**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет імені Петра Могили**

**Факультет комп'ютерних наук**

**Кафедра інтелектуальних інформаційних систем**

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____»_____ 20___ р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

# ІНТЕЛЕКТУАЛЬНА СИСТЕМА РЕКОМЕНДАЦІЙ МУЛЬТИМЕДІЙНОГО КОНТЕНТУ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

***Здобувач*** _____ Руслан ХОРОШЕВ

«___»_____ 20__ р.

***Керівник*** канд. техн. наук, доцент _____ Євген СІДЕНКО

«___»_____ 20__ р.

**Миколаїв – 2026**

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

| | |
|---|---|
| Факультет | Комп’ютерних наук |
| Кафедра | Інтелектуальних інформаційних систем |
| Рівень вищої освіти | Перший (бакалаврський) |
| Освітній ступень | Бакалавр |
| Спеціальність | 122 Комп’ютерні науки |
| Освітня програма | Комп’ютерні науки |

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____» _____ 20___ р.

**ЗАВДАННЯ**
**на кваліфікаційну роботу здобувача**

**Хорошев Руслан Андрійович**
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Інтелектуальна система рекомендацій мультимедійного контенту».

Керівник роботи: Сіденко Євген Вікторович, завідувач кафедри інтелектуальних інформаційних систем, канд. техн. наук, доцент.

Затверджена наказом ЧНУ ім. Петра Могили від «25» грудня 2025 р. № 353.

2. Строк представлення кваліфікаційної роботи «___» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: вебсистема персоналізованих рекомендацій мультимедійного контенту (музичних треків, фільмів, відеоігор) з підтримкою однодоменної та крос-доменної фільтрації; каталог мультимедійного контенту з трьох доменів (Last.fm, TMDB, RAWG); семантичні вкладення елементів каталогу, отримані за допомогою LLM та моделі sentence-transformers.

4. Перелік питань, що підлягають розробці: аналіз предметної сфери інтелектуальних рекомендаційних систем та існуючих платформ-аналогів; дослідження методів контентно-орієнтованої фільтрації, метрик подібності та підходів до крос-доменних рекомендацій; проєктування та реалізація вебсистеми з модулями однодоменної рекомендації на основі TF-IDF, крос-доменної рекомендації на основі семантичних вкладень та соціального порівняння смаків користувачів; інтеграція великих мовних моделей для генерації vibe-описів, пояснень рекомендацій та AI-режиму пошуку; оцінювання якості рекомендацій із застосуванням метрик Precision@K, Recall@K, MRR@K та порівняльного аналізу з базовими алгоритмами.

5. Перелік графічних матеріалів: презентація.

**Керівник роботи** _____ Євген СІДЕНКО
*(Особистий підпис)* *(Власне ім'я ПРІЗВИЩЕ)*

**Здобувач** _____ Руслан ХОРОШЕВ
*(Особистий підпис)* *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «21» грудня 2025 р.

**КАЛЕНДАРНИЙ ПЛАН**
**кваліфікаційної роботи**

Тема: **«Інтелектуальна система рекомендацій мультимедійного контенту»**

| № | Найменування роботи | Початок | Закінчення | Примітки |
|---|---|---|---|---|
| 1 | Отримання завдання на виконання КР | 21.12.2025 | 24.12.2025 | Виконано |
| 2 | Аналіз предметної області та постановка задачі | 25.12.2025 | 30.01.2026 | Виконано |
| 3 | Огляд літературних джерел за темою КР: методи фільтрації контенту, підходи до крос-доменних рекомендацій, семантичний аналіз на основі векторних представлень | 31.01.2026 | 01.03.2026 | Виконано |
| 4 | Дослідження та вибір методів побудови рекомендаційних систем, метрик подібності та засобів інтеграції великих мовних моделей | 02.03.2026 | 01.04.2026 | Виконано |
| 5 | Реалізація вебсистеми рекомендацій мультимедійного контенту з модулями однодоменної та крос-доменної фільтрації та функціями штучного інтелекту | 02.04.2026 | 24.05.2026 | Виконано |
| 6 | Перший попередній захист КР на засіданні комісії кафедри | 25.05.2026 | 25.05.2026 | Виконано |
| 7 | Корегування роботи за результатами попереднього захисту | 26.05.2026 | 04.06.2026 | Виконано |
| 8 | Другий попередній захист КР на засіданні комісії кафедри | 05.06.2026 | 05.06.2026 | Виконано |
| 9 | Доробка та остаточне оформлення КР | 06.06.2026 | 14.06.2026 | Виконано |
| 10 | Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту | 15.06.2026 | 19.06.2026 | Виконано |

**Керівник роботи** _____ Євген СІДЕНКО
*(Особистий підпис)* *(Власне ім'я ПРІЗВИЩЕ)*

**Здобувач** _____ Руслан ХОРОШЕВ
*(Особистий підпис)* *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«29» січня 2026 р.

# АНОТАЦІЯ

до кваліфікаційної роботи

здобувача групи 402 ЧНУ ім. Петра Могили

**Хорошев Руслан Андрійович**

на тему **“ІНТЕЛЕКТУАЛЬНА СИСТЕМА РЕКОМЕНДАЦІЙ МУЛЬТИМЕДІЙНОГО КОНТЕНТУ”**


Об’єктом роботи є процеси персоналізованого підбору мультимедійного контенту на основі аналізу вподобань користувачів.

Предметом роботи є методи та алгоритми побудови інтелектуальних рекомендаційних систем у мультимедійному середовищі.

Метою роботи є розробка інтелектуальної вебсистеми персоналізованого добору фільмів, музичних треків і відеоігор з механізмом порівняння смаків користувачів.

Методи роботи: контентно-орієнтована фільтрація, семантичний аналіз на основі векторних представлень, інтеграція великих мовних моделей.

У першому розділі проаналізовано предметну сферу рекомендаційних систем та існуючі платформи-аналоги, сформульовано постановку задачі. У другому розділі досліджено методи фільтрації, метрики подібності та засоби обробки мультимедійних даних. У третьому розділі представлено проєктування та реалізацію вебсистеми з модулями рекомендацій і профілювання користувачів та проведено аналіз якості алгоритмів. У четвертому розділі наведено керівництво користувача та проведено функціональне, інтеграційне і тестування продуктивності системи.

Розроблена система інтегрує фільми, музику та відеоігри в єдину екосистему персоналізованих рекомендацій із соціальним порівнянням смаків та генерацією пояснень на основі штучного інтелекту. Систему реалізовано у вигляді повноцінного вебзастосунку, готового до розгортання як самостійна платформа персоналізованих рекомендацій мультимедійного контенту. Результати роботи

можуть бути використані для впровадження у відкритих сервісах та як основа для подальших досліджень крос-доменних рекомендаційних систем. Загальний обсяг роботи – 96 сторінок. Кваліфікаційна робота містить 19 таблиць, 24 рисунки, 37 посилань та 6 додатків.

Ключові слова: *рекомендаційна система, контентно-орієнтована фільтрація, крос-доменна рекомендація, семантичне вкладення, великі мовні моделі, профіль користувача, персоналізація, мультимедійний контент, векторний пошук, схожість інтересів.*

# ABSTRACT

to the qualification work by the student of the group 402 of Petro Mohyla Black Sea National University

**Khoroshev Ruslan**

## «INTELLIGENT MULTIMEDIA CONTENT RECOMMENDATION SYSTEM»

The object of the study is the processes of personalized selection of multimedia content based on user preference analysis.

The subject of the study is the methods and algorithms for building intelligent recommendation systems in a multi-category environment.

The aim of the work is to develop an intelligent web system for personalized selection of films, music tracks, and video games with a mechanism for comparing user tastes.

Research methods: content-based filtering, semantic analysis based on vector representations, integration of large language models.

The first section analyzes the subject area of recommendation systems and existing analogous platforms, and formulates the problem statement. The second section examines filtering methods, similarity metrics, and multimedia data processing tools. The third section presents the design and implementation of the web system with recommendation and user profiling modules, and conducts an analysis of algorithm quality. The fourth section provides a user guide and performs functional, integration, and performance testing of the system.

The developed system integrates films, music, and video games into a unified ecosystem of personalized recommendations with social taste comparison and AI-based explanation generation. The system is implemented as a fully functional web application ready for deployment as a standalone personalized multimedia content recommendation platform. The results of the work can be used for implementation in open services and as a basis for further research into cross-domain recommendation systems.

The overall scope of the work is 96 pages. The qualification work contains 19 tables, 24 figures, 37 references and 6 appendices.

