати, що обсяг мультимедійного контенту постійно зростає. Користувачі мають доступ до великої кількості медіа: фільмів, книг, відеоігор, музики та іншого цифрового контенту, через онлайн-платформи та стрімінгові сервіси. Такий стрімкий розвиток цифрових медіа зумовив появу проблеми інформаційного перенавантаження.

Сам термін «інформаційне перенавантаження» (англ. Information Overload) був уперше введений у широкий науковий обіг Елвіном Тоффлером у праці «Шок майбутнього» (1970), а згодом детально операціоналізований у контексті цифрових систем [1].

У середовищі, де платформа Netflix станом на 2024 рік містить понад 7 000 одиниць відеоконтенту [2], книжкова платформа Goodreads налічує понад 9 мільйонів активних читачів, які відстежують сотні мільйонів книг щороку [3], а сервіс Steam пропонує понад 170 000 ігор [4], людина об’єктивно не здатна самостійно здійснити швидкий та раціональний вибір релевантного контенту без використання додаткових інструментів підтримки.

Саме для вирішення подібних проблем були розроблені системи рекомендацій (англ. Recommender Systems, RS) – програмні комплекси, що автоматично формують персоналізовані пропозиції контенту або товарів для кожного конкретного користувача відповідно до інтересів, поведінки та вподобань користувача. Згідно з визначенням Ф. Річчі, Л. Рокача та Б. Шапіри, системи рекомендацій є програмними системами, що забезпечують персоналізовані рекомендації та допомагають користувачам

знаходити релевантні об’єкти серед великої кількості доступних варіантів [5].

Ключова мета таких систем – подолання інформаційного перенавантаження через автоматизовану персоналізацію. RS широко використовуються у різних сферах цифрових сервісів. Найбільш поширеними прикладами є стрімінгові платформи, електронні магазини, соціальні мережі та медіасервіси.

Також однією з найважливіших сфер застосування рекомендаційної системи є платформи мультимедійного контенту. До таких платформ належать сервіси для перегляду фільмів (Netflix, Amazon Prime Video, Disney+ тощо), електронні бібліотеки (Goodreads, Amazon Kindle і тд.) та платформи цифрової дистрибуції відеоігор (Steam, GOG, Epic Games Store). Через значний обсяг доступного контенту системи рекомендацій стали невід’ємною частиною таких платформ. Вони аналізують вподобання користувачів, історію переглядів, рейтинги та інші поведінкові фактори для формування персоналізованих рекомендацій.

Фільми, книги та відеоігри як об'єкти рекомендації мають суттєві відмінності, що безпосередньо впливають на архітектуру рекомендаційних систем. Фільми характеризуються відносно стандартизованим набором метаданих: жанр, режисер, акторський склад, рік випуску, тривалість. Книги відрізняються значно більшою варіативністю у довжині та стилі, ніж фільми.

Крім того, видавнича індустрія менш стандартизована щодо жанрових класифікацій: одна й та сама книга може бути віднесена до кількох категорій різними платформами. Відеоігри є найбільш гетеро-генним видом медіаконтенту з точки зору рекомендаційних систем. Вони поєднують у собі елементи наративу (як книги), візуальної атмосфери (як фільми), а також специфічні ігрові механіки, що не мають аналогів в інших медіа.

У літературі виділяють три основні підходи до побудови рекомендаційної системи [5]:

– контентна фільтрація (англ. content-based filtering) базується на аналізі характеристик самих об'єктів рекомендації, таких як жанрові теги, ключові слова та формальні атрибути контенту;

– колаборативна фільтрація (англ. collaborative filtering) ґрунтується на аналізі поведінки користувачів та пошуку схожих профілів або об'єктів на основі матриці взаємодій;

– гібридна рекомендаційна система поєднують обидва підходи для підвищення точності рекомендацій і компенсації їхніх індивідуальних недоліків.

Попри їх широке використання, класичні системи рекомендацій мають низку суттєвих обмежень.

По-перше, залежність від ключових слів та формальних характеристик не дозволяє системам розпізнавати приховану семантичну схожість між об'єктами.

По-друге, колаборативна фільтрація має проблему «холодного старту» (англ. cold start problem), коли система не здатна формувати якісні рекомендації для нових користувачів або нового контенту через відсутність достатньої кількості даних.

По-третє, більшість рекомендаційних систем працюють лише в межах одного типу медіа та не здатні виявляти змістові зв’язки між різними категоріями контенту. Крім того, алгоритми, засновані на популярності, схильні рекомендувати переважно масовий контент, ігноруючи менш відомі, але потенційно релевантні твори.

Таким чином, класичні системи рекомендацій мають концептуальні обмеження, пов’язані з недостатнім семантичним розумінням контенту, слабкою персоналізацією та відсутністю міждоменної сумісності. Подолання цих проблем потребує застосування сучасних методів семантичного аналізу тексту та інтелектуальних моделей персоналізації.

1.2 Огляд аналогів та існуючих рішень

Системи рекомендацій в наш час є невід'ємною складовою цифрових платформ, що працюють із великими обсягами мультимедійного контенту. Найбільш активно рекомендаційні системи використовуються у стрімінгових сервісах, електронних магазинах, соціальних мережах та платформах цифрової дистрибуції медіа. Основною метою таких систем є персоналізований підбір контенту відповідно до інтересів та поведінки користувача.

Одним із найвідоміших прикладів рекомендаційної системи є платформа Netflix, яка використовує комбінацію колаборативної фільтрації, контентного аналізу та алгоритмів персоналізації для формування рекомендацій фільмів і серіалів. У роботі Gomez-Uribe та Hunt зазначається, що система рекомендацій Netflix аналізує історію переглядів, оцінки, час взаємодії з контентом та поведінкові патерни користувачів для формування індивідуалізованих рекомендацій [6].

Характерною рисою підходу Netflix є персоналізація не лише переліку рекомендацій, але й мініатюр обкладинок: різним користувачам демонструються різні зображення одного й того самого фільму залежно від їхніх уподобань [6]. Окреме місце займає модель матричної факторизації (англ. matrix factorization), що дозволяє виявляти латентні фактори вподобань користувача навіть за умови неповних даних.

Крім того, у Netflix використовується багатоступеневий процес оцінювання ефективності рекомендаційні алгоритми. Схему процесу оцінювання рекомендаційного алгоритму у Netflix наведено на рисунку 1.1.

Нові алгоритми спочатку тестуються в режимі offline experimentation шляхом аналізу історичних даних користувачів та прогнозування ймовірної взаємодії з контентом. Після отримання позитивних результатів проводиться A/B-тестування за участю реальних користувачів платформи. Лише після успішного проходження A/B-тесту новий

рекомендаційний алгоритм інтегрується в основну систему рекомендацій [6].

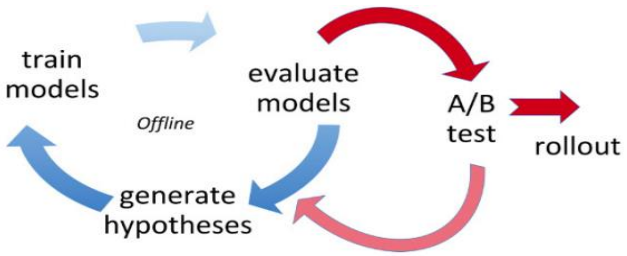


Рисунок 1.1 – Процес тестування рекомендаційного алгоритму у Netflix [6]

Рекомендаційна система YouTube, описана у дослідженні Ковінгтона, Адамса та Сарджіна [7], побудована на основі глибоких нейронних мереж (англ. Deep Neural Networks) та реалізує багаторівневий підхід до формування персоналізованих рекомендацій. На рисунку 1.2 наведено загальну архітектуру рекомендаційної системи YouTube, побудовану за принципом «recommendation funnel». Такий підхід передбачає багатоступеневий процес відбору та ранжування контенту, у межах якого система поступово звужує множину потенційно релевантних відео для кінцевого користувача.

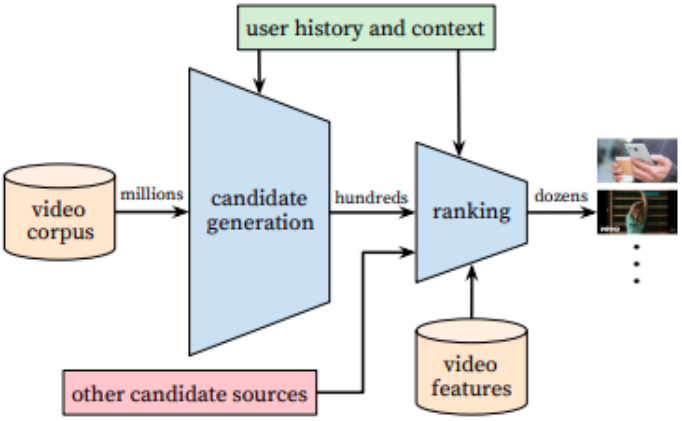


Рисунок 1.2 – Архітектура рекомендаційної системи YouTube та процес поетапного ранжування відео [7]

Архітектура системи складається з двох основних етапів. На першому етапі модель candidate generation відбирає кілька сотень потенційно релевантних відео з мільйонів доступних на платформі. На другому етапі ranking-модель виконує їх подальше ранжування та присвоює кожному кандидату фінальний рівень релевантності з урахуванням великої кількості поведінкових і контекстуальних ознак. Серед таких ознак використовуються історія переглядів, пошукові запити, демографічні характеристики користувача та особливості взаємодії з платформою. Автори підкреслюють, що ключовою метою рекомендаційної системи YouTube є максимізація часу перегляду контенту, а не лише кількості переходів або кліків [7].

У сфері музичного контенту рекомендаційна система активно використовує Spotify. Музичний стрімінговий сервіс використовує гібридний підхід, що поєднує три компоненти: колаборативну фільтрацію на основі матриці взаємодій користувач–трек, контентний аналіз аудіосигналу засобами згорткових нейронних мереж, а також NLP-аналіз текстів музичних рецензій та плейлистів [8].

У сфері електронної комерції одним із найвідоміших прикладів є Amazon, де використовується алгоритм item-to-item collaborative filtering для аналізу взаємозв’язків між товарами та формування персоналізованих рекомендацій [9]. Принцип роботи алгоритму полягає у попередньому обчисленні матриці схожості між товарами на основі спільних покупок, що дозволяє здійснювати рекомендації в режимі реального часу навіть при масштабі в мільярди товарів. На відміну від user-based підходу, item-based метод є значно стабільнішим та масштабованішим, оскільки схожість між товарами змінюється рідше, ніж вподобання користувачів [9]. Сучасна система Amazon доповнена моделями глибокого навчання та сесійними рекомендаціями.

Книжкова соціальна мережа Goodreads застосовує системи рекомендацій, що базуються переважно на колаборативній фільтрації:

аналізуються списки прочитаного, рейтинги та користувацькі полиці. Додатково використовуються елементи контентної фільтрації на основі жанрових тегів та авторів. Дослідники зазначають, що традиційні рекомендаційні системи Goodreads значною мірою залежать від формальних метаданих і поведінкових факторів, що обмежує здатність системи виявляти приховану семантичну схожість між творами [10]. Зокрема, системи рекомендацій можуть недостатньо точно рекомендувати книги зі схожою атмосферою або тематикою, якщо вони належать до різних жанрових категорій. Архітектура гібридної системи рекомендацій для книжкового контенту, що поєднує collaborative filtering та content-based filtering, наведено на рисунку 1.3.

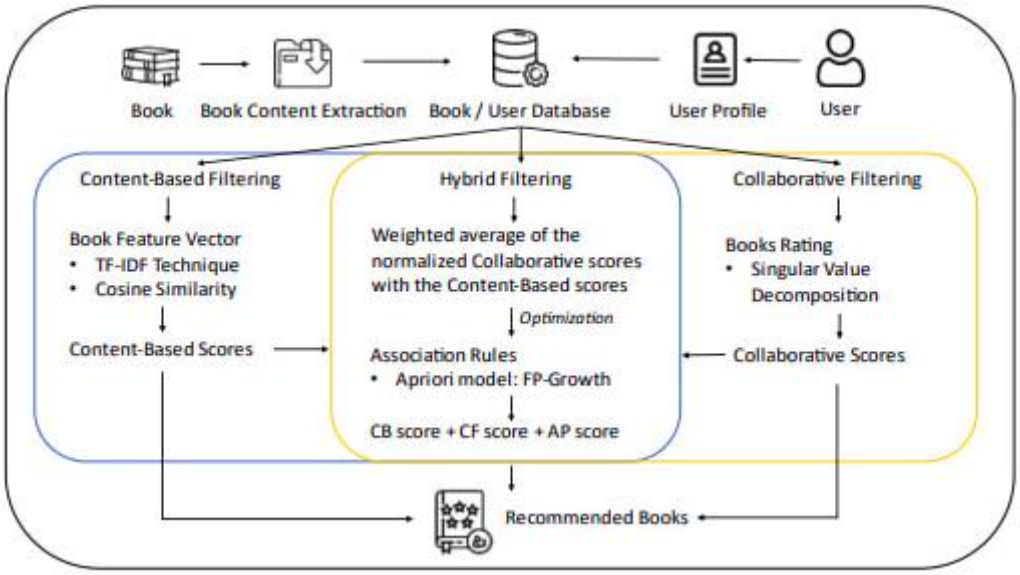


Рисунок 1.3 – Архітектура гібридної рекомендаційної системи для книжкового контенту [10]

Платформа цифрової дистрибуції відеоігор Steam накопичує масиви поведінкових даних гравців, що слугують основою для рекомендаційних механізмів. Дослідження Sifa, Drachen та Bauckhage [11], проведене на основі даних понад 6 мільйонів гравців та 3000 ігор, виявило стійкі патерни у розподілі часу гри та структурі ігрових бібліотек користувачів. Зокрема,

авторами було ідентифіковано 11 кластерів гравців – від тих, хто зосереджується на одній грі, до тих, хто рівномірно розподіляє час між багатьма проєктами (рисунок 1.4).

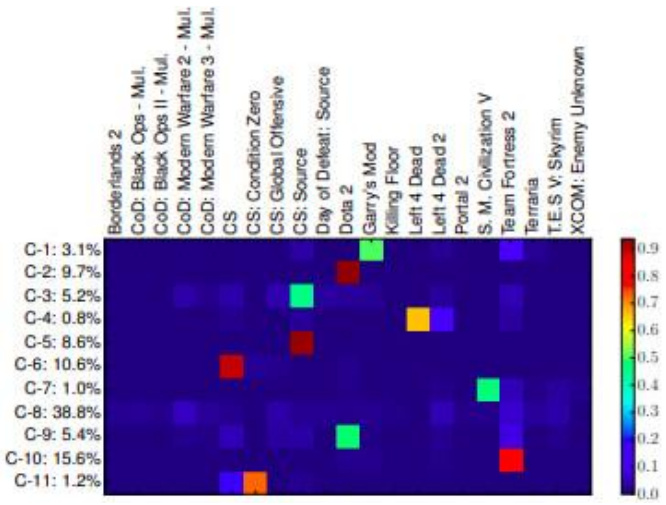


Рисунок 1.4 – Кластеризація 6 мільйонів гравців Steam за поведінковими профілями [11]

Ці поведінкові профілі є основою для колаборативної фільтрації: система аналізує схожість між бібліотеками та часом гри різних користувачів для формування рекомендацій. Додатково Steam використовує краудсорсингову систему тегів, де гравці самостійно присвоюють іграм описові характеристики, що формує багатовимірний простір для контентної фільтрації.

Паралельно з розвитком комерційних рішень активно розвивається академічний напрям дослідження рекомендаційних систем. Ключовими темами сучасних досліджень є застосування текстових ембедингів, семантичного пошуку та персоналізації на основі поведінки користувача.

Фундаментальною роботою у сфері семантичного представлення тексту є дослідження Деввліна та ін. [12], присвячене моделі BERT (Bidirectional Encoder Representations from Transformers). Автори запропонували підхід до попереднього навчання двонаправленого

трансформера на великих текстових корпусах з подальшим донавчанням на конкретних задачах. Отримані ембедінги кодують семантичний зміст тексту у вигляді щільного вектора фіксованої розмірності, що дозволяє порівнювати документи за змістом, а не лише за ключовими словами. Застосування BERT-подібних моделей у рекомендаційних системах дозволяє долати обмеження класичного TF-IDF підходу, забезпечуючи розуміння синонімії та семантичних зв'язків.

Рімер та Гулладжані [13] розробили архітектуру Sentence-BERT (SBERT) – модифікацію BERT, оптимізовану для ефективного обчислення семантичної схожості між реченнями. Ключовою інновацією є використання сіамської мережі (англ. Siamese network), що дозволяє отримувати ембедінги речень придатні для прямого порівняння через косинусну схожість. На відміну від оригінального BERT, SBERT скорочує час обчислення схожості між двома реченнями з 65 годин до менш ніж 5 секунд на стандартному наборі з 10 000 речень [13]. Саме ця властивість робить Sentence Transformers практично придатними для рекомендаційних систем реального часу, де необхідно швидко порівнювати запит із великою базою описів контенту.

Персоналізовані рекомендації на основі поведінки користувача. Дослідження Фей Суна та ін. [14] присвячене побудові рекомендаційних систем на основі послідовного аналізу взаємодій користувача, де автори пропонують модель BERT4Rec, що застосовує механізм самоуваги (self-attention) трансформера для моделювання послідовності дій користувача – переглядів, лайків, оцінок. Модель навчається передбачати наступний елемент послідовності, що дозволяє враховувати не лише статичний профіль вподобань, але й динаміку їх зміни. Результати показують перевагу BERT4Rec над класичними рекурентними моделями на чотирьох стандартних датасетах [14]. Підхід демонструє перспективність застосування трансформерних архітектур для задач персоналізованої рекомендації.

Окремим напрямом досліджень є крос-доменна рекомендація (англ. cross-domain recommendation) – формування рекомендацій із залученням інформації з кількох предметних областей.

Фен Джу та ін. [15] систематизують підходи до крос-доменної рекомендації та виокремлюють три основні стратегії: перенесення знань (transfer learning), спільне навчання (joint learning) та відображення ознак між доменами (feature mapping). Автори підкреслюють, що крос-доменні системи здатні суттєво пом'якшувати проблему холодного старту, використовуючи інформацію з суміжних доменів для нових користувачів [15]. Так, знання про вподобання користувача у сфері книг можуть слугувати початковою точкою для рекомендації фільмів або ігор зі схожою тематикою.

Таким чином, огляд існуючих рішень свідчить про те, що комерційні платформи досягли значного рівня ефективності завдяки великим обсягам даних та складним гібридним алгоритмам. Водночас спільним обмеженням більшості розглянутих систем залишається недостатній рівень семантичного розуміння контенту та обмежені можливості крос-доменної рекомендації. Академічні дослідження у сфері текстових ембедингів та трансформерних архітектур відкривають шляхи до подолання цих обмежень, однак їх інтеграція у практичні системи лишається активною дослідницькою задачею.

1.3 Постановка задачі

Проведений аналіз предметної галузі та огляд існуючих рішень дозволяють стверджувати, що проблема ефективної рекомендації медіаконтенту залишається актуальною попри значний прогрес у розвитку комерційних платформ. Актуальність подальших досліджень обумовлена рядом чинників.

По-перше, обсяги цифрового контенту продовжують стрімко зростати. За даними аналітичної компанії Technavio, світовий ринок digital content демонструє стабільне щорічне зростання (рисунок 1.5), а сегмент цифрового відеоконтенту оцінювався у понад 534 млрд доларів США у 2024 році [16]. Це свідчить про постійне збільшення кількості доступного мультимедійного контенту на стрімінгових платформах, у цифрових бібліотеках та сервісах дистрибуції відеоігор. За таких умов класичні підходи до рекомендації, засновані лише на жанрових тегах, ключових словах та агрегованих рейтингах, дедалі менше відповідають реальним потребам користувачів, які прагнуть знайти контент відповідно до атмосфери, тематики або емоційного сприйняття.

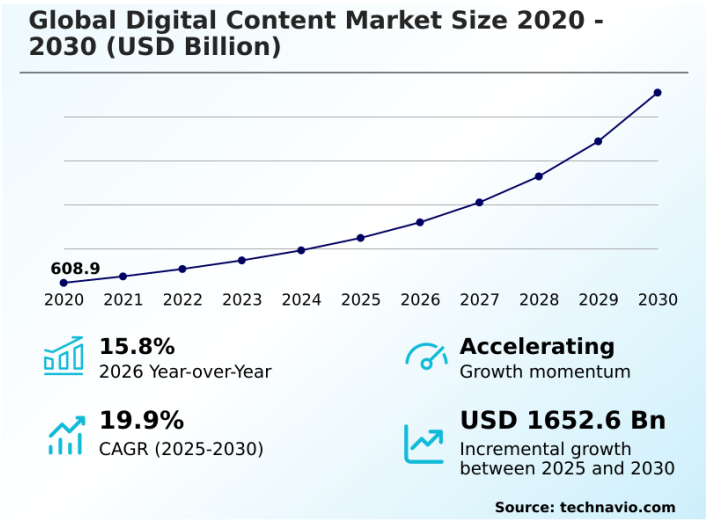


Рисунок 1.5 – Потенційне зростання цифрового ринку в проміжку з 2020 по 2030 [16]

По-друге, користувачі все частіше формулюють свої запити природною мовою – описово, емоційно, нечітко. Запити на кшталт «щось напружене з несподіваною кінцівкою» або «книга, схожа за атмосферою на гру Journey» не піддаються обробці засобами класичного пошуку за ключовими словами. Для адекватної інтерпретації таких запитів необхідне семантичне розуміння тексту – здатність системи виявляти зміст і контекст,

а не лише здійснювати лексичний збіг [12]. Статистика пошукових запитів на великих платформах підтверджує цю тенденцію: частка описових і емоційно забарвлених запитів стабільно зростає порівняно з точними назвами творів чи імен авторів. Це свідчить про те, що користувачі інтуїтивно намагаються використовувати природну мову як інструмент пошуку, навіть коли платформа технічно не підтримує такий формат.

По-третє, існуючі комерційні системи здебільшого функціонують у межах одного медіадомену: кінорекомендатори не пропонують ігор, а книжкові сервіси не враховують ігровий або кінематографічний досвід користувача. Між тим, культурні вподобання людини мають крос-доменну природу: читач, який захоплюється постапокаліптичною літературою, з високою ймовірністю зацікавиться відповідними фільмами та іграми. Відсутність крос-доменних рекомендаційних систем для медіаконтенту є суттєвою прогалиною у сучасному ландшафті рішень [15].

По-четверте, академічні дослідження останніх років демонструють значний потенціал методів обробки природної мови та текстових ембедингів для побудови семантичних рекомендаційних систем [13], [14]. Проте перехід від теоретичних моделей до практично застосовних систем, орієнтованих на реальних користувачів, залишається недостатньо розробленим напрямом. Більшість наявних досліджень зосереджені або на вузькоспеціалізованих датасетах (виключно фільми або виключно книги), або на теоретичній валідації моделей без реальної інтеграції у користувацький інтерфейс. Розрив між академічними результатами та їх практичним втіленням у вигляді повноцінних веб-застосунків залишається значним.

Таким чином, сукупність наведених факторів визначає актуальність дослідження у сфері інтелектуальних систем рекомендацій мультимедійного контенту, орієнтованих на семантичний аналіз тексту, персоналізацію та крос-доменні рекомендації.

У зв’язку з цим метою дослідження є аналіз сучасних методів та підходів до побудови інтелектуальних рекомендаційних систем мультимедійного контенту, здатних забезпечувати семантичний аналіз текстових описів, персоналізований підбір контенту та формування рекомендацій між різними типами медіа.

Для досягнення поставленої мети необхідно:

– проаналізувати сучасні підходи до побудови рекомендаційної системи мультимедійного контенту;

– дослідити методи semantic search, текстових ембедингів та визначення семантичної схожості тексту;

– проаналізувати підходи до персоналізації рекомендацій і побудови крос-доменної рекомендаційної системи;

– сформулювати вимоги до інтелектуальної системи рекомендацій мультимедійного контенту;

– розробити архітектуру системи семантичних рекомендацій;

– реалізувати модуль обробки природномовних запитів користувача;

– реалізувати механізм пошуку та ранжування контенту на основі текстових ембедингів;

– реалізувати рекомендаційну систему для підбору фільмів, книг та відеоігор;

– провести експериментальну перевірку роботи системи та оцінити якість сформованих рекомендацій.

2 МЕТОДИ ТА ТЕХНОЛОГІЇ ПОБУДОВИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

2.1 Методи побудови рекомендаційних систем

Рекомендаційні системи є одним із ключових інструментів персоналізації цифрових сервісів. Залежно від використовуваних підходів до аналізу даних та формування рекомендацій виділяють три основні класи методів: контентна фільтрація, колаборативна фільтрація та гібридні підходи.

Контентна фільтрація базується на аналізі характеристик самих об'єктів рекомендації. Система формує профіль уподобань користувача на основі атрибутів контенту, з яким він взаємодіяв раніше, та рекомендує нові об'єкти зі схожими характеристиками. У класичних реалізаціях такий підхід спирається на метадані – жанрові теги, ключові слова, опис, авторство. Перевагою методу є незалежність від даних інших користувачів та здатність пояснювати рекомендації через конкретні ознаки об'єкта. Водночас контентна фільтрація має суттєве обмеження – надмірна спеціалізація: система схильна рекомендувати лише контент, схожий на вже переглянутий, що ускладнює відкриття нових тематичних напрямів [5].

Колаборативна фільтрація ґрунтується на принципі «користувачі зі схожими вподобаннями цікавляться схожим контентом». Метод аналізує матрицю взаємодій «користувач–об'єкт» та виявляє латентні закономірності у поведінці користувачів. Розрізняють два основні різновиди: memory-based та model-based підходи. Memory-based методи, зокрема user-based та item-based collaborative filtering, обчислюють схожість між користувачами або об'єктами безпосередньо на основі матриці взаємодій. Model-based методи, серед яких матрична факторизація та її варіанти SVD++ і ALS, навчають компактне латентне представлення користувачів та об'єктів. Основним недоліком колаборативної фільтрації є проблема «холодного старту» –

неможливість формування якісних рекомендацій для нових користувачів або нового контенту через відсутність достатньої кількості даних взаємодій [5].

Гібридні рекомендаційні системи поєднують контентну та колаборативну фільтрацію з метою компенсації індивідуальних недоліків кожного підходу. Бурке виділяє декілька стратегій гібридизації: зважена (англ. weighted), де результати обох компонент об'єднуються з ваговими коефіцієнтами; перемикальна (англ. switching), де система обирає метод залежно від контексту; каскадна (англ. cascade), де один метод уточнює результати іншого [17]. Саме гібридні системи використовуються більшістю сучасних комерційних платформ – Netflix, Spotify, Amazon – оскільки поєднання підходів дозволяє досягти вищої точності рекомендацій за різних умов використання [6], [8], [9].

Окремим і відносно новим напрямом є підходи н (англ. Large Language Models, LLM). На відміну від класичних методів, LLM-підходи дозволяють інтерпретувати довільні природномовні запити користувача та перетворювати їх на структуровані параметри пошуку. Така архітектура усуває необхідність накопичення великих обсягів поведінкових даних і долає проблему холодного старту, оскільки мовна модель спирається на попередньо набуті знання про контент [18]. Порівняння основних методів побудови рекомендаційних систем наведено у таблиці 2.1.

Таблиця 2.1 – Порівняння методів побудови рекомендаційних систем

Метод	Основа	Переваги	Недоліки
Контентна фільтрація	Атрибути об'єктів	Незалежність від інших користувачів, пояснюваність	Надмірна спеціалізація
Колаборативна фільтрація	Матриця взаємодій	Виявлення латентних вподобань	Проблема холодного старту

Продовження таблиці 2.1

Метод	Основа	Переваги	Недоліки
Гібридна система	Комбінація підходів	Висока точність, компенсація недоліків	Складність реалізації
LLM-підхід	Мовна модель	Розуміння природної мови, відсутність холодного старту	Залежність від якості моделі

Таким чином, вибір методу побудови рекомендаційної системи визначається характером доступних даних, вимогами до персоналізації та особливостями цільового контенту. У контексті мультимедійних платформ з різнорідним контентом найбільш перспективним є застосування гібридних підходів та LLM-архітектур, що забезпечують семантичне розуміння запитів та ефективну роботу за умов обмеженості поведінкових даних.

2.2 Методи семантичного аналізу тексту

Семантичний аналіз тексту є ключовою складовою сучасних рекомендаційних систем, що дозволяє долати обмеження класичних підходів, заснованих на ключових словах та формальних метаданих. На відміну від лексичного пошуку, семантичні методи спрямовані на виявлення тематики, контексту та прихованих зв'язків між поняттями, що формують змістове значення тексту.

Одним із перших широко застосовуваних методів семантичного аналізу є латентно-семантичний аналіз (англ. Latent Semantic Analysis, LSA). Метод ґрунтується на матричному розкладанні (SVD) матриці «термін–документ» для виявлення прихованих семантичних структур у великих текстових корпусах. LSA дозволяє виявляти синонімічні відносини між термінами та групувати тематично схожі документи навіть за відсутності

спільних ключових слів. Проте LSA має суттєві обмеження: метод не враховує порядок слів у тексті та погано масштабується на великих обсягах даних [5].

Значним кроком уперед у розвитку семантичного аналізу стала модель word2vec, запропонована Міколовим та ін. [19], яка дозволяє отримувати щільні векторні представлення слів (англ. word embeddings) шляхом навчання нейронної мережі на великих текстових корпусах. Ключовою властивістю word2vec є здатність кодувати семантичні відносини між словами у геометричній структурі векторного простору – наприклад, вектори слів «король» та «королева» відрізняються на той самий вектор, що і «чоловік» та «жінка». Застосування word2vec у рекомендаційних системах дозволяє порівнювати описи контенту за змістом, а не лише за збігом ключових слів [19].

Подальший розвиток методів побудови текстових ембедингів пов'язаний із появою контекстуальних моделей, заснованих на механізмі уваги (attention mechanism) [20]. Модель BERT, розроблена Девліном та ін. [12], реалізує двонаправлене кодування тексту, що дозволяє враховувати контекст кожного слова з обох боків речення одночасно. На відміну від word2vec, де кожне слово має фіксований вектор незалежно від контексту, BERT генерує динамічні ембединги: те саме слово матиме різні векторні представлення залежно від контексту вживання. Попереднє навчання BERT на великих корпусах із подальшим донавчанням на конкретних задачах (англ. fine-tuning) забезпечує високу якість семантичного представлення для широкого спектру прикладних задач [12].

Для застосування у рекомендаційних системах особливого значення набуває ефективне обчислення семантичної схожості між реченнями та документами. Рімерс та Гуревич розробили архітектуру Sentence-BERT (SBERT) – модифікацію BERT, оптимізовану для порівняння речень [13]. Ключовою інновацією є використання сіамської мережі (англ. Siamese network), що дозволяє отримувати ембединги речень, придатні для

прямого порівняння через косинусну схожість. На відміну від оригінального BERT, SBERT скорочує час обчислення схожості між двома реченнями з 65 годин до менш ніж 5 секунд на стандартному наборі з 10 000 речень [13]. Саме ця властивість робить Sentence Transformers практично придатними для рекомендаційних систем реального часу.

Окремим напрямом є застосування великих мовних моделей для семантичного аналізу запитів у природномовній формі. На відміну від статичних ембедингів, LLM здатні інтерпретувати складні описові запити – наприклад, «книга схожа на Гаррі Поттера, але для дорослих» – та перетворювати їх на структуровані параметри пошуку. Такий підхід дозволяє рекомендаційній системі працювати безпосередньо з природномовними запитами користувача без попередньої обробки або структурування вхідних даних [18], [20].

Порівняння основних методів семантичного аналізу тексту наведено у таблиці 2.2.

Таблиця 2.2 – Порівняння методів семантичного аналізу тексту

Метод	Тип представлення	Контекстуальність	Застосування у RS
LSA	Матричне розкладання	Відсутня	Тематичне групування
Word2Vec	Статичні ембединги	Відсутня	Схожість контенту
BERT	Контекстуальні ембединги	Двонаправлена	Семантичний пошук
Sentence-BERT	Ембединги речень	Двонаправлена	Порівняння описів
LLM	Генеративна модель	Повна	Аналіз запитів

Еволюція методів семантичного аналізу тексту відображає перехід від статистичних підходів до нейромережевих архітектур, здатних розуміти контекст та семантичні відносини у тексті. Застосування сучасних

трансформерних моделей та LLM у рекомендаційних системах дозволяє суттєво підвищити якість семантичного зіставлення запитів користувача з контентом бази даних.