Keywords*: recommendation system, content-based filtering, cross-domain recommendation, semantic embedding, large language models, user profile, personalization, multimedia content, vector search, interest similarity.*

# ЗМІСТ

# СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

СУБД – Система управління базами даних

ШІ – Штучний інтелект

AI – Artificial Intelligence

API – Application Programming Interface

CBF – Content-Based Filtering

CF – Collaborative Filtering

CORS – Cross-Origin Resource Sharing

HTTP – Hyper Text Transfer Protocol

JSON – JavaScript Object Notation

IDF – Inverse Document Frequency

LLM – Large Language Model

SQL – Structured Query Language

TF – Term Frequency

# ВСТУП

Стрімке зростання обсягів цифрового контенту у сферах фільмів, музики та відеоігор породжує критичну проблему інформаційного перевантаження користувачів. Сучасні платформи пропонують десятки мільйонів одиниць контенту, і без ефективних механізмів персоналізації користувач витрачає значну частину часу на самостійний пошук релевантного матеріалу замість його безпосереднього споживання. Рекомендаційні системи стали ключовим інструментом вирішення цієї проблеми та активно досліджуються як в академічному середовищі, так і в індустрії.

При цьому сучасне споживання медіа перестало бути ізольованим за типами, користувач у межах одного дня може дивитися фільм, слухати музику та грати у відеогру, іноді роблячи дві дії одночасно. Ці види дозвілля переплітаються між собою на рівні настрою, тематики та естетики. Однак існуючі рекомендаційні системи цього зв’язку не відображають.

Провідні платформи – Netflix, Spotify, Steam – реалізують високоефективні рекомендаційні системи у межах своїх доменів. Проте кожна з них будує ізольований профіль користувача виключно у межах одного типу медіа, переглянуті фільми не впливають на музичні рекомендації, а ігрові вподобання ніяк не пов’язані з рештою. Користувач, який є фанатом певного фентезійного фільму, не отримає підказки, що варто спробувати тематично близьку гру чи музичний альбом подібного настрою. Найближчий загальнодоступний мультимедійний агрегатор Tastedive охоплює декілька типів медіа, але реалізує лише одноразове зіставлення елементів без накопиченого профілю та без соціальної взаємодії між користувачами.

Наукові дослідження крос-доменних рекомендаційних систем підтверджують перспективність підходів на основі семантичного аналізу та великих мовних моделей для знаходження зв’язків між різними типами медіа. Проте на практиці подібні підходи здебільшого обмежені двома доменами і не

охоплюють одночасно фільми, музику та ігри у вигляді повноцінної системи з накопиченим профілем користувача.

Актуальність даної роботи зумовлена відсутністю загальнодоступних платформ, що об’єднують усі три типи медіа в єдину екосистему на основі накопиченого профілю користувача з підтримкою крос-доменних рекомендацій та соціального порівняння смаків.

Декларація про використання ШІ. Під час підготовки наукової роботи (академічного тексту) було використано інструмент генеративного ШІ Claude Sonnet 4.6. Відповідно до таксономії GAIDeT (2025), наведені нижче завдання були делеговані інструментам генеративного ШІ за повного людського нагляду:

- оцінювання здійсненності та ризиків;
- генерування коду;
- оптимізація коду;
- вичитування та редагування;
- резюмування тексту;
- адаптація та коригування емоційного тону;
- оцінювання якості;
- рекомендації.

Повну відповідальність за фінальний рукопис несе автор.

Інструменти генеративного ШІ не зазначаються як автор та не несуть відповідальності за кінцеві результати.

Декларацію подав: Хорошев Руслан Андрійович.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ РЕКОМЕНДАЦІЙ МУЛЬТИМЕДІЙНОГО КОНТЕНТУ. ПОСТАНОВКА ЗАДАЧІ

Для визначення напряму розробки та обґрунтування технічних рішень необхідно провести аналіз предметної сфери інтелектуальних рекомендаційних систем. У даному розділі розглянуто особливості мультимедійного контенту як об’єкта рекомендацій, проаналізовано існуючі платформи-аналоги та наукові підходи до побудови рекомендаційних систем, а також сформульовано мету, об'єкт, предмет і завдання роботи.

## 1.1 Мультимедійний контент як предметна сфера інтелектуальних рекомендаційних систем

У сучасну цифрову епоху обсяг доступної мультимедійної інформації зростає темпами, що значно випереджають здатність людського сприйняття до її обробки. Предметна сфера інтелектуальних систем рекомендацій мультимедійного контенту охоплює складний комплекс технологій, економічних моделей та психологічних аспектів споживання, що фокусуються на наданні користувачеві найбільш релевантних цифрових об’єктів – фільмів, музичних треків та відеоігор. Глобальний ринок медіа та розваг демонструє стабільну експансію [1]. За оцінками SNS Insider, обсяг ринку у 2025 році досяг 3235,49 мільярдів доларів США, а до 2035 року прогнозується зростання до 6165,06 мільярдів доларів із середньорічним темпом росту (CAGR - Compound Annual Growth Rate) на рівні 6,67% [2].

Фундаментальним поняттям досліджуваної сфери є мультимедійний контент, це інтегрована сукупність цифрових даних різних форматів (аудіо, відео, текст, графіка), що поєднуються для забезпечення розважального досвіду користувача. У контексті даної роботи особлива увага приділяється трьом сегментам (табл. 1.1), що характеризуються явищем гіперпродукції. Станом на 2026 рік база даних IMDb містить майже 26 мільйонів назв [3], з яких понад 3,5 мільйони було додано лише

протягом 2025 року. Музична платформа Spotify щодня приймає близько 120 тисяч нових композицій, а загальна кількість треків у її бібліотеці перевищила 100 мільйонів [4]. Ринок відеоігор на платформі Steam у 2025 році продемонстрував рекордний приріст – було випущено понад 21 тисячу нових ігор [5].

Таблиця 1.1 – Динаміка обсягів мультимедійного контенту (2024–2026 рр.)

| **Показник** | **2024 рік** | **2025 рік** | **Прогноз на 2026 рік** |
|---|---|---|---|
| Загальна кількість одиниць контенту на IMDb | ~22 млн [3] | ~26 млн [3] | ~30 млн (очікувано) [3] |
| Щоденні завантаження треків на Spotify | ~60 тис. [4] | ~100 тис. [6] | >125 тис. [7] |
| Кількість релізів відеоігор на Steam за рік | 18,556 [5] | 21,517 [5] | >23,000 [5] |
| Дохід ринку медіа та розваг (млрд доларів США) | ~2,750 [8] | 2,870.56 [8] | 3,080.52 [8] |

Така надмірність інформації породжує критичну проблему "паралічу вибору" (choice paralysis). Користувачі витрачають значну частину часу на навігацію каталогами замість безпосереднього споживання контенту. Дослідження Deloitte вказують на те, що середній споживач у 2025–2026 роках витрачає близько 6 годин на день на медіа-активність, при цьому маючи в середньому чотири платні підписки на відеосервіси, щонайменше дві з них на медіаплатформах різного типу, а понад 60% споживають ігри, фільми та музику в рамках одного дня. Високий рівень відтоку клієнтів (churn rate), що сягає 41% протягом півроку, свідчить про те, що існуючі механізми фільтрації не завжди здатні утримати увагу аудиторії релевантними пропозиціями [9].

Інтелектуальні системи рекомендацій виступають головним інструментом вирішення цієї проблеми. Вони автоматично аналізують історичні дані, явні оцінки та неявну поведінку користувачів для побудови персоналізованих ранжованих

списків контенту [10]. Ефективність реалізації цих завдань залежить від того, наскільки точно алгоритм здатний інтерпретувати контекст споживання та передбачити майбутні потреби суб’єкта. Впровадження інтелектуальних методів обробки даних дозволяє перетворити пасивне сховище мультимедіа на динамічне середовище, що адаптується до кожного кроку користувача.

Історія рекомендаційних систем пройшла довгий шлях від найпростіших анкетних методів до сучасних нейромережевих архітектур [11]. Першою відомою системою вважається Grundy, розроблена Елейн Річ у 1979 році [12]. Ця система виконувала роль цифрового бібліотекаря, який на основі відповідей користувача на ряд запитань класифікував його за певними "стереотипами" (наприклад, "любитель наукової фантастики") і пропонував книги, що відповідають цьому профілю. Хоча Grundy не використовувала динамічне машинне навчання, вона заклала фундаментальну ідею моделювання профілю користувача для персоналізації досвіду [13].

З функціональної точки зору рекомендаційна система виконує головні три задачі:

– персоналізація, формування унікального досвіду для кожного користувача на основі його індивідуального профілю вподобань;

– утримання аудиторії, підтримання зацікавленості користувача шляхом безперервного постачання релевантного контенту;

– відкриття нового контенту, розширення горизонтів споживання шляхом пропозиції об’єктів, які користувач самостійно не знайшов би серед великого обсягу доступного матеріалу.

З архітектурної точки зору рекомендаційна система отримує на вході профіль користувача, сформований на основі явних оцінок та неявної поведінки, застосовує алгоритми фільтрації і повертає ранжований список рекомендацій (рис. 1.1).