2.3 Трансформерні моделі та текстові ембединги

Трансформерна архітектура, запропонована Васвані та ін. у роботі «Attention Is All You Need» (2017), стала фундаментальним проривом у галузі обробки природної мови. На відміну від рекурентних нейронних мереж (RNN) та довгої короткострокової пам'яті (LSTM), трансформери обробляють усі елементи послідовності паралельно завдяки механізму самоуваги, що дозволяє значно прискорити навчання та ефективніше моделювати довгострокові залежності у тексті [20].

Механізм самоуваги є ключовим компонентом трансформерної архітектури. Для кожного токена вхідної послідовності модель обчислює вагові коефіцієнти відносно всіх інших токенів, що дозволяє динамічно визначати найбільш релевантні частини тексту для формування контекстного представлення. Завдяки цьому трансформери здатні ефективно враховувати взаємозв’язки між словами незалежно від їхньої відстані у тексті, що суттєво покращує якість семантичного аналізу [20].

Модель BERT, розроблена Девліним та ін. [12], реалізує двонаправлений трансформерний енкодер, що попередньо навчається на двох задачах: маскований прогноз токенів та прогноз наступного речення. Завдяки двонаправленому контексту BERT генерує значно якісніші ембединги порівняно з однонаправленими моделями – кожен токен кодується з урахуванням усього оточуючого контексту. Архітектура BERT існує у двох варіантах: BERT-Base (12 шарів трансформера, 110 млн параметрів) та BERT-Large (24 шари, 340 млн параметрів). Донавчання BERT на конкретних задачах дозволяє досягати високих результатів на широкому спектрі задач обробки природної мови [12].

Окрім BERT та SBERT, у сфері текстових ембедингів активно розвиваються й інші трансформерні архітектури. Модель RoBERTa є вдосконаленою версією BERT з оптимізованими параметрами попереднього навчання та відмовою від задачі прогнозування наступного речення, що дозволяє покращити якість мовних представлень [21]. Моделі сімейства GPT використовують архітектуру трансформерного декодера та орієнтовані переважно на генерацію тексту, проте також можуть застосовуватися для формування семантичних представлень текстових даних [22]. Сучасні мультимовні моделі, зокрема mBERT та XLM-RoBERTa, навчаються на текстах багатьох мов одночасно та забезпечують побудову спільного семантичного простору для крос-лінгвістичних задач пошуку й рекомендацій [23].

Застосування трансформерних ембедингів у рекомендаційних системах реалізується через декілька підходів. По-перше, ембединги описів контенту використовуються для обчислення семантичної схожості між об'єктами рекомендації – фільмами, книгами та іграми. По-друге, ембединги користувацьких запитів порівнюються з ембедингами об'єктів з метою пошуку найбільш релевантних результатів. По-третє, трансформерні моделі застосовуються для аналізу послідовності взаємодій користувача – як у моделі BERT4Rec [14], де механізм самоуваги моделює динаміку зміни вподобань у часі.

Порівняння основних трансформерних моделей для задач семантичного представлення тексту наведено у таблиці 2.3.

Таблиця 2.3 – Порівняння трансформерних моделей для текстових ембедингів

Модель	Архітектура	Кількість параметрів	Основне призначення
BERT-Base	Двонаправлений енкодер	110 млн	Розуміння тексту

Продовження таблиці 2.3

Модель	Архітектура	Кількість параметрів	Основне призначення
RoBERTa	Двонаправлений енкодер	125 млн	Семантичне представлення
Sentence-BERT	Сіамська мережа на основі BERT	110 млн	Порівняння речень
BERT4Rec	Трансформерний енкодер	Варіативно	Послідовна рекомендація
GPT	Однонаправлений декодер	117 млн+	Генерація тексту

Трансформерні моделі та текстові ембединги забезпечують якісно новий рівень семантичного розуміння контенту у рекомендаційних системах. Здатність моделей типу BERT та SBERT кодувати глибокий семантичний зміст текстів у щільних векторних представленнях дозволяє будувати рекомендаційні системи, що враховують змістову схожість між об'єктами незалежно від збігу ключових слів.

2.4 Методи персоналізації рекомендацій

Персоналізація є фундаментальною властивістю сучасних рекомендаційних систем, що відрізняє їх від загальних систем пошуку та сортування контенту. Мета персоналізації полягає у формуванні унікальних рекомендацій для кожного окремого користувача на основі його індивідуальних вподобань, поведінки та контексту взаємодії з системою.

Найбільш поширеним підходом до персоналізації є профілювання користувача – побудова формального представлення вподобань та інтересів конкретного користувача на основі накопичених даних про його взаємодію з системою. Профіль користувача може формуватися на основі явних сигналів – оцінок, лайків, відгуків – та неявних сигналів – історії переглядів, часу взаємодії з контентом, повторних звернень. Явні сигнали є більш

точними, проте вимагають активної участі користувача, тоді як неявні сигнали збираються автоматично, але можуть бути неоднозначними у трактуванні [5].

Підхід на основі послідовного моделювання враховує не лише статичний профіль вподобань, але й динаміку їх зміни у часі. Фей Сун та ін. [14] запропонували модель BERT4Rec, що застосовує механізм самоуваги трансформера для моделювання послідовності взаємодій користувача. Модель навчається передбачати наступний елемент послідовності на основі попередніх взаємодій, що дозволяє враховувати короткострокові зміни інтересів поряд із довгостроковими вподобаннями. Результати показують перевагу BERT4Rec над класичними рекурентними моделями на стандартних наборах даних для рекомендаційних систем [14].

Важливим аспектом персоналізації є вирішення проблеми холодного старту для нових користувачів. Класичні методи колаборативної фільтрації не здатні формувати якісні рекомендації без достатньої кількості даних взаємодій. Крос-доменна рекомендація є одним із підходів до вирішення цієї проблеми – система використовує інформацію про вподобання користувача з одного домену для формування рекомендацій в іншому. Жу та ін. [15] систематизують три основні стратегії крос-доменної рекомендації: перенесення знань, спільне навчання та відображення ознак між доменами. Автори підкреслюють, що крос-доменні системи здатні суттєво пом'якшувати проблему холодного старту, використовуючи інформацію з суміжних доменів для нових користувачів [15].

Контекстуальна персоналізація враховує зовнішні фактори, що впливають на вподобання користувача в конкретний момент часу – час доби, місцезнаходження, поточний настрій або соціальний контекст. Наприклад, один і той самий користувач може надавати перевагу різному контенту вдома ввечері та під час поїздки у транспорті. Контекстуальні системи використовують тензорне розкладання або моделі з контекстними ознаками для врахування таких факторів [5].

Підхід на основі великих мовних моделей відкриває якісно нові можливості для персоналізації. LLM здатні інтерпретувати контекст профілю користувача, виражений у природномовній формі, та враховувати його при формуванні рекомендацій. Замість формального векторного представлення профілю система може передавати до LLM опис у вигляді тексту – наприклад, «користувач нещодавно переглядав детективи та фентезі, високо оцінив книги Агати Крісті» – і отримувати персоналізовані рекомендації або параметри пошуку [18]. Такий підхід особливо ефективний у сценаріях з обмеженою кількістю поведінкових даних, оскільки мовна модель здатна компенсувати відсутність великої матриці взаємодій завдяки попередньо набутим знанням про контент [18].

Оцінювання ефективності персоналізації здійснюється за допомогою метрик, що вимірюють точність, різноманітність та новизну рекомендацій. Точність та повнота визначають частку релевантних об'єктів серед рекомендованих та серед усіх релевантних об'єктів відповідно. Метрика nDCG додатково враховує позицію релевантних об'єктів у списку рекомендацій. Різноманітність та новизна характеризують здатність системи пропонувати несхожий між собою та маловідомий контент, що допомагає уникати ефекту «бульбашки фільтрів» [5, 24].

Порівняння основних підходів до персоналізації рекомендацій наведено у таблиці 2.4.

Таблиця 2.4 – Порівняння методів персоналізації рекомендацій

Метод	Тип даних	Вирішення холодного старту	Врахування динаміки
Профілювання користувача	Явні та неявні сигнали	Часткове	Обмежене
Послідовне моделювання	Історія взаємодій	Ні	Так
Крос-доменна рекомендація	Міждоменні дані	Так	Обмежене

Продовження таблиці 2.4

Метод	Тип даних	Вирішення холодного старту	Врахування динаміки
Контекстуальна рекомендація	Контекстні ознаки	Ні	Так
LLM-персоналізація	Природномовний профіль	Так	Так

Сучасні методи персоналізації рекомендацій еволюціонували від статичного профілювання до динамічних моделей, що враховують послідовність взаємодій, контекст та міждоменні зв'язки. Застосування великих мовних моделей для персоналізації відкриває перспективний напрям розвитку рекомендаційних систем, поєднуючи гнучкість природномовної взаємодії з можливістю формування індивідуалізованих рекомендацій.

2.5 Методи оцінювання якості рекомендаційних систем

Оцінювання якості рекомендаційних систем є необхідним етапом їх розробки та вдосконалення. Вибір метрик оцінювання визначається типом рекомендаційної задачі, характером доступних даних та кінцевими цілями системи. У літературі виділяють три основні підходи до оцінювання: офлайн-оцінювання на основі історичних даних, онлайн-оцінювання з реальними користувачами та користувацькі дослідження [5], [25].

Офлайн-оцінювання є найбільш поширеним підходом у академічних дослідженнях завдяки відтворюваності та незалежності від реальних користувачів. Метод передбачає розбиття наявного датасету взаємодій на навчальну та тестову вибірки, після чого система навчається на першій і оцінюється на другій. Стандартним протоколом є k-fold крос-валідація або розбиття за часовим критерієм, коли тестова вибірка містить найновіші взаємодії користувачів. Ключовими метриками точності при офлайн-

оцінюванні є Precision@K та Recall@K, що вимірюють частку релевантних об'єктів серед перших K рекомендацій та частку охоплених релевантних об'єктів відповідно [5], [25].

Метрика Mean Average Precision (MAP) узагальнює Precision@K для різних порогових значень K та усереднює результати по всіх користувачах, що дозволяє отримати більш стабільну оцінку якості ранжування. Normalized Discounted Cumulative Gain (nDCG) є однією з найбільш широко використовуваних метрик ранжування у рекомендаційних системах. На відміну від Precision@K, nDCG враховує позицію релевантних об'єктів у списку рекомендацій – об'єкти, розміщені вище у списку, вносять більший внесок у підсумкову оцінку. Нормалізація забезпечує порівнянність значень між різними користувачами та датасетами [5], [26].

Метрика Mean Reciprocal Rank (MRR) вимірює середнє значення зворотного рангу першого релевантного об'єкта у списку рекомендацій. Вона особливо доречна у сценаріях, де користувач зацікавлений лише у першому точному результаті – наприклад, при пошуку конкретної книги або фільму за описом. Hit Rate@K є спрощеною бінарною метрикою, що фіксує частку випадків, коли хоча б один релевантний об'єкт потрапив до перших K рекомендацій. Обидві метрики широко застосовуються у порівняльних дослідженнях рекомендаційних алгоритмів як базові показники якості ранжування [14], [25].

Поряд із точністю, сучасні дослідження приділяють значну увагу метрикам різноманітності та новизни рекомендацій. Різноманітність вимірює середню попарну несхожість об'єктів у списку рекомендацій – висока різноманітність запобігає формуванню одноманітних підбірок із надто схожого контенту. Новизна характеризує здатність системи рекомендувати маловідомий контент, з яким користувач ще не стикався. Несподіваність є більш складною метрикою, що вимірює здатність системи рекомендувати контент, який є водночас несподіваним та релевантним для користувача. Дослідження показують, що оптимізація виключно за

метриками точності може призводити до ефекту «бульбашки фільтрів», коли система замикає користувача у вузькому колі знайомого контенту [5], [26].

Онлайн-оцінювання передбачає тестування рекомендаційної системи в умовах реальної взаємодії з користувачами. Найбільш поширеним методом є A/B-тестування, де різні групи користувачів отримують рекомендації від різних версій системи, після чого порівнюються поведінкові показники – коефіцієнт кліків (CTR), час сесії, конверсія. Саме такий підхід використовується у комерційних платформах – зокрема, Netflix застосовує багатоступеневий процес A/B-тестування перед впровадженням нових алгоритмів [6]. Перевагою онлайн-оцінювання є безпосереднє вимірювання впливу рекомендацій на поведінку реальних користувачів, проте метод є ресурсомістким та потребує значного трафіку для отримання статистично значущих результатів.

Для оцінювання систем на основі природномовних запитів, зокрема LLM-підходів, додатково застосовуються метрики якості семантичного зіставлення. Косинусна схожість між ембедингами запиту та рекомендованих об'єктів дозволяє кількісно оцінити семантичну релевантність результатів. У дослідженні Ліу та ін. [18] для оцінювання LLM-рекомендацій використовуються стандартні метрики Hit Rate та nDCG у порівнянні з класичними базовими методами, що дозволяє об'єктивно зіставити ефективність різних підходів. Автори підкреслюють, що LLM-системи демонструють конкурентоспроможні результати навіть без навчання на специфічних датасетах взаємодій, що підтверджує перспективність таких підходів для сценаріїв з обмеженими поведінковими даними [18].

Порівняння основних метрик оцінювання рекомендаційних систем наведено у таблиці 2.5.

Таблиця 2.5 – Порівняння метрик оцінювання рекомендаційних систем

Метрика	Тип	Що вимірює	Галузь застосування
Precision@K	Точність	Частка релевантних у топ-K	Загальне оцінювання
Recall@K	Повнота	Охоплення релевантних об'єктів	Загальне оцінювання
nDCG	Ранжування	Позиція релевантних об'єктів	Ранжовані списки
MRR	Ранжування	Ранг першого релевантного	Точковий пошук
Diversity	Різноманітність	Несхожість рекомендацій	Запобігання фільтр-бульбашці
CTR	Поведінкова	Частота кліків	Онлайн-оцінювання

Таким чином, комплексне оцінювання якості рекомендаційної системи потребує поєднання метрик точності, різноманітності та новизни як в офлайн-, так і в онлайн-режимах. Вибір конкретних метрик визначається цілями системи та особливостями взаємодії користувача з контентом – для систем, орієнтованих на семантичний пошук за природномовними запитами, доцільним є поєднання стандартних метрик ранжування з оцінкою семантичної релевантності результатів [25], [26].

3 ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

3.1 Архітектура рекомендаційної системи

Розроблена рекомендаційна система є веборієнтованим застосунком із клієнт-серверною архітектурою, що складається з трьох основних рівнів: клієнтська частина (фронтенд), серверна частина (бекенд) та зовнішні сервіси. Загальну архітектуру системи наведено на рисунку 3.1.

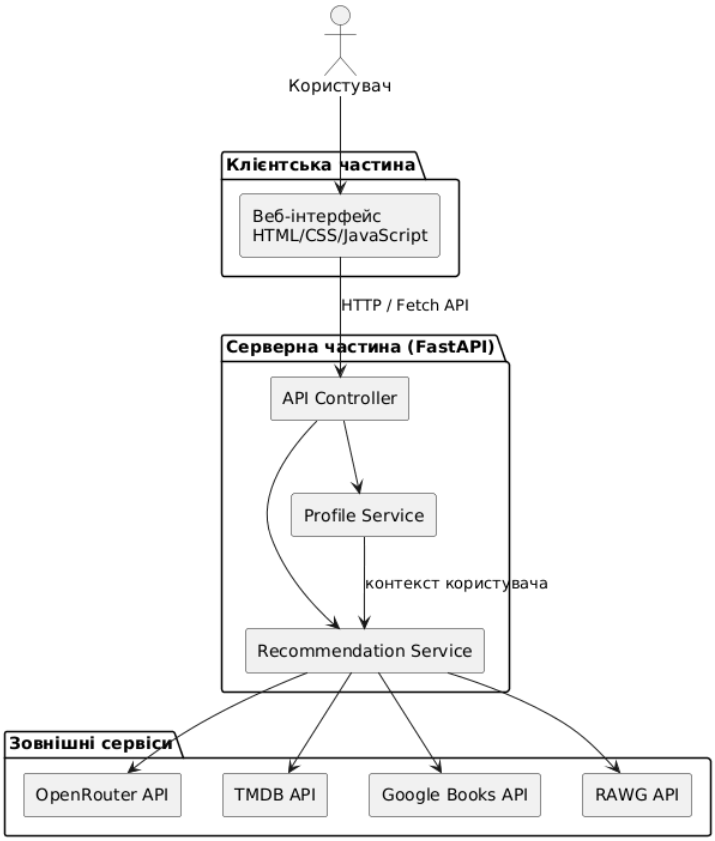


Рисунок 3.1 – Архітектура рекомендаційної системи

Клієнтська частина реалізована у вигляді багатосторінкового веб-застосунку на основі HTML, CSS та JavaScript. Інтерфейс містить окремі сторінки для пошуку фільмів, книг та ігор, сторінку результатів

рекомендацій, сторінку популярного контенту та кабінет користувача. Взаємодія з серверною частиною здійснюється через асинхронні HTTP-запити за допомогою Fetch API.