У 1990-х роках фокус змістився на колаборативну фільтрацію (collaborative filtering). Система "Tapestry" (1992 р.) вперше впровадила механізм фільтрації документів на основі анотацій спільноти користувачів. Подальший розвиток у

проєкті "GroupLens" дозволив автоматизувати процес, використовуючи алгоритми для прогнозування оцінок новинних статей Usenet на основі схожості поведінки людей. Цей підхід став комерційно успішним завдяки Amazon, яка запатентувала метод "item-to-item collaborative filtering", що дозволив масштабувати рекомендації на мільйони товарів. Визначальною подією стало змагання Netflix Prize, у якому команді BellKor вдалося покращити точність рекомендаційного алгоритму на 10,06% за рахунок ансамблевих методів матричної факторизації, що стимулювало академічний та індустріальний інтерес до цієї сфери [11].

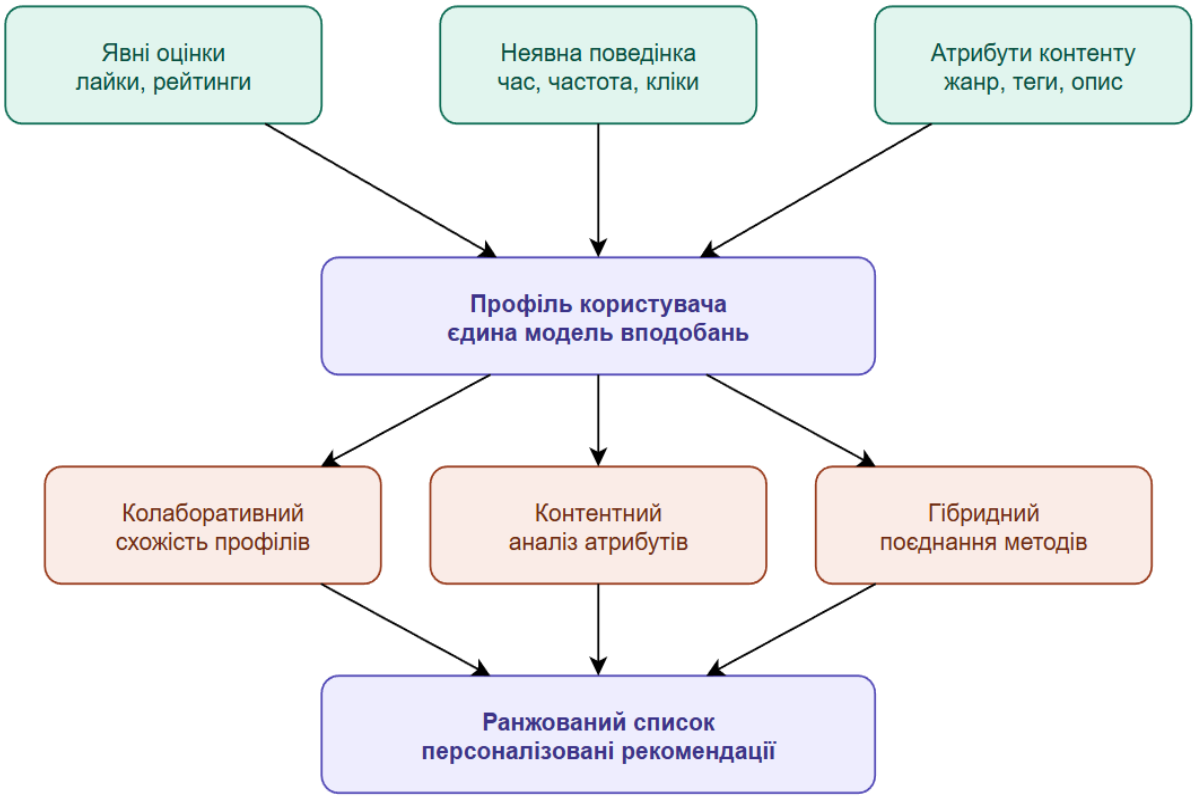


Рисунок 1.1 – Узагальнена архітектура рекомендаційних систем [10, 14]

Сучасний етап характеризується домінуванням глибокого навчання (Deep Learning) та інтеграцією великих мовних моделей [1]. Інтелектуальні системи стають не просто фільтрами, а адаптивними агентами, здатними розуміти семантичний контекст та емоційний стан користувача [15].

Аналіз ринку вказує на зміну парадигми споживання: користувачі все частіше відмовляються від традиційного лінійного телебачення на користь

OTT-платформ (Over-the-top) та відеосервісів. У 2025 році дохід від рекламної моделі в медіа склав 47% ринку, проте підписочна модель залишається найбільш динамічним сегментом [2]. Північна Америка утримує лідерство з часткою 38%, тоді як Азійсько-Тихоокеанський регіон демонструє найвищі темпи зростання (CAGR 9,96%) завдяки швидкому проникненню смартфонів та мобільного інтернету [2].

Особливе значення набуває концепція мультимедійності. У 2025–2026 роках спостерігається "сплетіння" інтересів: користувач, який є фанатом певного серіалу, з високою ймовірністю буде шукати музику до нього на Spotify та грати у тематичні відеоігри на Steam [9]. Ця залежність є значущою для розробки інтегрованих інтелектуальних систем, які здатні будувати єдиний профіль інтересів суб’єкта.

## 1.2 Аналіз існуючих платформ-аналогів та методів побудови рекомендаційних систем

Для формування науково обґрунтованої концепції інтелектуальної системи рекомендацій необхідно провести критичний аналіз існуючих комерційних платформ та актуальних наукових досліджень. Сучасні аналоги можна розділити на однодоменні (спеціалізовані на одному типі контенту) та мультидоменні агрегатори.

Netflix залишається еталоном у сфері відеорекомендацій. Система використовує понад 1300 сигналів, включаючи метадані об’єктів, поведінку під час перегляду (паузи, повтори) та контекстні дані (тип пристрою, час доби). Гібридна архітектура поєднує колаборативну фільтрацію з глибокими нейронними мережами для динамічної адаптації навіть мініатюр фільмів відповідно до вподобань конкретного користувача [16, 17]. Проте Netflix (рис. 1.2) є замкненою системою, яка не враховує вподобання користувача за межами відеоконтенту.

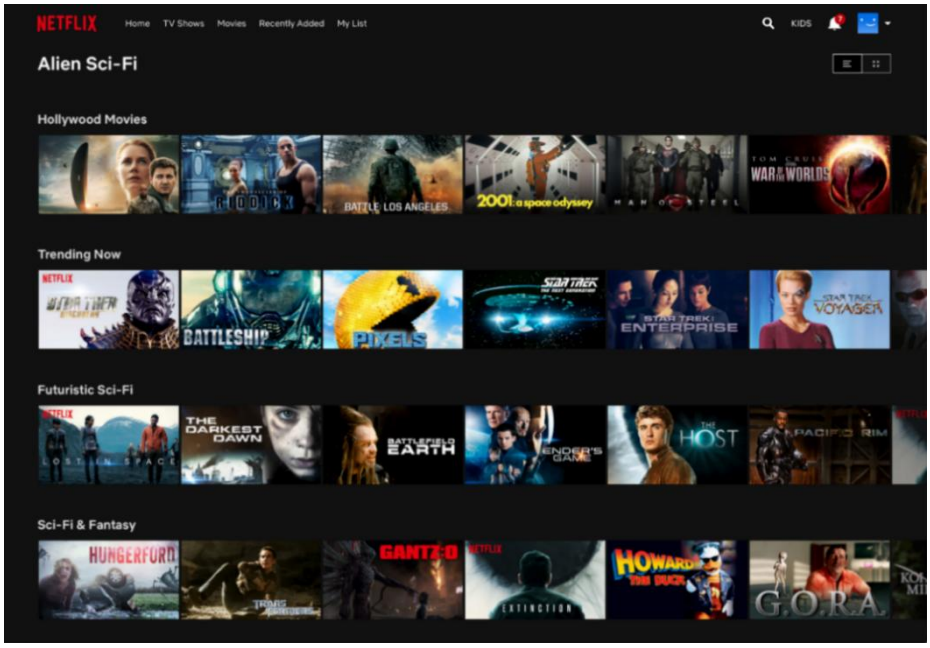


Рисунок 1.2 – Платформа фільмів та серіалів Netflix

Spotify (рис. 1.3) реалізує одну з найбільш інтелектуальних систем у музичній індустрії. Алгоритм "Discover Weekly" [18] використовує три незалежні моделі: колаборативну фільтрацію, аналіз текстів та описів через обробку природної мови, а також безпосередній звуковий аналіз аудіосигналу (Audio Analysis). Це дозволяє системі рекомендувати музику навіть нових виконавців без історії оцінок. Незважаючи на високу точність, платформа обмежена аудіофайлами і не надає можливості підбору користувачів зі схожими "стилями життя".

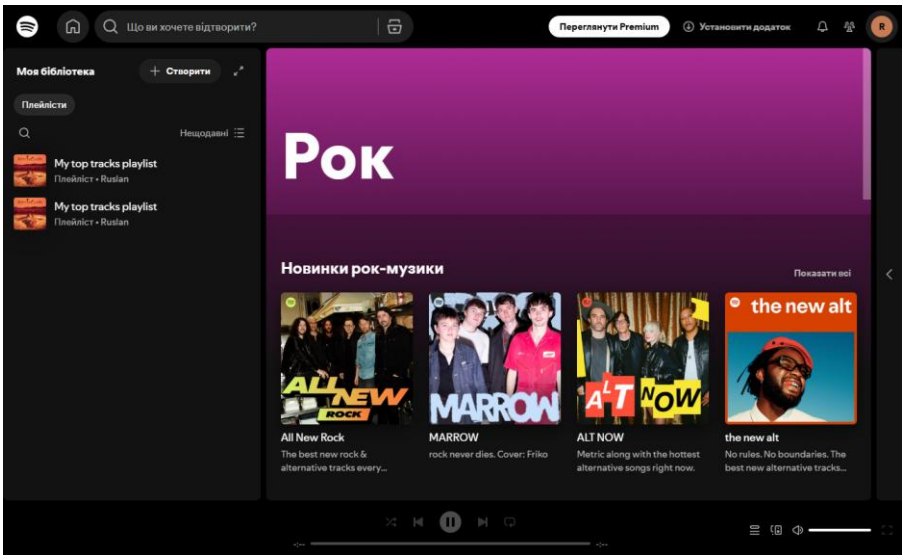


Рисунок 1.3 – Платформа музики Spotify

Steam фокусується на відеоіграх, використовуючи переважно контентно-орієнтований підхід (Content-Based Filtering) через систему тегів, жанрів та відгуків. Платформа надає детальну статистику часу у грі, що є важливим неявним сигналом інтересу. Має відому проблему "бульбашки фільтрів", коли система рекомендує лише відеоігри, схожі на вже зіграні, без врахування ширшого контексту інтересів користувача.

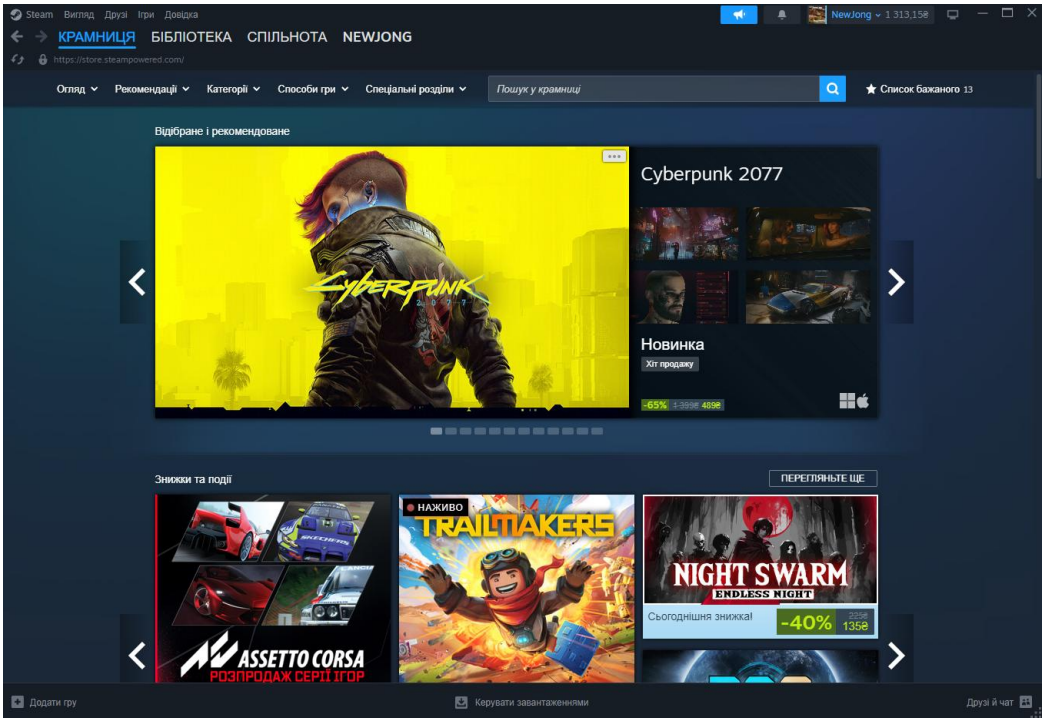


Рисунок 1.4 – Платформа відеоігор Steam

Tastedive – агрегатор рекомендацій, що охоплює фільми, музику, книги, відеоігри та телесеріали. Система будує рекомендації на основі контентно-орієнтованої фільтрації, аналізуючи семантичну схожість [19] між об’єктами різних категорій. Tastedive є найближчим аналогом до концепції даної роботи з точки зору мультимедійності. Проте сервіс не формує персоналізованого профілю користувача, крос-доменна рекомендація Tastedive є разовою, користувач вводить один об'єкт і отримує схожі з інших категорій, без накопиченої пам’яті про вподобання. Не реалізовано порівняння користувачів за схожістю вподобань.

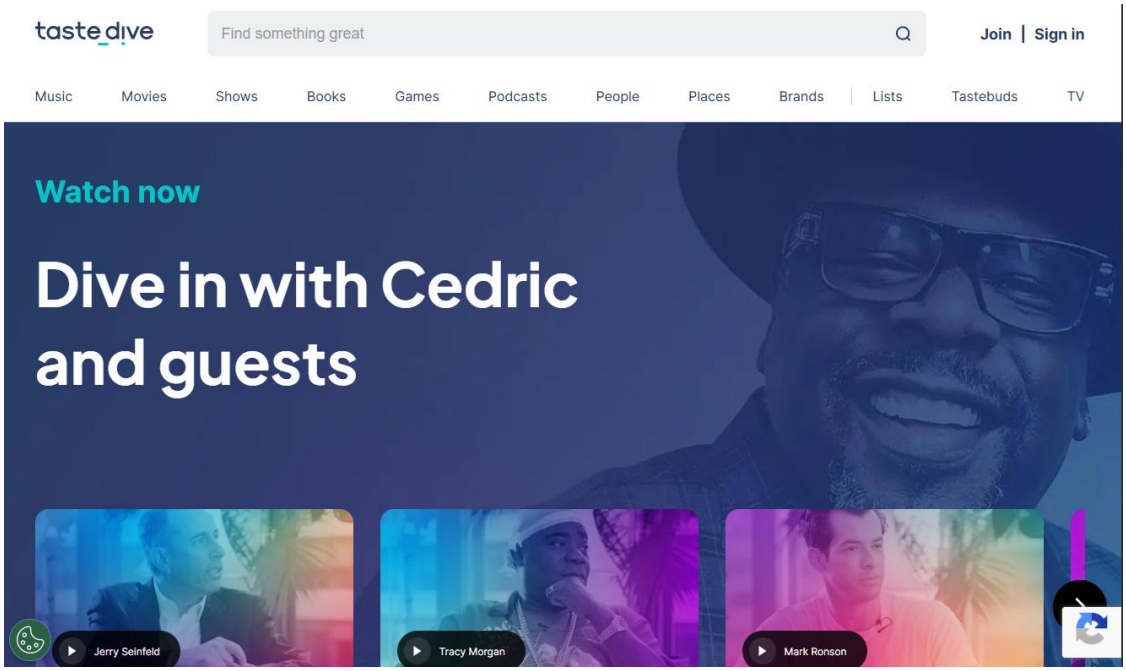


Рисунок 1.5 – Платформа ігор, фільмів, книг та музики Tastedive

Порівняльний аналіз розглянутих платформ наведено у табл. 1.2.

Таблиця 1.2 – Порівняння існуючих платформ

| **Платформа** | **Netflix** | **Spotify** | **Steam** | **Tastedive** | **Пропонована система** |
|---|---|---|---|---|---|
| **Сфера охоплення** | Фільми та серіали | Музика та подкасти | Відеоігри | Мультимедійна | Мультимедійна |
| **Тип фільтрації** | Гібридна (CF, CBF та ML) | Гібридна (CF, CBF та Audio) | Гібридна (CBF tags, CF, поведінкова) | CBF з обробкою природної мови | CBF. Семантична (LLM та векторний пошук) |
| **Крос-доменний аналіз** | Відсутній | Відсутній | Відсутній | Одноразовий, без пам'яті про користувача | Інтегрований на рівні профілю користувача |
| **Соціальний підбір** | Спільний перегляд, профіль сім'ї | Крос-профільний плейліст з оцінкою збігу у відсотках | Лише список друзів | Відсутній | Порівняння профілів з LLM-поясненнями |
| **Переваги** | Точність у відео | Аналіз аудіо-сигналу | Глибока метадата ігор | Універсальність | Єдиний мультимедійний профіль |
| **Недоліки** | Ізольованість домену | Обмеження домену | Обмеження у межах жанру | Відсутність профілю | Прототипний обсяг каталогу |

Проведений порівняльний аналіз дозволяє виокремити ряд системних обмежень, притаманних існуючим рекомендаційним платформам. Ці обмеження безпосередньо впливають на якість користувацького досвіду та визначають напрямки подальших досліджень і розробки рекомендаційних систем.