Серверна частина реалізована на основі фреймворку FastAPI мовою програмування Python. Бекенд виконує роль посередника між клієнтом, мовною моделлю та зовнішніми API. Сервер приймає запити від фронтенду, передає їх на аналіз до LLM, отримує структуровані параметри пошуку та звертається до відповідного зовнішнього API для отримання рекомендацій.

До зовнішніх сервісів системи належать: OpenRouter API для доступу до великої мовної моделі, TMDB API для пошуку фільмів, Google Books API для пошуку книг та RAWG API для пошуку відеоігор. Взаємодія з усіма зовнішніми сервісами здійснюється через HTTP-запити з використанням бібліотеки httpx.

Схему обробки користувацького запиту наведено на рисунку 3.2.



Рисунок 3.2 – Схема обробки користувацького запиту

Процес обробки запиту складається з таких етапів. На першому етапі користувач вводить запит у довільній природній мові на сторінці

відповідної медіа-категорії та натискає кнопку «Підібрати». Фронтенд формує JSON-об'єкт, що містить текст запиту, тип медіа-категорії, ідентифікатор користувача та додаткові фільтри, після чого відправляє POST-запит на ендпоінт /api/recommend серверної частини.

На другому етапі бекенд отримує запит та завантажує контекст профілю користувача – останні пошукові запити та високо оцінені елементи. Запит користувача разом із контекстом профілю передається до мовної моделі через OpenRouter API. LLM аналізує природномовний запит та повертає структурований JSON-об'єкт із параметрами пошуку, специфічними для відповідного зовнішнього API.

На третьому етапі бекенд використовує отримані параметри для звернення до відповідного зовнішнього API – TMDB, Google Books або RAWG – залежно від обраної медіа-категорії. Отримані результати проходять фільтрацію та дедуплікацію, після чого кожному елементу присвоюється відсоток схожості із запитом користувача на основі алгоритму ранжування.

На четвертому етапі сформований список рекомендацій повертається фронтенду у вигляді JSON-відповіді. Паралельно запит зберігається в історії користувача, а назви рекомендованих об'єктів – у файлі глобальної статистики для формування сторінки популярного контенту. Фронтенд отримує відповідь, зберігає дані у localStorage та перенаправляє користувача на сторінку результатів.

Взаємодія між компонентами системи здійснюється за принципом REST API – всі запити є stateless, тобто кожен запит містить усю необхідну інформацію для його обробки незалежно від попередніх запитів. Клієнт-серверна архітектура системи забезпечує чіткий розподіл відповідальності між компонентами: фронтенд відповідає за відображення інтерфейсу та взаємодію з користувачем, бекенд – за бізнес-логіку та координацію зовнішніх сервісів, а LLM – за семантичний аналіз природномовних запитів та генерацію параметрів пошуку.

3.2 Реалізація аналізу користувацьких запитів за допомогою LLM

Ключовою особливістю розробленої рекомендаційної системи є використання великої мовної моделі для аналізу природномовних запитів користувача. На відміну від класичних підходів, що вимагають структурованого введення даних через форми та фільтри, запропонована архітектура дозволяє користувачеві описувати бажаний контент довільною мовою – галузь наприклад, «гра про дівчину археологиню на острові Яматай» або «детектив у стилі Агати Крісті». Мовна модель інтерпретує такий запит та перетворює його на структуровані параметри пошуку для відповідного зовнішнього API.

Мовна модель виконує функцію семантичного перекладача між природномовним запитом користувача та формальними параметрами зовнішніх API. Доступ до LLM реалізований через сервіс OpenRouter, що надає уніфікований інтерфейс для роботи з різними мовними моделями. У системі використовується модель класу GPT з низькою температурою генерації (temperature = 0.3) для забезпечення стабільного та передбачуваного виводу у форматі JSON. Взаємодія з OpenRouter реалізована через офіційний Python SDK бібліотеки openai із кастомним base_url, що дозволяє використовувати стандартний інтерфейс OpenAI API для звернення до будь-якої моделі, доступної через OpenRouter.

Для кожної медіа-категорії розроблено окремий системний промпт, що зберігається у текстовому файлі в директорії prompts. Структуру каталогу промтів мовної моделі наведено на рисунку 3.3.

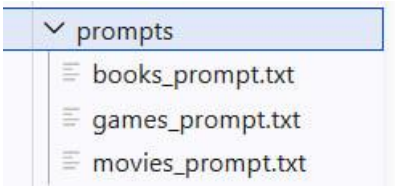


Рисунок 3.3 – Структура каталогу промптів мовної моделі

Системний промпт визначає роль моделі, описує структуру очікуваної відповіді у форматі JSON, надає перелік допустимих значень параметрів та містить набір прикладів запитів і відповідних параметрів пошуку. Використання окремих промптів для кожної категорії дозволяє точно налаштувати поведінку моделі відповідно до специфіки конкретного зовнішнього API.

Системний промпт для фільмів містить перелік жанрових ідентифікаторів TMDB та інструкції щодо формування полів keywords, genre_ids, year_from, year_to, vote_average_gte, specific_titles та sort_by. Фрагмент системного промпту для фільмів наведено на рисунку 3.4.

```
- 53: Thriller (Трилер)
- 10752: War (Воєнний)
- 37: Western (Вестерн)

You MUST respond with ONLY a valid JSON object (no markdown, no explanation), with these fields:
{
  "keywords": "string — English keywords for the /discover/movie endpoint's 'with_keywords' or general theme",
  "genre_ids": [list of integer TMDB genre IDs],
  "year_from": integer or null,
  "year_to": integer or null,
  "vote_average_gte": float or null (minimum rating, e.g. 6.5),
  "specific_titles": ["list of specific movie titles to search for, if the user mentioned any or you think of hi
  "sort_by": "string — one of: popularity.desc, vote_average.desc, primary_release_date.desc, revenue.desc"
}

IMPORTANT RULES:
1. If the user describes a specific well-known movie or franchise (by plot, characters, setting), always include
2. Keep keywords SHORT and thematic — 2-4 words maximum.
```

Рисунок 3.4 – Фрагмент системного промпту для пошуку фільмів

Промпт для книг описує синтаксис спеціальних операторів Google Books API – intitle:, inauthor:, subject: – та правила формування полів search_query, subject, author, language та order_by. Фрагмент системного промпту для книг наведено на рисунку 3.5.

Промпт для ігор містить перелік жанрових слагів та тегів RAWG API, а також ідентифікатори платформ та інструкції щодо формування полів search_query, genres, tags, platforms, ordering та metacritic. Фрагмент системного промпту для ігор наведено на рисунку 3.6.

```
Google Books API query supports special keywords:
- intitle: – search in title
- inauthor: – search by author
- subject: – search by category/subject

Common subjects: fiction, nonfiction, science fiction, fantasy, mystery, romance, thriller, horror,

You MUST respond with ONLY a valid JSON object (no markdown, no explanation), with these fields:
{
  "search_query": "string – main search query in English for the 'q' parameter",
  "subject": "string – book subject/category",
  "author": "string – author name if specified or relevant, otherwise empty string",
  "language": "string – always empty string",
  "order_by": "string – 'relevance' or 'newest'"
}

IMPORTANT RULES:
1. If the user describes a specific well-known book, set search_query to the EXACT book title.
```

Рисунок 3.5 – Фрагмент системного промпту для пошуку книг

```
RAWG Genre slugs for reference:
action, indie, adventure, rpg, strategy, shooter, casual, simulation, puzzle, arcade, platformer,

RAWG common tags:
singleplayer, multiplayer, atmospheric, dark, story-rich, open-world, sci-fi, fantasy, horror, sur

Platform IDs:
- 4: PC
- 187: PlayStation 5
- 18: PlayStation 4
- 1: Xbox One
- 186: Xbox Series S/X
- 7: Nintendo Switch
- 3: iOS
- 21: Android

You MUST respond with ONLY a valid JSON object (no markdown, no explanation), with these fields:
{
  "search_query": "string – main search query in English",
  "genres": "string – comma-separated genre slugs, e.g. 'action,rpg'",
  "tags": "string – comma-separated tag slugs, e.g. 'atmospheric,dark,story-rich'",
  "platforms": "string – comma-separated platform IDs, e.g. '4,18' or empty for all",
  "ordering": "string – one of: -rating, -released, -added, -metacritic, name",
  "metacritic": "string – range like '70,100' for minimum-maximum metacritic score, or empty"
```

Рисунок 3.6 – Фрагмент системного промпту для пошуку ігор

Процес аналізу запиту реалізований у модулі services/llm_service.py. Функція analyze_query формує повідомлення для моделі, що складається з трьох частин: системного промпту відповідної категорії, тексту запиту користувача та контексту профілю – останніх пошукових запитів і високо оцінених елементів. Якщо користувач задав додаткові фільтри через UI, вони також додаються до повідомлення у структурованому вигляді.

Відповідь моделі парситься як JSON-об'єкт. Для забезпечення стійкості до варіативності формату відповіді реалізована попередня обробка: видалення маркерів коду у форматі json якщо модель їх додає, та очищення пробільних символів. У разі отримання невалідного JSON або помилки звернення до API активується механізм фолбеку, що формує базові параметри пошуку безпосередньо з тексту запиту користувача без залучення LLM.

Залежно від обраної медіа-категорії LLM генерує параметри різної структури. Для фільмів модель повертає об'єкт із полями keywords, genre_ids, specific_titles, year_from, year_to, vote_average_gte та sort_by. Наявність поля specific_titles є ключовою особливістю: коли модель розпізнає конкретну відому назву у запиті користувача, вона включає її до цього списку, що дозволяє системі виконати прямий пошук за назвою та забезпечити вищу точність результатів.

Для книг модель повертає поля search_query, subject, author та order_by. Окремою функцією є правило щодо мови запиту: промпт містить явну інструкцію завжди генерувати англомовний search_query незалежно від мови введення користувача, оскільки Google Books API значно краще обробляє англомовні запити. У системі реалізована додаткова функція перекладу результатів українською мовою через окремий виклик LLM. Функція translate_items_to_ukrainian у модулі llm_service.py отримує список книг з їхніми назвами та описами, формує запит до моделі на переклад та повертає оновлені об'єкти з українськими текстами. Переклад застосовується лише до результатів пошуку книг і не впливає на фільми та ігри.

Для ігор модель повертає поля search_query, genres, tags, platforms, ordering та metacritic. Важливою особливістю є розмежування між конкретним пошуком за назвою та жанровим пошуком: при наявності search_query система не передає genres та tags до RAWG API, оскільки їх одночасне використання призводить до порожніх результатів.

Приклади перетворення природномовних запитів у параметри пошуку для різних медіа-категорій наведено у таблиці 3.1.

Таблиця 3.1 – Приклади перетворення запитів у параметри пошуку

Запит	Категорія	Згенеровані параметри
«гра про дівчину археологиню на острові Яматай»	Ігри	"search_query": "Tomb Raider", "genres": "action,adventure", "tags": "story-rich,exploration"
«книга про хлопця зі шрамом блискавкою»	Книги	"search_query": "Harry Potter", "subject": "fantasy", "author": "Rowling", "order_by": "relevance"
«фільм про світ де час це основна валюта»	Фільми	"keywords": "time as currency", "specific_titles": ["In Time"], "sort_by": "popularity.desc"

Використання LLM як семантичного посередника між природномовним запитом та зовнішніми API є ключовою інновацією розробленої системи. Такий підхід дозволяє системі розуміти описові, метафоричні та порівняльні запити, що є недосяжним для класичних методів пошуку за ключовими словами.

3.3 Реалізація серверної частини системи

Серверна частина рекомендаційної системи реалізована на основі фреймворку FastAPI мовою програмування Python. FastAPI було обрано завдяки підтримці асинхронного програмування, автоматичній валідації даних через Pydantic, вбудованій документації OpenAPI та високій продуктивності порівняно з іншими Python-фреймворками. Сервер запускається через ASGI-сервер Uvicorn та обслуговує як API-запити від фронтенду, так і роздачу статичних файлів клієнтської частини.

Серверна частина системи організована у вигляді модульної структури директорій. Схему структури бекенду наведено на рисунку 3.7.

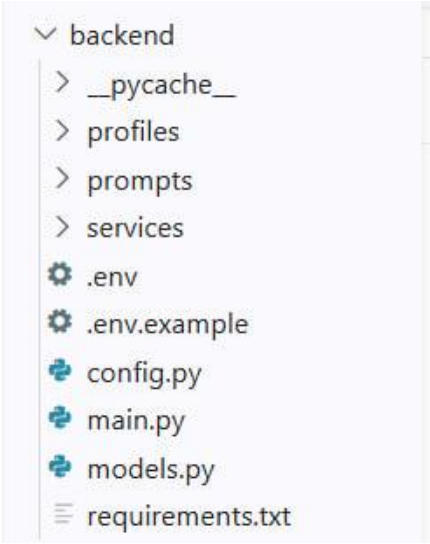


Рисунок 3.7 – Структура директорії серверної частини системи

Кореневий модуль main.py містить ініціалізацію FastAPI-застосунку, налаштування CORS-політики та визначення всіх API-ендпоінтів. Директорія services містить модулі для взаємодії із зовнішніми сервісами та внутрішню бізнес-логіку системи. Директорія prompts містить текстові файли системних промптів для кожної медіа-категорії. Файл models.py визначає Pydantic-моделі для валідації вхідних та вихідних даних. Файл config.py забезпечує завантаження конфігураційних параметрів із файлу .env. Директорія profiles використовується для зберігання JSON-файлів профілів користувачів.

Головний модуль застосунку (main.py) виконує роль центрального координатора всіх компонентів системи. Після ініціалізації FastAPI-застосунку та налаштування CORS-middleware модуль визначає п'ять основних ендпоінтів. Основний ендпоінт POST /api/recommend реалізує повний pipeline обробки запиту: отримання контексту профілю користувача, аналіз запиту через LLM, звернення до зовнішнього API відповідної

категорії, формування відсотків схожості, збереження запиту в історії та збереження статистики популярного контенту. Фрагмент коду основного ендпоінту наведено на рисунку 3.8.

```python
@app.post("/api/recommend", response_model=RecommendResponse)
async def recommend(request: RecommendRequest):

    logger.info(
        f"Новий запит: media_type={request.media_type}, "
        f"user={request.user_id}, query='{request.query[:50]}...'"
    )

    try:
        user_context = get_profile_context(request.user_id)
        logger.info(f"Контекст профілю: {user_context[:100]}...")

        llm_params = await llm_service.analyze_query(
            query=request.query,
            media_type=request.media_type,
            user_context=user_context,
            filters=request.filters,
        )
        logger.info(f"LLM параметри: {llm_params}")

        if request.media_type == "movies":
            items = await tmdb_service.search_movies(llm_params)
        elif request.media_type == "books":
            items = await books_service.search_books(llm_params)
```

Рисунок 3.8 – Реалізація основного ендпоінту рекомендацій

Алгоритм формування відсотка схожості реалізований окремо для кожної медіа-категорії. Для фільмів базовий відсоток становить 97% для об'єктів із поля specific_titles та 72% для результатів discover-пошуку, з корекцією на рейтинг TMDB. Для книг та ігор базовий відсоток становить 93% при збігу назви з пошуковим запитом та 72% в інших випадках. Фінальний список рекомендацій сортується за спаданням відсотка схожості.

Модуль конфігурації (config.py) реалізований на основі бібліотеки pydantic-settings та забезпечує типізоване завантаження змінних середовища з файлу .env. Клас Settings визначає такі параметри: OPENROUTER_API_KEY для доступу до LLM, TMDB_API_KEY для TMDB API, RAWG_API_KEY для RAWG API, GOOGLE_BOOKS_API_KEY для Google Books API, а також шляхи до директорій профілів та промптів. При завантаженні модуля автоматично

створюється директорія для зберігання профілів користувачів якщо вона відсутня.