Проблема "холодного старту" (Cold Start Problem). Незважаючи на високу ефективність колаборативної фільтрації, що використовується у системах-аналогах критичним недоліком залишається неможливість формування точних рекомендацій для нових користувачів або об’єктів контенту без історії взаємодій [20]. Для подолання цього явища у дослідженні [21] пропонується застосування крос-доменного перенесення знань (CDR – Cross Domain Report), яке дозволяє використовувати накопичені дані з одного домену для рекомендацій в іншому. Підхід CD-LLMCARS [20] розвиває цю ідею через fine-tuning великих мовних моделей: згідно з результатами авторів, якість рекомендацій на етапі холодного старту покращується на 15–20% порівняно з baseline-методами матричної факторизації [22], проте обчислювальна вартість підходу суттєво обмежує його пряме застосування. Контентно-орієнтований підхід на основі аналізу тегів та жанрів є альтернативою, що не залежить від накопиченої матриці взаємодій і здатна формувати рекомендації вже після перших вподобань користувача.

Проблема розрідженості даних (Data Sparsity Problem). Згідно з дослідженням [23], навіть у зрілих системах матриця взаємодій заповнена менш ніж на 1–2%. У мультимедійному контексті проблема загострюється кратно: активність користувача у домені музики жодним чином не компенсує порожній профіль в ігровому домені, оскільки кожна платформа веде ізольований облік взаємодій. Порівняльний аналіз методів фільтрації [24] підтверджує, що IDF-зважування є більш стійким до розрідженості каталогу порівняно з класичним підрахунком входжень, оскільки дискримінаційна здатність алгоритму зберігається навіть при невеликій кількості оцінок. Використання LLM пропонує альтернативний підхід – замість пошуку статистичних кореляцій у розрідженій матриці вони оперують семантичними представленнями, встановлюючи змістовні зв’язки між елементами навіть за відсутності спільних взаємодій [20].

Проблема "фільтрової бульбашки" (Filter Bubble). Steam замикає рекомендації у вузькому жанровому колі за рахунок контентної фільтрації виключно на основі тегів. Spotify частково зменшує цей ефект через функцію

"Discover Weekly", проте обмеженість одним доменом не дозволяє використовувати крос-категорійні сигнали для розширення рекомендацій. Відсутність механізмів випадкового відкриття, тобто здатності системи пропонувати несподівані, але релевантні елементи – знижує довгострокову залученість користувачів [23]. Дослідження підтверджує, що мультимедійні підходи з механізмами крос-доменної уваги здатні суттєво розширити простір рекомендацій за межі вузького жанрового кола окремого домену [25].

Проблема ізольованості доменів. Жодна з розглянутих комерційних платформ не забезпечує інтегрованого крос-доменного досвіду. Netflix, Spotify та Steam функціонують як повністю ізольовані екосистеми, кожна з яких будує окремий профіль користувача виключно у межах свого домену. Tastedive є найближчим аналогом за принципом мультимедійності, проте обмежується поверхневим зіставленням об’єктів без побудови єдиного накопиченого профілю та не реалізує механізмів соціальної взаємодії між користувачами. Систематичний огляд [26] виділяє чотири основні таксономічні класи крос-доменних рекомендаційних систем: методи перенесення рейтингів, методи перенесення знань, методи перенесення тегів та методи спільного навчання (joint learning). Більшість платформ або не реалізують жодного з цих класів, або обмежуються парним перенесенням між двома суміжними доменами. Дослідження [21] підтверджує перспективність перенесення знань між доменами, проте наукові експерименти здебільшого обмежені 2–3 доменами, що підкреслює відкритість задачі для повноцінних тридоменних систем. Окремою перешкодою є неоднорідність атрибутного простору, теги музичних треків та жанри відеоігор описують об’єкти принципово різної природи і не мають прямої семантичної відповідності, що ускладнює побудову єдиного крос-доменного профілю. Модель Sentence-BERT [27] демонструє здатність проєктувати текстові описи різної доменної природи у єдиний семантичний простір, де близькість відображає змістовну спорідненість, а не поверхневий збіг ключових слів, що відкриває перспективний напрямок для вирішення цієї проблеми.

Для вирішення виявлених проблем проведено аналіз актуальних науково-практичних публікацій (табл. 1.3)

Таблиця 1.3 – Аналіз науково-практичних публікацій за темою роботи

| Джерело | Метод / Тип аналізу | Ключові особливості та параметри | Внесок у роботу |
|---|---|---|---|
| Medium [1], Dong Z. et al [11] | Еволюційний та історичний аналіз | Ретроспектива від системи Grundy (1979) до матричної факторизації та нейромереж | Формування теоретичного підґрунтя історії розвитку алгоритмів та класифікація базових архітектур |
| Ricci F. et al. [10], Aggarwal C. [14] | Фундаментальні дослідження (Handbook) | Систематизація методів колаборативної, контентної та гібридної фільтрації, математичний апарат метрик схожості | Визначення теоретичної бази та метрик якості для оцінки рекомендаційних систем |
| Hsiao K.-J et al [16], HBSTOM [18] | Практичні Case Studies (Netflix, Spotify) | Аналіз реальних архітектур (Discover Weekly, 1300+ сигналів Netflix) | Виявлення функціональних обмежень існуючих систем та формування вимог до власної розробки |
| Cheema A. et al [20], Zhao Y. et al [21] | Крос-доменне перенесення знань (CDR) | Використання LLM для зв’язку між доменами; покращення cold-start на 15–20% порівняно з матричною факторизацією | Обґрунтування крос-доменного підходу та аналіз обмежень існуючих методів перенесення знань |
| Herimanto et al [24] | Порівняльний аналіз CBF та TF-IDF | Стійкість TF-IDF до розрідженості каталогу порівняно з класичним підрахунком входжень | Обґрунтування вибору IDF-зважування як методу формування профілю користувача |
| Shahbazi Z et al [23], Dai J. [25] | Адаптивні механізми та мультимедійності | Реальний час, пояснюваність AI та дуальні графічні мережі крос-доменної уваги | Теоретичне обґрунтування мультимедійного підходу та механізмів серендипіті |
| Zang T. et al. ACM [26] | Систематичний огляд крос-доменних систем | Таксономія чотирьох класів CDR: перенесення рейтингів, знань, тегів та спільне навчання | Класифікація підходів до крос-доменних рекомендацій та виявлення відкритих задач для тридоменних систем |
| Reimers N. & Gurevych I. [27] | Семантичні векторні представлення (Sentence-BERT) | Siamese BERT-мережі для побудови sentence embeddings у єдиному семантичному просторі | Теоретичне обґрунтування можливості зіставлення текстових описів різної доменної природи |
| Wu L. et al. [28] | Огляд LLM для рекомендаційних систем | Аналіз ролі LLM у персоналізації, генерації пояснень та подоланні розрідженості даних | Обґрунтування інтеграції LLM для генерації природномовних пояснень та аналізу профілів |

Таким чином, проведений аналіз наукових публікацій та комерційних платформ 2024–2026 років дозволяє констатувати, що існуючі рішення не забезпечують цілісного мультимедійному досвіду. Виявлені системні обмеження обумовлюють доцільність розробки інтегрованої системи, яка об’єднує фільми, музику та відеоігри в єдину екосистему з підтримкою крос-доменних рекомендацій, соціального порівняння профілів користувачів та генерації природномовних пояснень на основі великих мовних моделей.

### 1.3 Постановка задачі

На основі проведеного аналізу предметної сфери та існуючих рішень виявлено суттєву прогалину – відсутність інтегрованих систем, що об’єднують фільми, музику та відеоігри в єдину екосистему з можливістю соціальної взаємодії на основі схожості смаків.

Актуальність кваліфікаційної роботи обумовлена стрімким зростанням обсягів мультимедійного контенту, що призводить до інформаційного перевантаження користувачів та зниження ефективності традиційних методів пошуку. Ізольованість існуючих платформ не дозволяє користувачу сформувати цілісне уявлення про свій "цифровий смак" та знаходити однодумців за межами одного типу медіа. Впровадження інтелектуальних методів на основі контентно-орієнтованої фільтрації, крос-доменного семантичного аналізу та великих мовних моделей дозволить значно покращити якість персоналізації, забезпечуючи релевантні рекомендації вже з першого вподобання, без накопиченої матриці взаємодій.

Метою роботи є розробка інтелектуальної вебсистеми рекомендацій мультимедійного контенту, яка забезпечує персоналізований добір фільмів, музичних треків і відеоігор на основі єдиного профілю користувача, а також реалізує механізм порівняння користувачів за схожістю їхніх уподобань.

Об’єктом роботи є процеси інтелектуального аналізу та персоналізованого підбору мультимедійного контенту на основі аналізу вподобань користувачів.