Модуль models.py визначає Pydantic-моделі для валідації всіх вхідних та вихідних даних системи. Модель RecommendRequest описує структуру запиту на рекомендації та містить поля query, media_type, user_id та filters. Модель MediaItem є уніфікованим представленням медіа-об'єкта незалежно від категорії та містить поля id, title, genre, author, description, image_url, rating та year. Модель RecommendationItem об'єднує MediaItem із відсотком схожості similarity_percent. Модель LikeRequest описує запит на збереження оцінки користувача. Модель UserProfile визначає структуру профілю користувача з полями search_history, likes та favorite_genres.

Модуль llm_service.py реалізує взаємодію з мовною моделлю через OpenRouter API. Клієнт OpenAI ініціалізується лінivo – лише при першому зверненні – з кастомним base_url для OpenRouter. Системні промпти завантажуються з файлів та кешуються у словнику _prompts_cache для уникнення повторного читання файлів. Основна функція analyze_query формує повідомлення для моделі, що включає системний промпт, запит користувача, контекст профілю та додаткові фільтри з UI. Функція translate_items_to_ukrainian реалізує переклад результатів пошуку книг через окремий виклик LLM.

Модуль tmdb_service.py реалізує двоетапну стратегію пошуку фільмів: спочатку виконується прямий пошук за назвами з поля specific_titles, потім – discover-пошук за жанрами та фільтрами. Результати об'єднуються з дедуплікацією за ідентифікатором та назвою. Після отримання результатів застосовується фільтрація за мінімальним рейтингом та кількістю голосів для відсіювання маловідомих або нерелевантних об'єктів.

Модуль books_service.py реалізує пошук через Google Books API з формуванням запиту відповідно до наявних параметрів: при наявності автора використовується оператор inauthor:, при наявності лише жанру –

оператор subject:. Реалізований механізм повторних спроб з експоненційним відступом для обробки помилок 429 та 503. Результати проходять багаторівневу фільтрацію: за мовою видання, за категоріями (виключаються навчальні та дитячі видання), за ключовими словами у назві (виключаються збірники, путівники, щоденники) та дедуплікацію за нормалізованою назвою.

Модуль rawg_service.py реалізує пошук ігор через RAWG API з розмежуванням між конкретним пошуком за назвою та жанровим пошуком. При наявності search_query параметри genres та tags не передаються до API оскільки їх одночасне використання призводить до порожніх результатів. Після отримання результатів застосовується фільтрація за жанром та мінімальним рейтингом.

Модуль profile_service.py реалізує управління профілями користувачів без використання бази даних – дані зберігаються у JSON-файлах у директорії profiles. Кожен профіль ідентифікується унікальним рядковим ідентифікатором користувача, що генерується на фронтенді та зберігається у localStorage браузера. Функція add_search_to_history додає запит до історії та обмежує її останніми 50 записами. Функція add_like зберігає оцінку користувача разом із назвою та зображенням об'єкта для подальшого відображення в профілі. Функція get_profile_context формує текстовий опис профілю для передачі до LLM, що включає останні пошукові запити та ідентифікатори високо оцінених елементів. Функція add_results_to_stats оновлює глобальний файл статистики stats.json для формування сторінки популярного контенту.

Модульна архітектура серверної частини забезпечує чіткий розподіл відповідальності між компонентами системи та спрощує подальше розширення функціональності – додавання нових медіа-категорій або заміну зовнішніх API потребує мінімальних змін у решті компонентів системи.

3.4 Реалізація клієнтської частини системи

Клієнтська частина рекомендаційної системи реалізована у вигляді багатосторінкового веб-застосунку без використання JavaScript-фреймворків. Такий підхід забезпечує мінімальні залежності, швидке завантаження сторінок та простоту розгортання. Інтерфейс побудований на основі Bootstrap 5.3 для адаптивної розмітки та Bootstrap Icons для іконографії. Усі сторінки роздаються безпосередньо серверною частиною FastAPI через статичний файловий сервер.

Клієнтська частина складається з семи HTML-сторінок, кожна з яких відповідає окремому розділу системи. Структура HTML-сторінок наведено на рисунку 3.9.

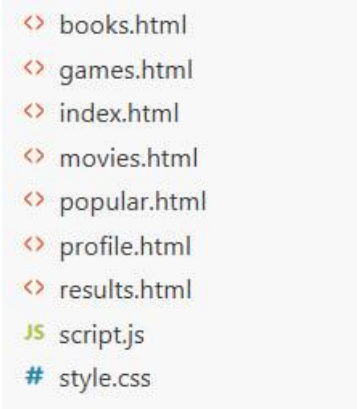


Рисунок 3.9 – Структура HTML-сторінок

Сторінка index.html є головною сторінкою застосунку та містить загальний опис системи, форму зворотного зв'язку та навігаційне меню. Сторінки movies.html, books.html та games.html реалізують інтерфейс пошуку для відповідних медіа-категорій і містять текстову галузь для введення природномовного запиту та блок додаткових фільтрів. Сторінка results.html відображає результати рекомендацій після виконання запиту. Сторінка popular.html містить таблиці популярного контенту та

інтерактивний графік аналітики. Сторінка profile.html реалізує кабінет користувача з вкладками для перегляду обраного контенту, історії оцінок та налаштувань.

Вся клієнтська логіка зосереджена у файлі script.js та реалізована у вигляді класу AIRecommendationClient. Клас ініціалізується при завантаженні сторінки та виконує такі функції: генерація унікального ідентифікатора користувача при першому відвідуванні та його збереження у localStorage, підключення обробників подій до кнопок відповідно до поточної сторінки, збір значень фільтрів із UI-елементів, відправка запитів до бекенду та перенаправлення на сторінку результатів.

Метод handlePageSubmit зчитує текст запиту із текстової області .ai-textarea, збирає значення всіх активних фільтрів зі select-елементів сторінки та передає їх до методу getRecommendations. Під час виконання запиту кнопка «Підібрати» блокується та відображає індикатор завантаження для інформування користувача про процес обробки запиту.

Метод showRecommendations зберігає отримані рекомендації у localStorage у форматі JSON разом із типом медіа, текстом запиту та часовою міткою, після чого перенаправляє користувача на сторінку results.html. Метод likeItem відправляє асинхронний запит на ендпоінт /api/like при виставленні зірочкової оцінки.

Стилізація інтерфейсу реалізована у файлі style.css із використанням CSS-змінних для підтримки світлої та темної теми. Основні змінні визначають кольори фону, панелей, тексту, акцентів та меж елементів. При перемиканні теми клас dark-mode додається до елемента body, що перевизначає значення CSS-змінних для темної палітри. Такий підхід забезпечує миттєве перемикання теми без перезавантаження сторінки та збереження обраної теми між сесіями через localStorage.

Інтерфейс у світлій та темній темах наведено на рисунку 3.10.

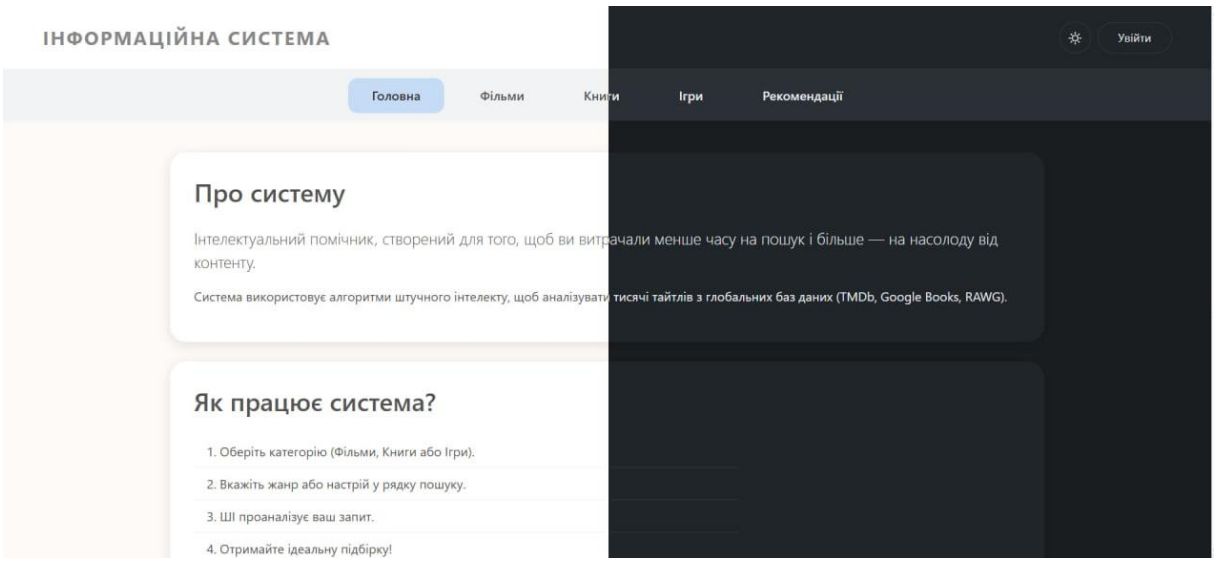


Рисунок 3.10 – Інтерфейс системи у світлій та темній темах

Компонент картки результату .result-card реалізований як flex-контейнер із трьох зон: контейнер постера фіксованого розміру 130×190 пікселів, блок інформації про медіа-об'єкт та блок дій користувача з зірочками оцінювання та кнопкою додавання до обраного. Сторінка результатів з прикладом рекомендацій наведена на рисунку 3.11.



Рисунок 3.11 – Сторінка результатів рекомендацій

Сторінки пошуку фільмів, книг та ігор мають однакову структуру з двох блоків. Перший блок містить текстову галузь для введення природномовного запиту з підказкою-плейсхолдером. Другий блок містить

додаткові фільтри та кнопку запуску пошуку. Набір фільтрів відрізняється залежно від категорії: для фільмів доступні фільтри року виходу та мінімального рейтингу, для книг – рейтинг читачів, для ігор – платформа та режим гри.

Сторінки пошуку фільмів, книг та ігор наведено на рисунках 3.12, 3.13 та 3.14 відповідно.

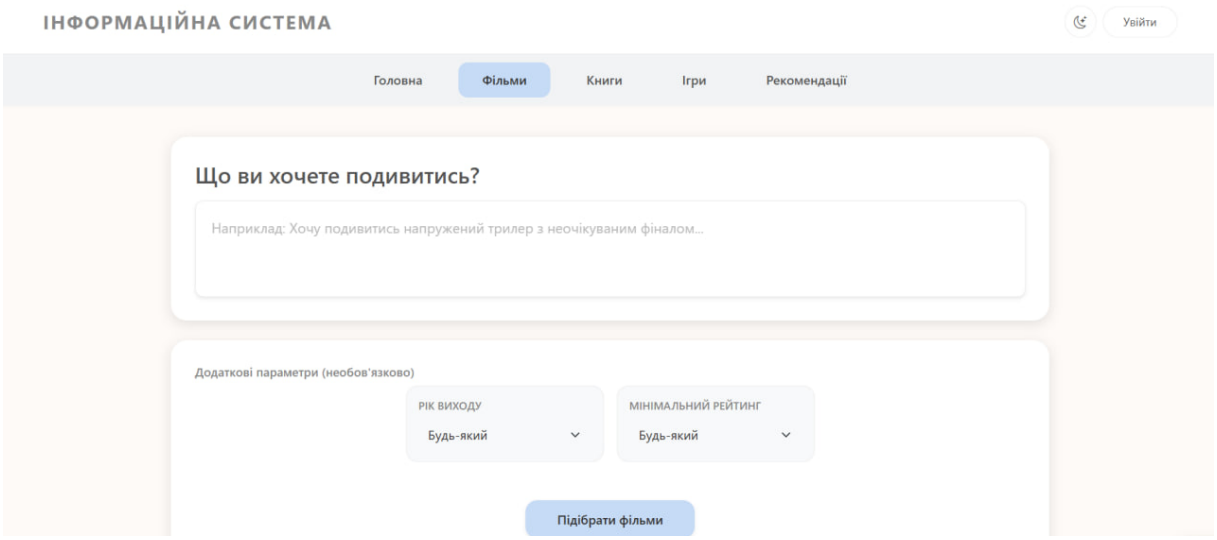


Рисунок 3.12 – Сторінка пошуку фільмів

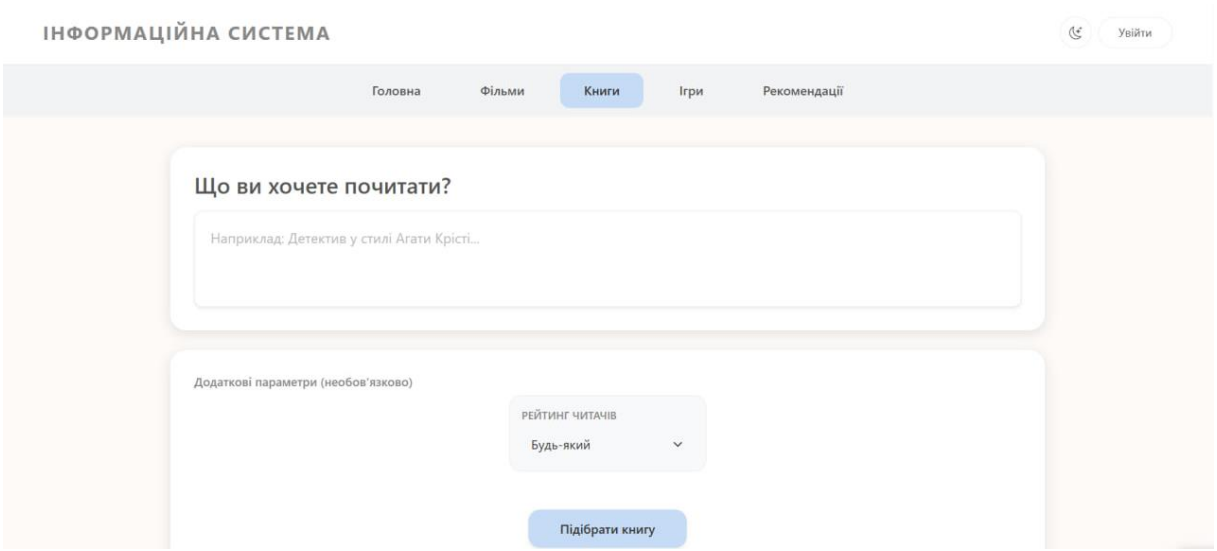


Рисунок 3.13 – Сторінка пошуку книг

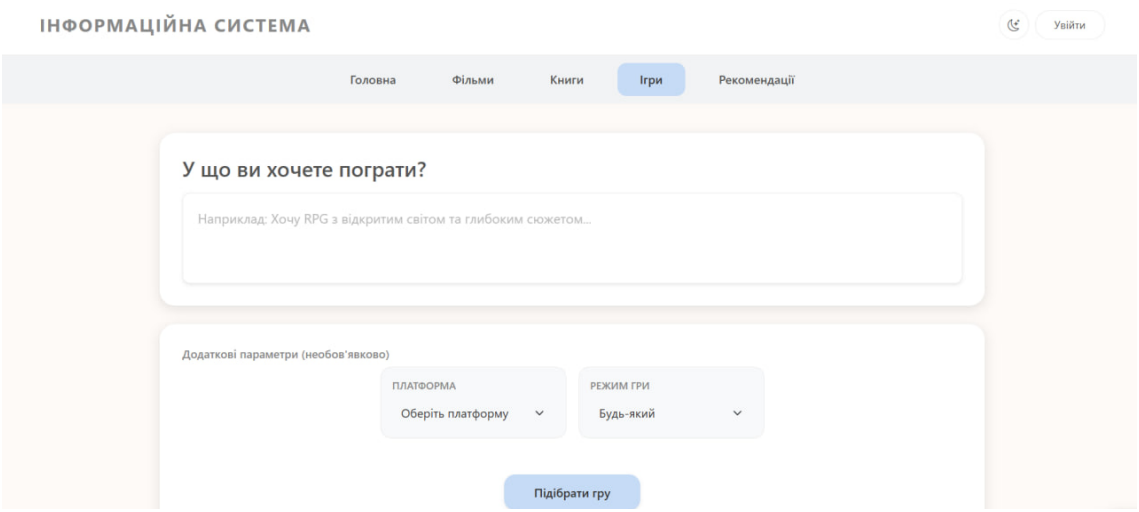


Рисунок 3.14 – Сторінка пошуку ігор

Сторінка popular.html відображає рейтинги найчастіше рекомендованого контенту окремо для фільмів, книг та ігор. Дані завантажуються асинхронно з ендпоінту /api/popular при відкритті сторінки. Кожна таблиця містить порядковий номер, назву запиту та кількість звернень. При натисканні на рядок таблиці оновлюється інтерактивний лінійний графік, побудований на основі бібліотеки Chart.js, що відображає умовну динаміку популярності обраного контенту протягом тижня. Сторінку популярного контенту наведено на рисунках 3.15 та 3.16.