Предметом роботи є методи та алгоритми побудови рекомендаційних систем, зокрема методи контентно-орієнтованої фільтрації та семантичного аналізу на основі векторних представлень, метрики схожості у багатовимірних просторах ознак та архітектурні рішення для крос-доменної інтеграції даних.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– дослідити існуючі методи побудови рекомендаційних систем для однодоменної та крос-доменної рекомендації;

– сформувати підхід до крос-доменного зіставлення контенту різних типів медіа на основі жанрів, тегів та семантичних вкладень;

– спроєктувати архітектуру інтелектуальної системи, що включає підсистему збору даних через зовнішні API, блок профілювання користувачів та ядро рекомендацій;

– інтегрувати велику мовну модель для забезпечення пояснюваності рекомендацій та генерації природномовних описів порівняння користувацьких профілів;

– розрахувати схожість профілів користувачів для реалізації функції соціального порівняння;

– реалізувати програмний продукт у вигляді вебзастосунку із використанням сучасних технологій розробки;

– провести тестування системи та оцінити якість рекомендацій.

**Висновки до розділу 1**

У першому розділі проведено комплексний аналіз предметної області інтелектуальних рекомендаційних систем мультимедійного контенту. Встановлено, що стрімка експансія ринку у 2025–2026 роках зумовлює критичне зростання інформаційного навантаження на користувача. Критичний огляд комерційних аналогів (Netflix, Spotify, Steam, Tastedive) виявив їхню функціональну ізольованість у межах одного домену та п’ять ключових системних проблем: холодний старт, розрідженість даних, ефект фільтрової бульбашки,

ізольованість доменів та відсутність пояснюваності рекомендацій. Аналіз наукових публікацій підтвердив перспективність крос-доменного перенесення знань, семантичного аналізу на основі векторних представлень та інтеграції великих мовних моделей для генерації природномовних пояснень рекомендацій . Встановлено, що жодна з існуючих платформ не реалізує повноцінного мультимедійного профілю користувача з підтримкою крос-доменних рекомендацій та соціального порівняння смаків.

На підставі виявлених проблем сформульовано постановку задачі: розробка інтелектуальної вебсистеми, що об’єднує фільми, музику та відеоігри в єдину мультимедійну екосистему. Визначено мету роботи, об’єкт та предмет роботи, а також перелік завдань, що охоплюють реалізацію контентно-орієнтованої фільтрації, крос-доменного семантичного аналізу, механізму соціального порівняння профілів та інтеграції великої мовної моделі для генерації природномовних пояснень рекомендацій.

# 2 МЕТОДИ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ СИСТЕМИ РЕКОМЕНДАЦІЙ МУЛЬТИМЕДІЙНОГО КОНТЕНТУ

Проєктування інтелектуальної системи рекомендацій мультимедійного контенту вимагає обґрунтованого вибору як математичних методів фільтрації, так і технологічного стеку для їх реалізації. У цьому розділі описано методи, що безпосередньо застосовуються у пропонованій системі, та обґрунтовано вибір інформаційних технологій розробки.

## 2.1 Методи для вирішення поставленої задачі

Вибір методів для побудови рекомендаційної системи ґрунтується на поєднанні класичних підходів до фільтрації контенту з семантичним аналізом на основі векторних представлень та інтеграцією великої мовної моделі. Відповідно до постановки задачі (п. 1.3), система реалізує три ключові механізми: контентно-орієнтовану фільтрацію на основі TF-IDF для однодоменних рекомендацій, кросдоменну рекомендацію через LLM-нормалізацію та векторний пошук, а також соціальне порівняння профілів користувачів на основі косинусної схожості.

### 2.1.1 Контентно-орієнтована фільтрація на основі TF-IDF

Контентно-орієнтована фільтрація (CBF) базується на аналізі внутрішніх характеристик об’єктів і не залежить від оцінок інших користувачів, що робить її ефективною для вирішення проблеми холодного старту [10]. У пропонованій системі CBF реалізується через зважування тегів методом IDF та косинусну схожість між профілем користувача і векторами контенту.

Перевагою IDF-зважування над простим підрахунком тегів є те, що популярні теги, які зустрічаються у більшості елементів каталогу (наприклад, "drama" або "action"), несуть менше інформації про унікальність смаку користувача. Натомість рідкісні теги (наприклад, "neo-noir" або "psychological")

точніше відображають специфічні вподобання і отримують вищу вагу при формуванні профілю [29].

Кожен елемент каталогу представлений набором тегів. Для треків та ігор використовуються поля жанрів і тегів, для фільмів – жанри та ключові слова. Для зважування тегів використовується метод IDF, який підвищує вагу рідкісних тегів та знижує вагу надто поширених [24]:

$$\mathrm{I}DF(t) = \log\left(\frac{N}{df(t)+1}\right), \tag{2.1}$$

де $N$ – кількість елементів у домені;

$df(t)$ – кількість елементів, що містять тег t.

Профіль користувача формується як зважений вектор тегів його вподобаних елементів. Для кожного тегу, що зустрічається у вподобаному контенті:

$$u(t) = \mathrm{c}(t) \times IDF(t), \tag{2.2}$$

де u(t) – вага тегу *t* у профілі користувача;

c(t) – кількість вподобаних елементів, що містять тег *t*;

IDF – обернена частота тегу *t* (2.1).

Кожен елемент каталогу представляється бінарним вектором: значення 1, якщо елемент містить тег, 0 – якщо ні. Схожість між профілем користувача та елементом визначається через косинусну міру подібності [10, 14]:

$$\cos(u, v_i) = (u \times v_i) \div (\|u\| \times \|v_i\|), \tag{2.3}$$

де $u$ – вектор профілю користувача;

$v_i$ – бінарний вектор ознак елемента.

Результатом є список елементів каталогу, відсортованих за спаданням косинусної схожості, за виключенням уже вподобаних – система повертає задану кількість найрелевантніших рекомендацій. Для фільмів застосовується мінімальний поріг частоти тегу, якщо тег зустрічається щонайменше у двох

елементах каталогу то тег враховується, для треків та ігор цей поріг становить один елемент. Перевагою такого підходу є незалежність від історії інших користувачів, система здатна формувати рекомендації вже після перших вподобань без необхідності накопиченої бази взаємодій.

### 2.1.2 Крос-доменна рекомендація через LLM-нормалізацію та векторний пошук

Основною проблемою прямого застосування векторних представлень у крос-доменному контексті є різниця лексики між доменами: опис фільму оперує поняттями сюжету та персонажів, гра описується через механіки та жанри, музика – через настрій та темп. Навіть тематично близькі елементи різних доменів матимуть низьку косинусну схожість у такому просторі через відсутність спільного словника [21]. Для вирішення цієї проблеми у системі реалізовано двоступеневий процес нормалізації (vibe-pipeline).

Генерація нормалізованого текстового опису. Для кожного елемента каталогу (трек, фільм або відеогра) велика мовна модель Claude Haiku генерує нормалізований текстовий опис – параграф обсягом 60–90 слів у спільному описовому словнику: настрій, енергетика, атмосфера, контекст використання та тематичні абстракції. Опис формується без згадок назви, сюжету або ігрових механік, що забезпечує представлення незалежне від специфіки домену. Згенерований текст зберігається у відповідній таблиці бази даних.

Побудова векторного представлення. Нормалізований текстовий опис перетворюється на 384-вимірний числовий вектор за допомогою моделі sentence-transformers/all-MiniLM-L6-v2 [27]. Вектор зберігається у базі даних з використанням розширення pgvector для подальшого векторного пошуку. Семантична близькість між двома елементами визначається як:

$$sem(a, b) = 1 - cosine\_distance(emb(a), emb(b)), \quad (2.4)$$

де emb(a), emb(b) – векторні представлення елементів a та b.

Гібридне ранжування. Чистий семантичний сигнал доповнюється бонусом за перетин тегів між елементами різних доменів, що покращує результати для схожості франшиз (наприклад, фільм Spider-Man та гра Spider-Man), де семантична схожість нормалізованого текстового опису може бути недостатньою:

$$bonus(a,b) = \min\left(MAX\_BONUS, |tags(a) \cap tags(b)| \times TAG\_BONUS\right), \quad (2.5)$$

де tags(a) ∩ tags(b) – кількість спільних тегів між елементами a та b;

TAG_BONUS – коефіцієнт бонусу за один спільний тег;

MAX_BONUS – максимально допустиме значення бонусу.

$$hybrid(a,b) = \min(1, sem(a,b) + bonus(a,b)), \quad (2.6)$$

де sem(a, b) – семантична схожість між елементами (2.4);

bonus(a, b) – бонус за перетин тегів (2.5).

Для формування крос-доменних рекомендацій система спочатку відбирає кандидатів з іншого домену за найвищою семантичною схожістю, після чого переранжовує їх за гібридною формулою з урахуванням тегового бонусу. Це дозволяє швидко відфільтрувати нерелевантні елементи та уточнити порядок найкращих кандидатів.

Для коректного відображення відсотка схожості користувачу застосовується лінійне перетворення. Це необхідно оскільки значення косинусної схожості між векторними представленнями елементів концентрується у вузькому діапазоні 0.55–0.95, де навіть семантично непов’язані пари мають схожість близько 0.60. Без такого перетворення система відображала б завищені значення схожості навіть для випадкових пар елементів, що вводило б користувача в оману щодо реальної близькості контенту [23]:

$$displayed\_percent = clamp((hybrid - 0.5) \times 2, 0, 1) \times 100, \quad (2.7)$$

де hybrid – гібридна оцінка схожості пари елементів (2.6);

clamp – функція обмеження значення у діапазоні від 0 до 1.

### 2.1.3 Соціальне порівняння профілів користувачів

Окрім генерації рекомендацій, система реалізує механізм соціального порівняння – при перегляді профілю іншого користувача відображається ступінь схожості смаків, спільний контент та унікальні вподобання кожного з них.

Сумісність смаків між двома користувачами обчислюється як косинусна схожість між їхніми тегово-зваженими профільними векторами, побудованими за тим самим IDF-принципом що і в однодоменній фільтрації (п. 2.1.1). Для двох користувачів *u* та *v* схожість (2.3), де замість вектора елемента використовується профільний вектор іншого користувача. Результат перетворюється у відсоток схожості та відображається у профілі.

Окрім відсотка схожості система визначає три складові порівняння:

– спільні вподобання, які обидва користувачі додали до своєї бібліотеки, визначаються окремо для кожного з трьох доменів як перетин їхніх колекцій;

– елементи бібліотеки другого користувача, які відсутні у бібліотеці першого, ранжовані за різницею тегово-зважених профільних векторів, це найкраще відображає унікальну частину смаку другого користувача і може бути цікавим для першого;

– генерується великою мовною моделлю на основі обчислених даних і містить короткий природномовний аналіз спільного, відмінного та рекомендацію щодо унікального контенту (п. 2.1.4).

Такий підхід дозволяє не просто показати числову схожість, а надати користувачу змістовне уявлення про те, в чому саме збігаються або відрізняються їхні смаки [21].

### 2.1.4 Інтеграція великої мовної моделі (LLM)

Великі мовні моделі активно інтегруються у рекомендаційні системи для забезпечення пояснюваності та покращення взаємодії з користувачем [28]. У пропонованій системі LLM використовується не для обчислення рекомендацій, а

як компонент генерації природномовних пояснень на основі вже розрахованих даних.

Генерація нормалізованих текстових описів. Для кожного елемента каталогу модель генерує описовий параграф у спільному словнику настрою та атмосфери, що є основою крос-доменного векторного пошуку (п. 2.1.2). Ця функція виконується одноразово при наповненні каталогу і не залежить від дій користувача.

Пояснення крос-доменних пар. На запит користувача модель формує одне коротке речення, що пояснює чому конкретний фільм, трек або гра схожі між собою за настроєм чи тематикою. Результат кешується у базі даних, тому повторний запит тієї самої пари не потребує додаткового звернення до моделі.

AI-пошук за описом. Користувач може описати бажаний контент довільною фразою, після чого модель аналізує каталог і повертає до п’яти найрелевантніших елементів з поясненням вибору.

Технічно інтеграція реалізується через API-виклики до моделі Claude Haiku 4.5 (Anthropic) із серверної частини системи. Порівняння LLM-моделей, розглянутих для інтеграції, наведено у табл. 2.1.

Таблиця 2.1 – Порівняння великих мовних моделей

| **Модель** | **Контекст (токени)** | **Якість українською** | **API-доступ** | **Рішення** |
|---|---|---|---|---|
| Claude Haiku 4.5 (Anthropic) | 200 000 | Висока | REST API | Обрано |
| GPT-4o (OpenAI) | 128 000 | Висока | REST API | Альтернатива |
| Llama 3 (Meta) | 8 000–128 000 | Середня | Self-hosted | Потребує інфраструктури |
| Gemini Pro (Google) | 1 000 000 | Нестабільна | REST API | Нестабільна якість генерації |

Claude Haiku 4.5 обрано завдяки високій якості генерації тексту, підтримці великого контексту що дозволяє передавати повний профіль користувача в одному

запиті, та прозорій тарифікації за токени [28]. Відповіді моделі кешуються у базі даних для уникнення повторних звернень до API.

### 2.1.5 Метрики оцінки якості рекомендацій

Для об'єктивної оцінки якості роботи рекомендаційної системи необхідно визначити формалізовані метрики [10]. Оскільки система реалізує два принципово різних типи рекомендацій однодоменний та крос-доменний, то для кожного з них застосовується окремий набір метрик.

Precision@K – частка релевантних елементів серед рекомендованих [29]:

$$Precision@K = |relevant \cap recommended@K| \div K, \quad (2.8)$$

де relevant – релевантні елементи;

recommended@K – рекомендовані елементи;

K – кількість рекомендованих елементів.

Ця метрика відповідає на питання "яка частина рекомендацій виявилася корисною?" та є важливою для інтерфейсу, де кожна нерелевантна рекомендація знижує довіру користувача до системи.

Recall@K – частка знайдених релевантних об'єктів серед усіх релевантних [29]:

$$Recall@K = |relevant \cap recommended@K| \div |relevant|, \quad (2.9)$$

де relevant – релевантні елементи;

recommended@K – рекомендовані елементи;

K – кількість рекомендованих елементів.

MRR@K (Mean Reciprocal Rank) – середнє значення оберненого рангу першого релевантного елемента у списку рекомендацій:

$$\text{MRR@K} = mean(1 \div rank\_of\_first\_hit), \quad (2.10)$$

де rank_of_first_hit – позиція першого релевантного елемента у списку.

Якщо релевантних елементів немає – значення дорівнює 0.

MRR@K оцінює наскільки високо система розміщує перший корисний результат, чим вище він у списку, тим краще.

Метрики крос-доменної рекомендації. Крос-доменна задача не має розміченого еталону – неможливо однозначно визначити які фільми мають відповідати яким іграм. Тому замість класичних метрик точності використовується порівняння алгоритму з випадковим базовим рівнем на однакових наборах елементів. Застосовуються три метрики:

- семантична схожість, середнє значення косинусної схожості між векторними представленнями рекомендованих елементів та вихідного елемента;
- частка спільних тегів, середня кількість спільних тегів між рекомендованими елементами та вихідним;
- покриття тегів, частка рекомендованих елементів, що мають щонайменше один спільний тег з вихідним.

Для кожної метрики обчислюється показник підсилення (lift) як відношення значення алгоритму до значення випадкового базового рівня. Чим вище підсилення, тим більше алгоритм відрізняється від випадкового вибору [21].

## 2.2 Технології розробки системи

Вибір технологічного стеку для реалізації системи обумовлений вимогами до швидкодії обчислення рекомендацій, гнучкості інтеграції із зовнішніми API та забезпечення сучасного інтерфейсу користувача. У цьому підрозділі обґрунтовано вибір кожного компонента.

### 2.2.1 Серверна частина: Python та FastAPI

Для реалізації серверної частини обрано мову програмування Python з фреймворком FastAPI. Python є домінантною мовою у сфері машинного навчання та обробки даних, що забезпечує доступ до необхідних бібліотек: NumPy для

векторних обчислень, scikit-learn для реалізації метрик схожості та sentence-transformers для побудови векторних представлень [30].

FastAPI – сучасний асинхронний вебфреймворк з вбудованою валідацією даних та автоматичною генерацією документації API [31]. Ключовою перевагою для даної задачі є нативна підтримка асинхронних операцій, що дозволяє паралельно обробляти запити до зовнішніх API (TMDB, RAWG, Last.fm).

Порівняння серверних фреймворків наведено у табл. 2.2.

Таблиця 2.2 – Порівняння серверних фреймворків

| **Фреймворк** | **Модель I/O** | **Документація API** | **Екосистема** | **Рішення** |
|---|---|---|---|---|
| FastAPI (Python) | Async (ASGI) | Автоматична (OpenAPI) | Висока | Обрано |
| Django REST(Python) | Sync (WSGI) | Через DRF Browsable | Висока | Надмірна складність |
| Express.js (Node.js) | Async (Event Loop) | Ручна (Swagger) | Середня | Відсутність ML-бібліотек |
| Spring Boot (Java) | Sync/Reactive | Через SpringDoc | Дуже висока | Надлишковий для даної задачі |

Таким чином, FastAPI(Python) забезпечує оптимальне поєднання продуктивності, простоти розробки та доступу до екосистеми машинного навчання, що робить його найбільш відповідним вибором для реалізації серверної частини системи рекомендацій.

### 2.2.2 База даних: PostgreSQL

Для зберігання даних обрано реляційну СУБД PostgreSQL з відкритим вихідним кодом [32]. Є підтримка розширення pgvector, яке дозволяє зберігати векторні представлення елементів каталогу безпосередньо в базі даних та виконувати ефективний пошук за косинусною схожістю, що є основою крос-

доменного механізму рекомендацій (п. 2.1.2). Це дозволяє уникнути використання окремого векторного сховища та спростити архітектуру системи.

Порівняння з альтернативними СУБД наведено у табл. 2.3.