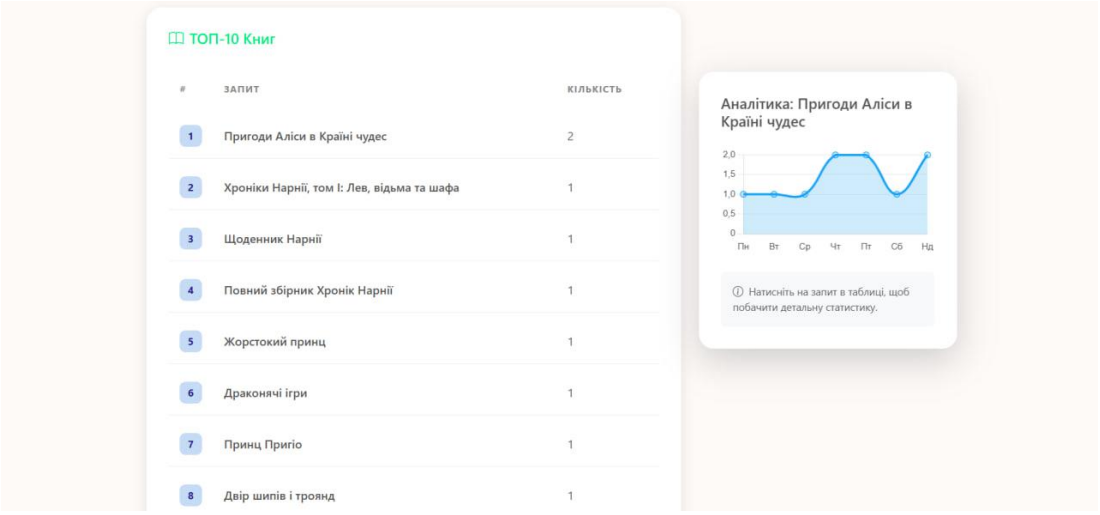


Рисунок 3.15 – Сторінка популярного контенту топ фільмів

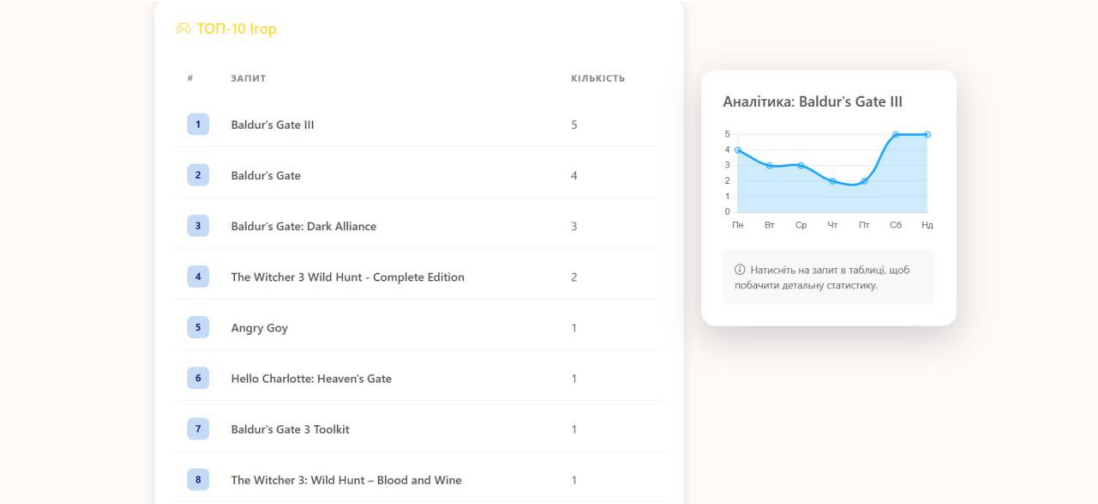


Рисунок 3.16 – Сторінка популярного контенту топ ігор

Адаптивність інтерфейсу забезпечується сіткою Bootstrap з використанням класів col-md-3, col-lg-4 та col-lg-8 для перебудови макету залежно від ширини екрана. Сторінка популярного у меншому розмірі наведена на рисунку 3.17.

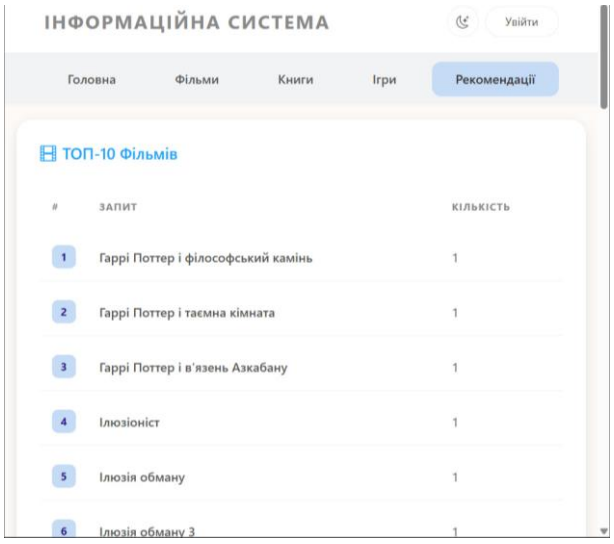


Рисунок 3.17 – Сторінка популярного контенту у меншому розмірі

На мобільних пристроях блоки фільтрів переходять у вертикальне розташування, а таблиця на сторінці популярного контенту отримує

горизонтальне прокручування через клас table-responsive. Графік на сторінці популярного контенту є адаптивним завдяки властивості responsive: true бібліотеки Chart.js.

Клієнтська частина системи забезпечує інтуїтивно зрозумілий інтерфейс для взаємодії з рекомендаційною системою, підтримує світлу та темну теми, є адаптивною для різних розмірів екрана та реалізує повний цикл взаємодії користувача – від введення запиту до перегляду та оцінювання результатів.

3.5 Збереження даних та персоналізація рекомендацій

У розробленій системі реалізовано гібридний підхід до збереження даних користувача: клієнтські дані зберігаються у localStorage браузера, тоді як дані профілю та статистика використання зберігаються у JSON-файлах на серверній стороні. Такий підхід дозволяє забезпечити персоналізацію рекомендацій без використання повноцінної бази даних, що спрощує розгортання системи та знижує вимоги до інфраструктури.

Ідентифікатор формується як рядок виду user_ з додаванням дев'яти випадкових символів у форматі base36, наприклад user_b0nogo1n1. Згенерований ідентифікатор зберігається у localStorage браузера та автоматично використовується в усіх наступних запитах до API. Це забезпечує стабільну ідентифікацію користувача в межах одного браузера без необхідності реєстрації та введення облікових даних.

localStorage браузера використовується для зберігання трьох типів даних. По-перше, ідентифікатор користувача userId зберігається при першому відвідуванні та використовується протягом усіх наступних сесій. По-друге, обрана тема оформлення theme зберігається при перемиканні між світлою та темною темами та застосовується при кожному відкритті сторінки. По-третє, список обраного контенту wishlist зберігає масив об'єктів із полями id, title, image_url, media_type та added – часовою міткою

додавання. Дані обраного зберігаються виключно на стороні клієнта та не передаються на сервер, що забезпечує їх доступність без мережевого з'єднання.

Крім того, результати останнього пошуку зберігаються у localStorage під ключем lastRecommendations для передачі між сторінками – зі сторінки пошуку на сторінку результатів. Об'єкт містить масив рекомендацій, тип медіа-категорії, текст запиту та часову мітку.

Кожен успішно виконаний пошуковий запит автоматично зберігається в історії користувача на серверній стороні. Функція add_search_to_history модуля profile_service.py додає до профілю запис із трьох полів: текст запиту query, тип медіа-категорії media_type та часова мітка timestamp у форматі ISO 8601. Для обмеження розміру файлу профіль зберігає не більше 50 останніх записів – старіші записи автоматично видаляються. Збережена історія запитів відображається у вкладці «Історія запитів» кабінету користувача та передається до LLM як контекст профілю при формуванні нових рекомендацій. Вкладку «Історія запитів» наведено на рисунку 3.18.

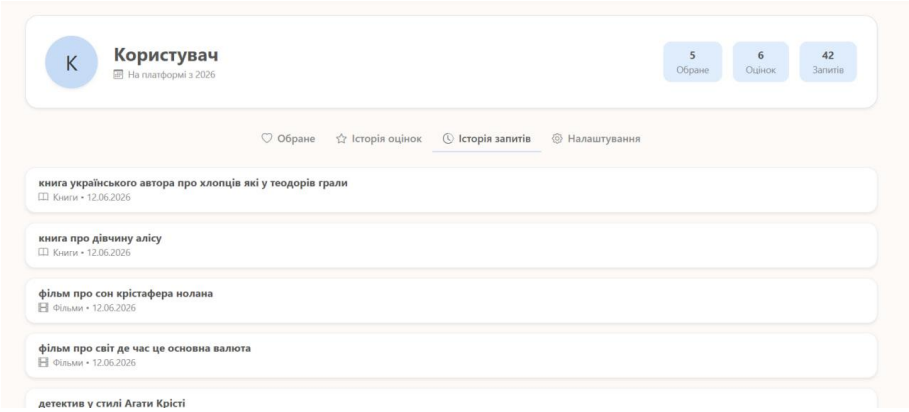


Рисунок 3.18 – Вкладка «Історія запитів» кабінету користувача

Система підтримує п'ятизіркове оцінювання рекомендованих об'єктів. При виставленні оцінки фронтенд відправляє POST-запит на ендпоінт

/api/like із параметрами user_id, item_id, media_type, rating, title та image_url. Збереження назви та зображення об'єкта разом із оцінкою дозволяє відображати повноцінні картки у вкладці «Історія оцінок» кабінету користувача без додаткових запитів до зовнішніх API. Функція add_like зберігає оцінку у профілі користувача з часовою міткою. У разі повторного оцінювання того самого об'єкта наявний запис оновлюється замість створення дубліката. Профіль зберігає не більше 100 останніх оцінок. Вкладку «Історія оцінок» кабінету користувача наведено на рисунку 3.19.

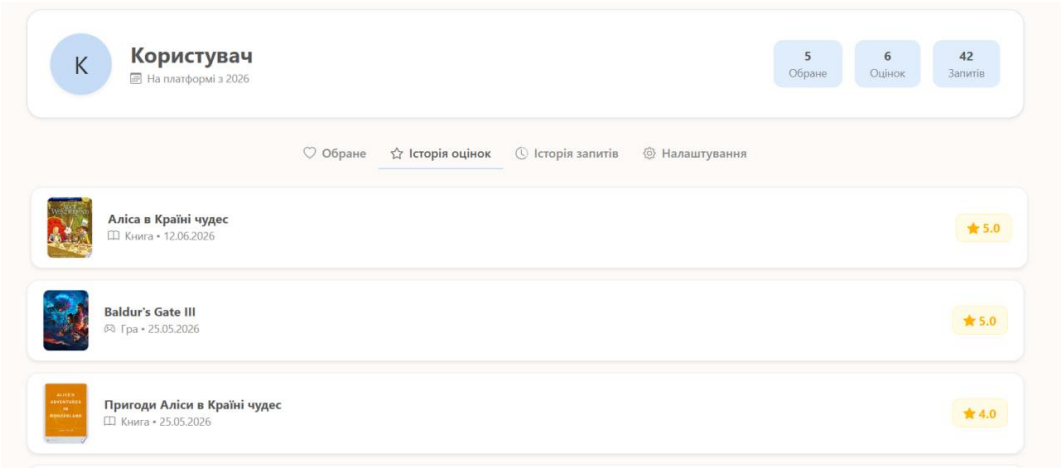


Рисунок 3.19 – Вкладка «Історія оцінок» кабінету користувача

Користувач може додавати рекомендовані об'єкти до списку обраного натисканням кнопки «В бажане» на сторінці результатів. При додаванні об'єкта функція toggleWishlist зберігає у localStorage об'єкт із полями id, title, image_url, media_type та added. При повторному натисканні об'єкт видаляється зі списку. Вкладка «Обране» кабінету користувача динамічно завантажує дані з localStorage при відкритті сторінки та відображає картки збережених об'єктів із зображенням, назвою, типом медіа та датою додавання. Кнопка видалення дозволяє прибрати об'єкт зі списку безпосередньо з кабінету користувача. Вкладку «Обране» кабінету користувача наведено на рисунку 3.20.

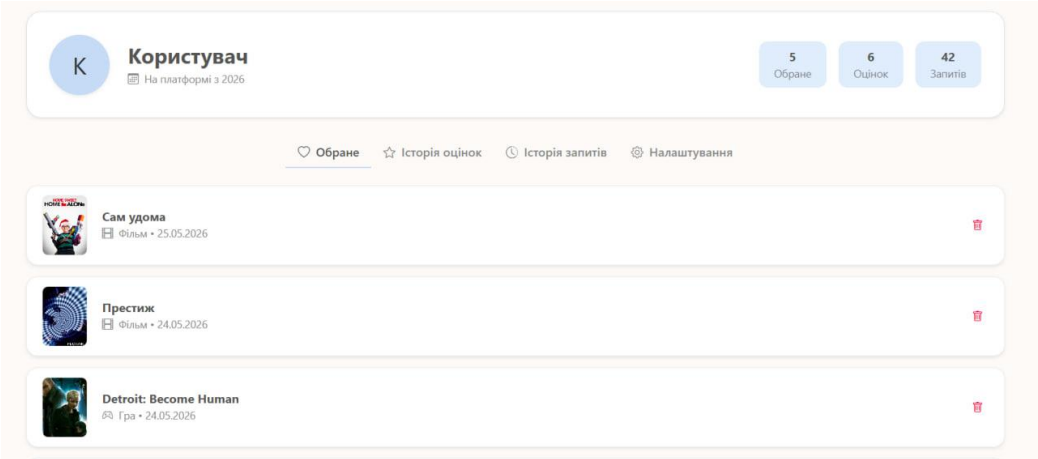


Рисунок 3.20 – Вкладка «Обране» кабінету користувача

Профіль користувача зберігається у JSON-файлі у директорії profiles/ під ім'ям {user_id}.json. Структуру файлу профілю наведено на рисунку 3.21.

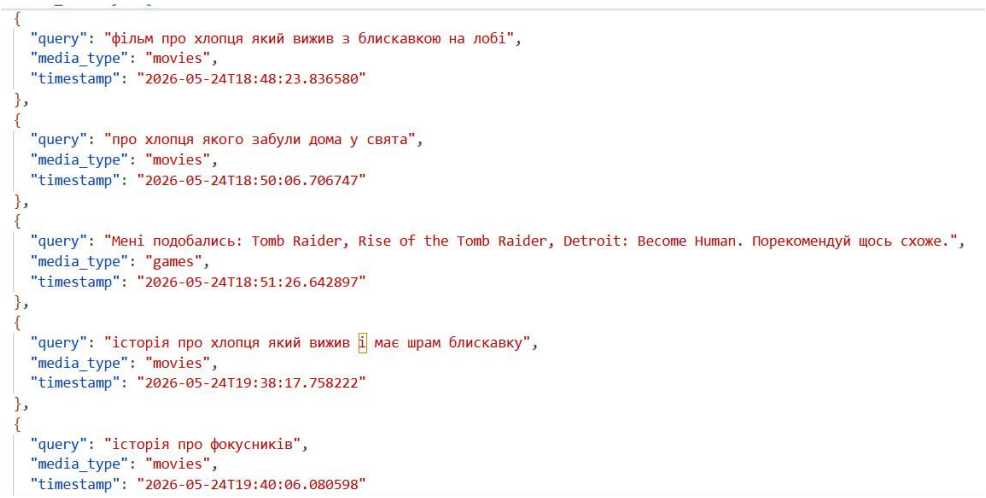


Рисунок 3.21 – Структура JSON-файлу профілю користувача

Персоналізація рекомендацій реалізована через передачу контексту профілю до LLM при кожному запиті. Функція get_profile_context формує текстовий опис профілю, що включає останні п'ять пошукових запитів та ідентифікатори об'єктів з оцінкою не нижче чотирьох зірок. Мовна модель

враховує цей контекст при генерації параметрів пошуку, що дозволяє системі адаптувати рекомендації до вподобань конкретного користувача.

Додатково у системі реалізована функція персональних рекомендацій на основі профілю. Кнопка «Підібрати на основі мого профілю» у кабінеті користувача формує запит із назв трьох найвище оцінених об'єктів та визначає найбільш часто використовувану медіа-категорію, після чого відправляє запит до системи для отримання персоналізованих рекомендацій без введення тексту запиту користувачем.

Для формування сторінки популярного контенту система веде глобальну статистику у файлі stats.json у директорії profiles/. При кожному успішному пошуку назви трьох перших рекомендованих об'єктів записуються до файлу статистики з лічильником кількості поява. Ендпоінт /api/popular зчитує файл статистики та повертає десять найпопулярніших об'єктів окремо для кожної медіа-категорії, відсортованих за спаданням лічильника.

Гібридний підхід до збереження даних – localStorage для клієнтських даних та JSON-файли для серверних – забезпечує необхідний рівень персоналізації без ускладнення інфраструктури системи. Збережені дані про вподобання користувача використовуються як для відображення персональної історії у кабінеті користувача, так і для покращення якості рекомендацій через передачу контексту профілю до мовної моделі.

3.6 Демонстрація роботи рекомендаційної системи

У даному підрозділі наведено демонстрацію функціональних можливостей розробленої рекомендаційної системи на основі реальних прикладів пошукових запитів для кожної медіа-категорії. Для кожного прикладу наведено текст запиту, згенеровані LLM параметри пошуку та отримані результати рекомендацій.

При відкритті системи користувач потрапляє на головну сторінку, що містить загальний опис функціональності та навігаційне меню для переходу до розділів фільмів, книг та ігор. Головну сторінку системи наведено на рисунку 3.22.

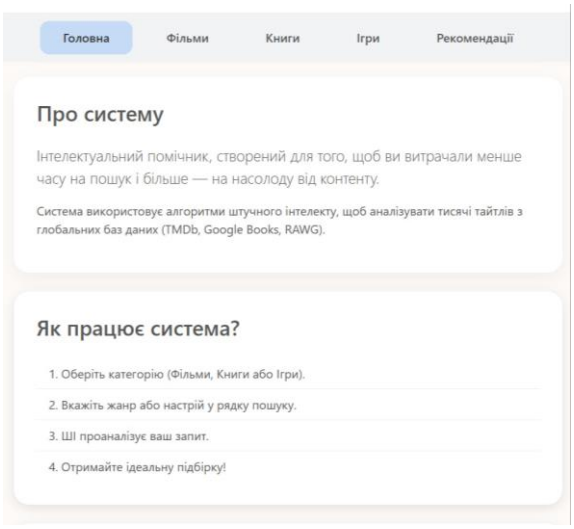


Рисунок 3.22 – Головна сторінка рекомендаційної системи

Сторінка пошуку фільмів дозволяє користувачеві ввести запит у довільній формі та за бажанням вказати додаткові фільтри – рік виходу та мінімальний рейтинг. Розглянемо три характерні приклади запитів різного типу.

Перший приклад демонструє запит із конкретною описовою характеристикою: «детектив по книзі Агати Крісті». LLM розпізнав відомого автора та сформував список конкретних назв фільмів у полі specific_titles, що забезпечило точний пошук за назвами. Результати запиту наведено на рисунку 3.23.

Другий приклад демонструє жанровий запит: «напружений трилер з несподіваним фіналом». У цьому випадку LLM не має конкретних назв для пошуку та формує параметри discover-пошуку за жанрами та ключовими словами. Результати містять популярні трилери відповідно до жанрових параметрів TMDB. Результати запиту наведено на рисунку 3.24.

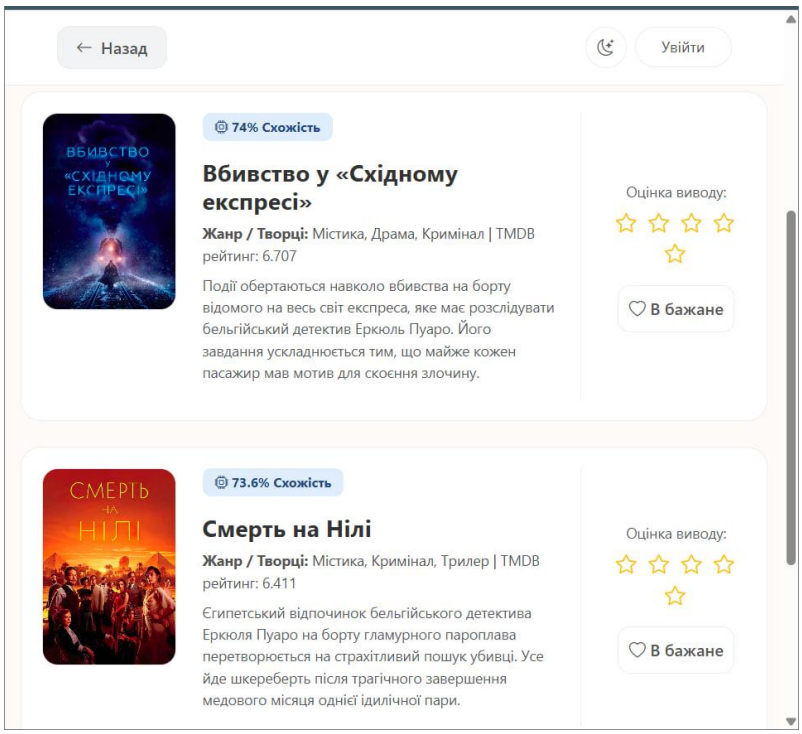


Рисунок 3.23 – Результати пошуку фільмів за запитом «детектив по книзі Агати Крісті»

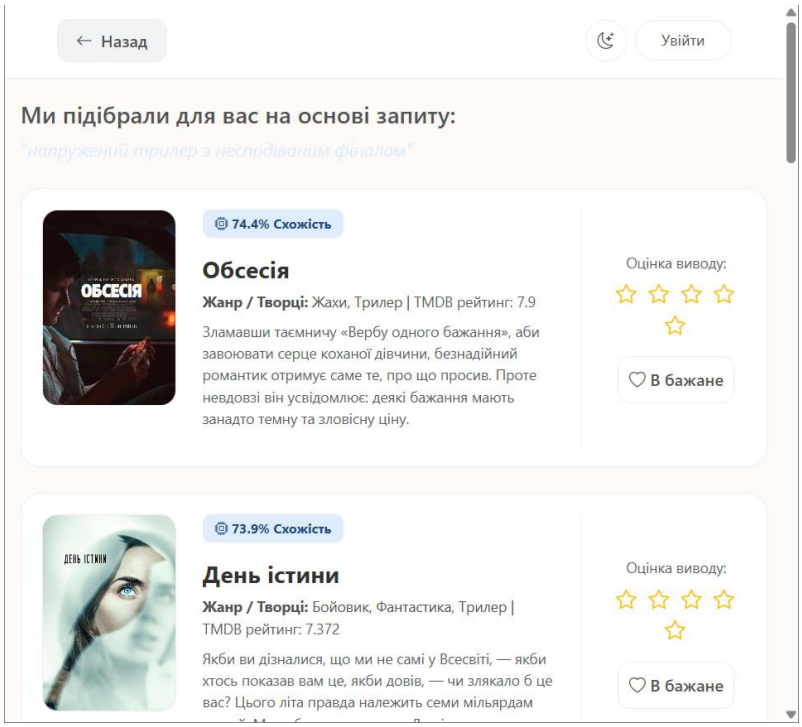


Рисунок 3.24 – Результати пошуку фільмів за жанровим запитом

Третій приклад демонструє запит із активованим фільтром року: «автобіографічний фільм про музиканта» з фільтром «з 2000 по 2010». LLM враховує фільтр UI та встановлює відповідні значення year_from та year_to, що обмежує результати вказаним часовим діапазоном. Результати запиту наведено на рисунку 3.25.

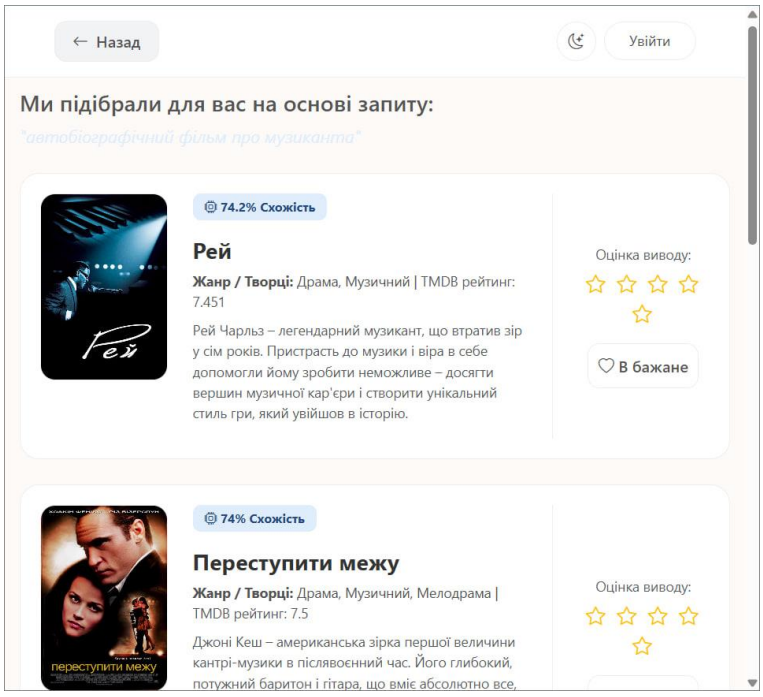


Рисунок 3.25 – Результати пошуку фільмів із застосуванням фільтру року виходу

Сторінка пошуку книг аналогічно дозволяє вводити запити у довільній формі. Особливістю пошуку книг є автоматичний переклад результатів українською мовою через додатковий виклик LLM.

Перший приклад – конкретний описовий запит: «книги про школу де вчаться злі та добрі герої». LLM розпізнав серію «Школа добра і зла» та сформував параметри search_query: "The School for Good and Evil" з автором "Chainani". Результати містять книги серії українською мовою. Результати запиту наведено на рисунку 3.26.

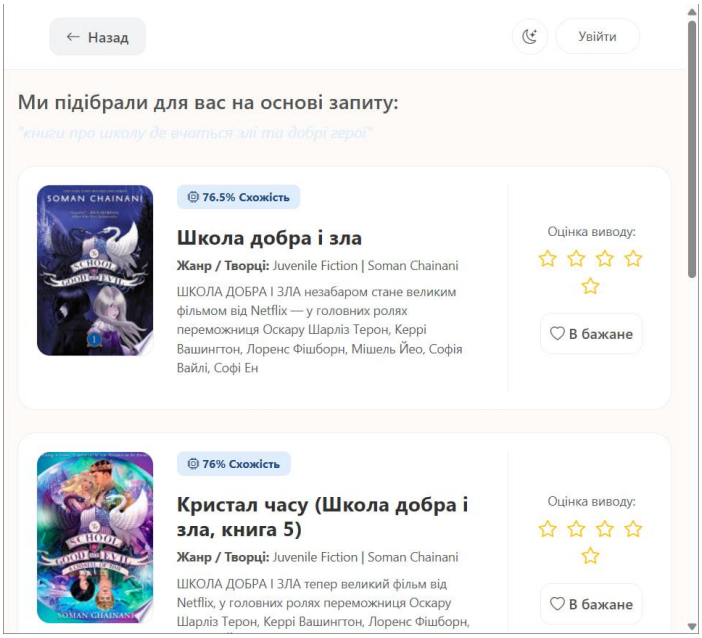


Рисунок 3.26 – Результати пошуку книг за описовим запитом

Другий приклад – запит за автором: «книги від Сари Дж. Маас». LLM коректно визначив автора та сформував параметри для пошуку всіх її книг. Результати містять серії «A Court of Thorns and Roses» та «Throne of Glass». Результати запиту наведено на рисунку 3.27.

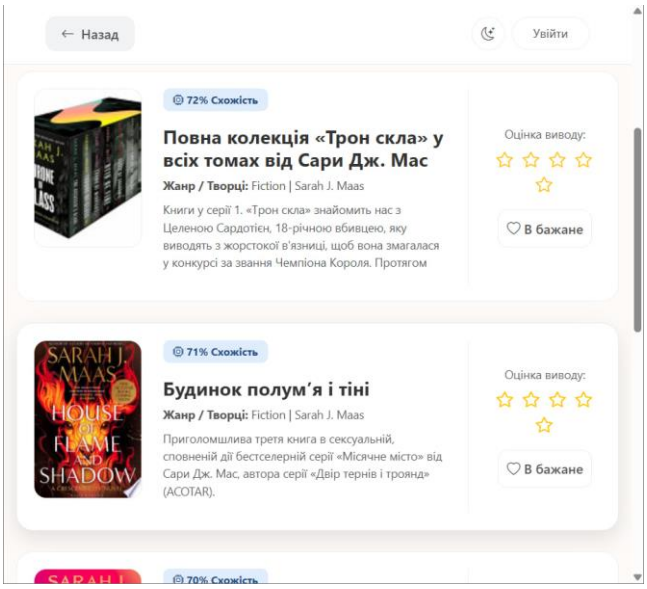


Рисунок 3.27 – Результати пошуку книг за запитом з іменем автора

Третій приклад – жанровий запит: «захоплюючий детектив». LLM сформував параметри search_query: "Sherlock Holmes" із жанром subject: "mystery", що забезпечує повернення класичних детективних творів замість мета-книг про детективний жанр. Результати запиту наведено на рисунку 3.28.

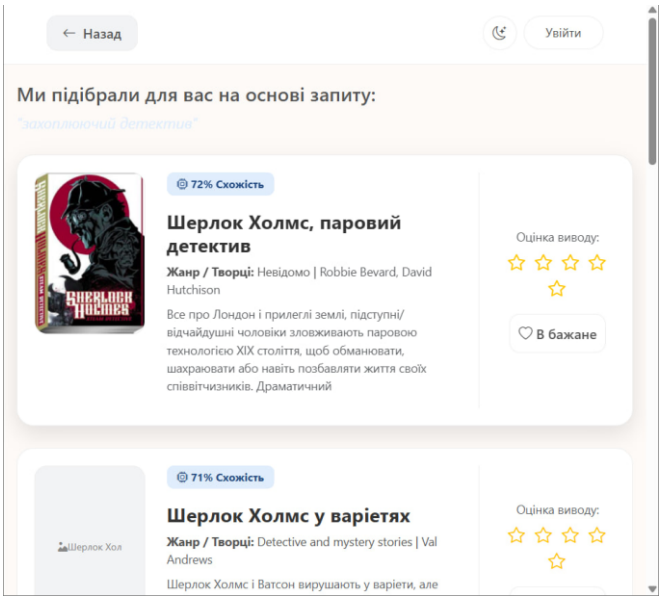


Рисунок 3.28 – Результати пошуку книг за жанровим запитом

Сторінка пошуку ігор додатково містить фільтри платформи та режиму гри, що передаються до RAWG API як параметри пошуку.

Перший приклад – конкретний описовий запит: «гра про дівчину археологиню на острові Яматай». LLM розпізнав серію Tomb Raider та сформував параметри search_query: "Tomb Raider" із жанром "action,adventure". Результати містять ігри серії Tomb Raider. Результати запиту наведено на рисунку 3.29.

Другий приклад – запит із механічною характеристикою: «гра з механікою ДнД». LLM сформував параметри search_query: "Baldur's Gate 3" із жанром "rpg", що забезпечило повернення ігор серії Baldur's Gate та схожих RPG. Результати запиту наведено на рисунку 3.30.

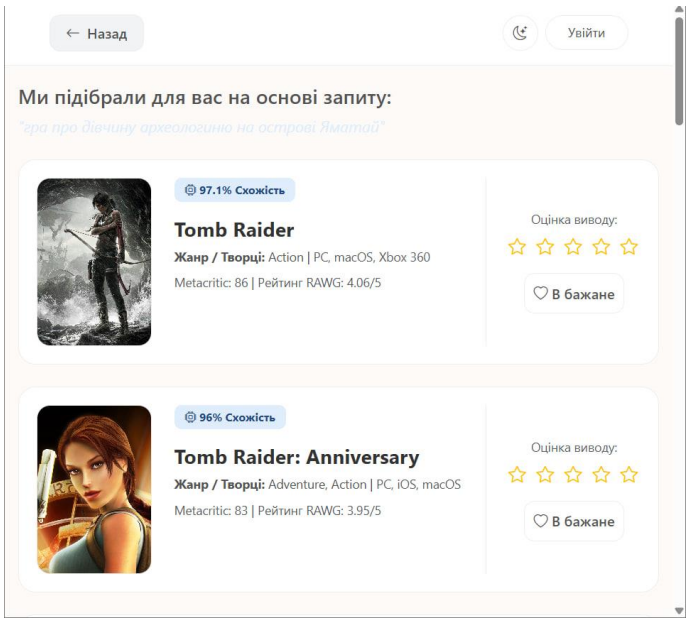


Рисунок 3.29 – Результати пошуку ігор за описовим запитом

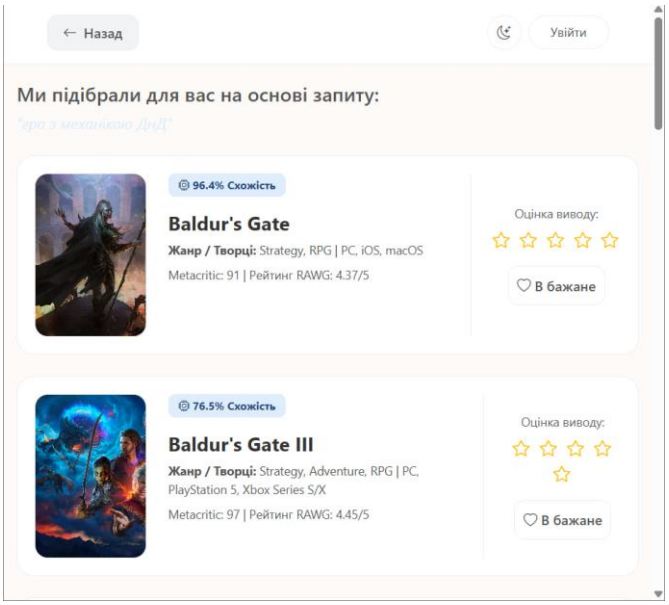


Рисунок 3.30 – Результати пошуку ігор за запитом із механічною характеристикою

Третій приклад – жанровий запит: «детективна гра». LLM сформував параметри search_query: "Disco Elysium" із тегами "story-rich,atmospheric", що забезпечило повернення детективних пригодницьких ігор на чолі з Disco Elysium та L.A. Noire. Результати запиту наведено на рисунку 3.31.

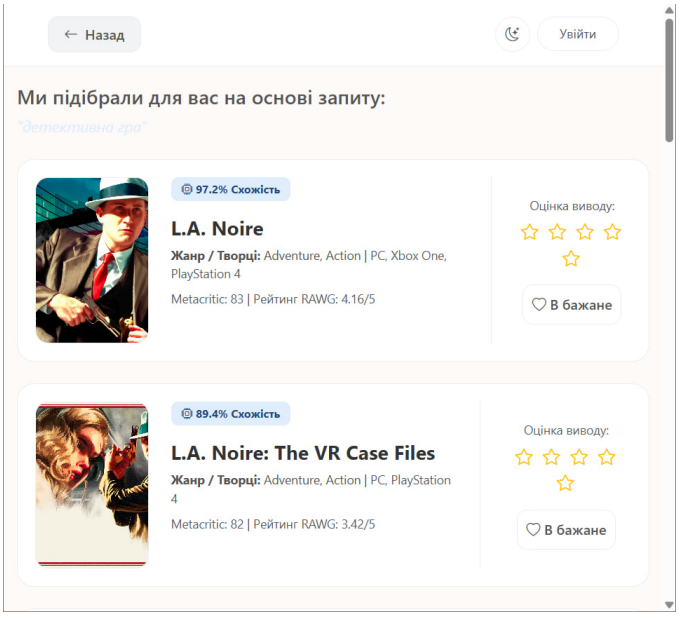


Рисунок 3.31 – Результати пошуку ігор за жанровим запитом

Після отримання результатів користувач може оцінити кожну рекомендацію за п'ятизірковою шкалою та додати об'єкт до списку обраного. При виставленні оцінки дані передаються на сервер та зберігаються у профілі користувача для подальшої персоналізації. Приклад оцінювання результатів наведено на рисунку 3.32.

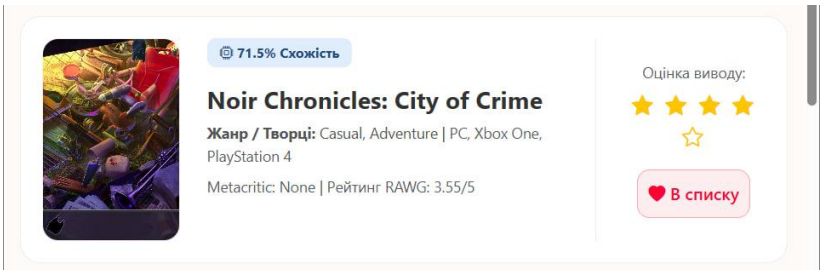


Рисунок 3.32 – Оцінювання рекомендацій користувачем

Кабінет користувача містить три вкладки. Вкладка «Обране» відображає список збережених об'єктів із зображеннями та датами додавання, вкладка «Історія оцінок» – оцінені об'єкти із зірковим рейтингом, вкладка «Налаштування» – форму редагування профілю та

перемикач теми. Кабінет користувача також містить кнопку «Підібрати на основі мого профілю», що формує персоналізований запит на основі збережених вподобань. Підбір на основі профілю наведено на рисунку 3.33.

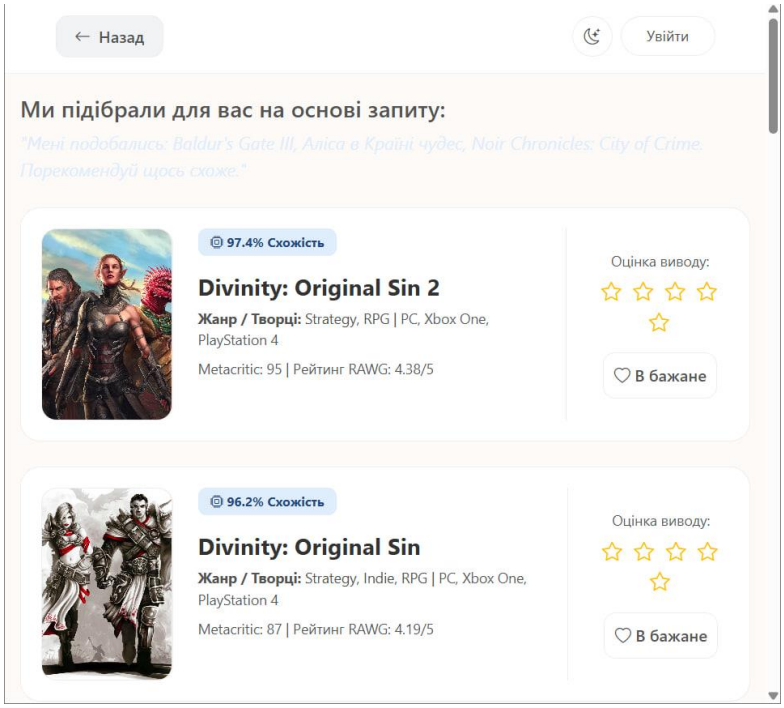


Рисунок 3.33 – Результати пошуку на основі профілю користувача

Проведена демонстрація підтвердила здатність системи коректно обробляти запити трьох основних типів: конкретні описові запити з розпізнаванням відомих назв, жанрові запити з пошуком за тематичними параметрами та запити із зазначенням автора або франшизи. Система забезпечує релевантні результати для запитів українською мовою завдяки семантичному аналізу LLM, що перетворює природномовний опис на точні параметри пошуку зовнішніх API.

ВИСНОВКИ

У кваліфікаційній роботі розроблено рекомендаційну систему для підбору фільмів, книг та відеоігор на основі аналізу природномовних запитів користувача засобами великих мовних моделей. За результатами виконання роботи можна зробити наступні висновки.

Завдання на кваліфікаційну роботу виконано в повному обсязі. Розроблено та реалізовано рекомендаційну систему, що забезпечує підбір мультимедійного контенту трьох категорій – фільмів, книг та відеоігор – на основі довільних природномовних запитів українською мовою. Система включає серверну частину на основі FastAPI, клієнтський веб-інтерфейс, модуль аналізу запитів на основі LLM, інтеграцію з трьома зовнішніми API та підсистему збереження даних користувача.

У процесі аналізу предметної галузі систематизовано основні підходи до побудови рекомендаційних систем – контентна фільтрація, колаборативна фільтрація та гібридні методи – та визначено їх ключові обмеження: проблему холодного старту, надмірну спеціалізацію та недостатнє семантичне розуміння контенту. Обґрунтовано доцільність застосування LLM-підходу як альтернативи класичним методам для сценаріїв з обмеженими поведінковими даними та необхідністю підтримки природномовного інтерфейсу.

Досліджено методи семантичного аналізу тексту та трансформерні архітектури, що становлять теоретичну основу сучасних рекомендаційних систем. Розглянуто моделі word2vec, BERT, Sentence-BERT та BERT4Rec, визначено їх переваги та обмеження у контексті задач рекомендації мультимедійного контенту. Проаналізовано методи персоналізації рекомендацій та підходи до оцінювання якості рекомендаційних систем із використанням метрик Precision@K, Recall@K, nDCG та diversity.

Реалізований LLM-підхід до аналізу запитів забезпечує коректне опрацювання трьох основних типів запитів: конкретних описових запитів із

розпізнаванням відомих назв, жанрових запитів із формуванням тематичних параметрів пошуку та запитів із зазначенням автора або франшизи. Система коректно обробляє запити українською мовою завдяки семантичному перекладу природномовного опису на параметри зовнішніх API.

Розроблена система реалізує базовий рівень персоналізації рекомендацій через передачу контексту профілю користувача до мовної моделі при кожному запиті. Профіль включає історію останніх пошукових запитів та ідентифікатори високо оцінених об'єктів, що дозволяє системі враховувати вподобання користувача при генерації параметрів пошуку. Додатково реалізована функція формування персональних рекомендацій на основі збережених оцінок без введення текстового запиту.

Матеріали кваліфікаційної роботи можуть бути використані за призначенням як основа для розробки комерційного застосунку рекомендації мультимедійного контенту, а також у навчальному процесі університету.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Bingham N. E. Toffler, Alvin. Future shock. New York: Bantam Books, Inc., 1971 (540 pages). *Science education*. 1972. Vol. 56, no. 3. P. 438–440. URL: https://doi.org/10.1002/sce.3730560328 (date of access: 15.05.2026).

2. Netflix library by the numbers 2024: 589 new originals released and library swells to over 7,000 titles. *What's on Netflix*. URL: https://www.whats-on-netflix.com/news/netflix-library-by-the-numbers-2024 (date of access: 15.05.2026).

3. The 144 most read books of the 2024 reading challenge. *Goodreads*. URL: https://www.goodreads.com/blog/show/2869-the-144-most-read-books-of-the-2024-reading-challenge (date of access: 15.05.2026).

4. Instsnt search | Algolia. *SteamdB*. URL: https://steamdb.info/instantsearch/?refinementList[appType][0]=Game (date of access: 15.05.2026).

5. Recommender systems handbook / ed. by F. Ricci, L. Rokach, B. Shapira. New York, NY : Springer US, 2022. URL: https://doi.org/10.1007/978-1-0716-2197-4 (date of access: 15.05.2026).

6. Gomez-Uribe C. A., Hunt N. The Netflix recommender system. *ACM transactions on management information systems*. 2016. Vol. 6, no. 4. P. 1–19. URL: https://doi.org/10.1145/2843948 (date of access: 15.05.2026).

7. Covington P., Adams J., Sargin E. Deep neural networks for Youtube recommendations. *RecSys '16: tenth ACM conference on recommender systems*, Boston Massachusetts USA. New York, NY, USA, 2016. URL: https://doi.org/10.1145/2959100.2959190 (date of access: 15.05.2026).

8. Current challenges and visions in music recommender systems research / M. Schedl et al. *International journal of multimedia information retrieval*. 2018. Vol. 7, no. 2. P. 95–116. URL: https://doi.org/10.1007/s13735-018-0154-2 (date of access: 15.05.2026).

9. Linden G., Smith B., York J. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet computing*. 2003. Vol. 7, no. 1. P. 76–80. URL: https://doi.org/10.1109/mic.2003.1167344 (date of access: 15.05.2026).

10. Sentiment analysis and innovative recommender system: enhancing goodreads book discovery using hybrid collaborative and content based filtering / L. C. Hui et al. *Lecture notes on data engineering and communications technologies*. Cham, 2024. P. 97–111. URL: https://doi.org/10.1007/978-3-031-59707-7_9 (date of access: 15.05.2026).

11. Sifa R., Drachen A., Bauckhage C. Large-scale cross-game player behavior analysis on steam. *Proceedings of the AAAI conference on artificial intelligence and interactive digital entertainment*. 2021. Vol. 11, no. 1. P. 198–204. URL: https://doi.org/10.1609/aiide.v11i1.12804 (date of access: 15.05.2026).

12. BERT: pre-training of deep bidirectional transformers for language understanding / J. Devlin et al. *Proceedings of the 2019 conference of the North*, Minneapolis, Minnesota. Stroudsburg, PA, USA, 2019. URL: https://doi.org/10.18653/v1/n19-1423 (date of access: 15.05.2026).

13. Reimers N., Gurevych I. Sentence-BERT: sentence embeddings using siamese BERT-networks. *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, Hong Kong, China. Stroudsburg, PA, USA, 2019. URL: https://doi.org/10.18653/v1/d19-1410 (date of access: 15.05.2026).

14. BERT4Rec: sequential recommendation with bidirectional encoder representations from transformer / F. Sun et al. *CIKM '19: the 28th ACM international conference on information and knowledge management*, Beijing China. New York, NY, USA, 2019. URL: https://doi.org/10.1145/3357384.3357895 (date of access: 15.05.2026).

15. Cross-domain recommendation: challenges, progress, and prospects / F. Zhu et al. *Thirtieth international joint conference on artificial intelligence {IJCAI-21}*, Montreal, Canada, 19–27 August 2021. California, 2021. URL: https://doi.org/10.24963/ijcai.2021/639 (date of access: 15.05.2026).

16. Digital content market growth analysis - size and forecast 2026-2030. *Market Research Reports - Industry Analysis Size & Trends - Technavio - Technavio*. URL: https://www.technavio.com/report/digital-content-market-industry-analysis (date of access: 15.05.2026).

17. Burke R. *User modeling and user-adapted interaction*. 2002. Vol. 12, no. 4. P. 331–370. URL: https://doi.org/10.1023/a:1021240730564 (date of access: 02.06.2026).

18. Is ChatGPT a good recommender? A preliminary study / J. Liu et al. 2023. URL: https://doi.org/10.48550/arXiv.2304.10149 (date of access: 02.06.2026).

19. Efficient estimation of word representations in vector space / T. Mikolov et al. *Proceedings of the International conference on learning representations (ICLR 2013)*. 2013. URL: https://doi.org/10.48550/arXiv.1301.3781 (date of access: 02.06.2026).

20. Attention is all you need / A. Vaswani et al. *Advances in neural information processing systems 30*. 2017. URL: https://doi.org/10.48550/arXiv.1706.03762 (date of access: 02.06.2026).

21. RoBERTa: A robustly optimized BERT pretraining approach / Y. Liu et al. 2019. URL: https://doi.org/10.48550/arXiv.1907.11692 (date of access: 02.06.2026).

22. Improving language understanding by generative pre-training / A. Radford et al. 2018. URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf (date of access: 02.06.2026).

23. Unsupervised cross-lingual representation learning at scale / A. Conneau et al. *Proceedings of the 58th annual meeting of the association for computational linguistics*, Online. Stroudsburg, PA, USA, 2020.

URL: https://doi.org/10.18653/v1/2020.acl-main.747 (date of access: 02.06.2026).

24. Gunawardana A., Shani G. A survey of accuracy evaluation metrics of recommendation tasks. *Journal of machine learning research*. 2009. URL: https://doi.org/10.1145/1577069.1755883 (date of access: 02.06.2026).

25. Shani G., Gunawardana A. Evaluating recommendation systems. *Recommender systems handbook*. Boston, MA, 2010. P. 257–297. URL: https://doi.org/10.1007/978-0-387-85820-3_8 (date of access: 02.06.2026).

26. Kaminskas M., Bridge D. Diversity, serendipity, novelty, and coverage. *ACM Transactions on interactive intelligent systems*. 2017. Vol. 7, no. 1. P. 1–42. URL: https://doi.org/10.1145/2926720 (date of access: 02.06.2026).