Таблиця 2.3 – Порівняння систем управління базами даних

| **СУБД** | **Ключові можливості** | **Масштабованість** | **Оптимальне застосування** | **Рішення** |
|---|---|---|---|---|
| PostgreSQL | ACID, JSONB, масиви; pgvector | Висока (MVCC) | Складний контент | Обрано |
| MySQL | ACID, JSON (обмежений) | Висока | Прості схеми | Відсутність pgvector |
| MongoDB | Документна, гнучка схема | Горизонтальна | Неструктуровані дані | Відсутність pgvector |
| SQLite | Вбудована, файлова | Один потік запису | Прототипи | Не масштабується |

PostgreSQL є єдиною з розглянутих СУБД що поєднує повноцінну реляційну модель з підтримкою векторного пошуку, що робить її оптимальним вибором для даної системи.

### 2.2.3 Клієнтська частина React та Astro

Для побудови клієнтського інтерфейсу обрано комбінацію бібліотеки React та фреймворку Astro. React забезпечує компонентний підхід до побудови інтерфейсу з ефективним оновленням лише тих елементів сторінки що змінились [33], що є важливим для динамічних сторінок каталогу та профілів користувачів.

Astro реалізує архітектуру островів, де більшість сторінок генерується як статичний HTML, а React-компоненти підключаються лише там де потрібна інтерактивність [34]. Це мінімізує обсяг JavaScript що завантажується користувачем, а швидкість завантаження безпосередньо впливає на залученість користувачів до платформи. Astro підтримує повну інтеграцію з React, що дозволяє

використовувати його для інтерактивних елементів системи: каталогів контенту, модулів рекомендацій та інтерфейсу соціального порівняння.

Порівняння клієнтських фреймворків наведено у табл. 2.4

Таблиця 2.4 – Порівняння клієнтських фреймворків

| **Фреймворк** | **Архітектура** | **Переваги** | **Оптимальне застосування** | **Рішення** |
|---|---|---|---|---|
| Next.js | SSR/SSG, React; API routes | Full-stack з одного фреймворку | SaaS, e-commerce | Надмірний для задачі |
| Vue.js + Nuxt | SSR/SSG, Composition API | Простота, менший розмір пакету | Середні проєкти | Менша екосистема |
| Angular | Full framework, TypeScript | Корпоративна стабільність | Enterprise | Overhead для прототипу |
| React + Astro | Islands; SSG + CSR, React-екосистема | Мінімальний JS; швидке завантаження | Медіаплатформи | Обрано |

Поєднання React та Astro забезпечує оптимальний баланс між інтерактивністю інтерфейсу та швидкістю завантаження, що є важливим для медіаплатформи з великою кількістю контенту.

### 2.2.4 Зовнішні API: TMDB, RAWG та LastFM

Для отримання метаданих мультимедійного контенту система інтегрується з трьома зовнішніми API.

TMDB (The Movie Database) API – відкритий API з базою даних фільмів, що містить понад 900 тисяч фільмів із детальними метаданими: жанри, ключові слова, оцінки та постери [35]. Метадані використовуються для побудови профілів фільмового контенту у контентно-орієнтованій фільтрації (п. 2.1.1) та крос-доменного семантичного аналізу.

RAWG API – база даних відеоігор з інформацією про понад 500 тисяч ігор з деталізованими тегами та жанрами [36]. Теги та жанри використовуються для

побудови профілів ігрового контенту у контентно-орієнтованій фільтрації (п. 2.1.1), а також як додатковий сигнал у гібридному ранжуванні крос-доменних рекомендацій (п. 2.1.2).

Last.fm API – музична платформа з базою даних понад 100 мільйонів треків. API надає жанрові мітки та теги створені спільнотою користувачів, що формує багату семантичну розмітку музичного контенту. Ці теги використовуються як атрибути для побудови профілів треків у контентно-орієнтованій фільтрації та крос-доменного аналізу.

Метадані з TMDB та RAWG завантажуються у базу даних одноразово, а Last.fm використовується як одноразово для наповнення, так і для отримання деталей треків у реальному часі за запитом.

### 2.2.5 Допоміжні бібліотеки та інструменти

Для реалізації алгоритмів системи використовуються наступні бібліотеки:

– NumPy бібліотека для векторних обчислень, що використовується для побудови профільних векторів користувачів та обчислення косинусної схожості [30];

– scikit-learn використовується для розрахунку метрик якості рекомендацій Precision@K, Recall@K та MRR@K [37];

– sentence-transformers для побудови векторних представлень тексту, використовується для перетворення нормалізованих текстових описів у 384-вимірні вектори (п. 2.1.2);

– anthropic SDK офіційна бібліотека для взаємодії з API Claude Haiku, використовується для всіх LLM-функцій системи (п. 2.1.4);

– psycopg2 драйвер для підключення до PostgreSQL з Python;

– Firebase Auth сервіс автентифікації користувачів, що забезпечує реєстрацію та вхід до системи.

Зведену інформацію про обраний технологічний стек наведено у табл. 2.5.

Таблиця 2.5 – Технологічний стек системи

| Компонент | Технологія | Призначення |
|---|---|---|
| Серверна частина | Python 3.10 + FastAPI | Асинхронний API, інтеграція з NumPy/scikit-learn для обчислення рекомендацій |
| База даних | PostgreSQL 17 + pgvector | Зберігання даних та векторний пошук для крос-доменних рекомендацій |
| Клієнтська частина | React 19 + Astro 6 | Islands-архітектура, мінімальний JS на клієнті |
| API фільмів | TMDB API v3 | Метадані фільмів: жанри, ключові слова, оцінки |
| API відеоігор | RAWG API | Метадані ігор: теги та жанри |
| API музики | LastFM API | Метадані треків: жанри та теги спільноти |
| LLM-компонент | Claude API (Anthropic) | Генерація текстових описів, пояснення рекомендацій, AI-пошук |
| Обчислення | NumPy, scikit-learn | Векторні операції, метрики якості рекомендацій |
| Векторні представлення | sentence-transformers/all-MiniLM-L6-v2 | Перетворення текстових описів у 384-вимірні вектори |

## Висновки до розділу 2

У другому розділі обґрунтовано вибір методів та інформаційних технологій для побудови інтелектуальної системи рекомендацій мультимедійного контенту. Описано п’ять методів: контентно-орієнтовану фільтрацію на основі IDF-зважування тегів, крос-доменну рекомендацію через LLM-нормалізацію та векторний пошук, соціальне порівняння профілів користувачів на основі косинусної схожості, інтеграцію великої мовної моделі Claude Haiku 4.5 для генерації природномовних пояснень та метрики оцінки якості – Precision@K, Recall@K та MRR@K для однодоменних рекомендацій і метрики семантичної схожості для крос-доменних.

Обґрунтовано вибір технологічного стеку: Python + FastAPI для серверної частини, PostgreSQL з розширенням pgvector для зберігання даних та векторного пошуку, React 19 та Astro 6 для клієнтського інтерфейсу, TMDB, RAWG та Last.fm API для отримання метаданих контенту, sentence-transformers для побудови векторних представлень. Кожен вибір підтверджено порівняльним аналізом альтернатив.

# 3 РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ РЕКОМЕНДАЦІЙ МУЛЬТИМЕДІЙНОГО КОНТЕНТУ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Розробка інтелектуальної системи рекомендацій мультимедійного контенту передбачає проєктування архітектури, визначення структури даних та реалізацію алгоритмів фільтрації, описаних у розділі 2. У цьому розділі представлено структуру вхідних даних та архітектуру системи, продемонстровано роботу основних модулів, а також проведено аналіз якості рекомендацій за визначеними метриками.

## 3.1 Опис вхідних даних та структури системи

Перед реалізацією алгоритмів рекомендацій необхідно визначити структуру вхідних даних, архітектуру системи та організацію бази даних. У цьому підрозділі описано загальну архітектуру системи, структуру таблиць бази даних та характеристики вхідного каталогу контенту.

### 3.1.1 Архітектура системи

Система побудована за трирівневою клієнт-серверною архітектурою, що включає клієнтську частину, серверну частину та рівень зберігання даних. Взаємодія між рівнями відбувається через REST API з передачею даних у форматі JSON по захищеному протоколу HTTPS.

Клієнтська частина реалізована на основі фреймворку Astro 6 з інтегрованими React 19 компонентами та стилізацією через Tailwind CSS v4. Автентифікація користувачів здійснюється через Firebase Auth з підтримкою Google OAuth, що дозволяє уникнути зберігання паролів у власній базі даних та забезпечує безпечний вхід до системи.

Серверна частина реалізована на FastAPI (Python) і містить окремі роутери для кожного функціонального модуля: content, users, likes, friends, search,

recommendations, cross та compare. Такий підхід забезпечує чітке розмежування відповідальності між модулями та спрощує подальше розширення системи. На рівні серверної частини також виконується побудова векторних представлень елементів каталогу за допомогою моделі sentence-transformers/all-MiniLM-L6-v2.

Рівень зберігання даних представлено PostgreSQL (Neon) з розширенням pgvector. З'єднання між серверною частиною та базою даних здійснюється через драйвер psycopg2. Зовнішні API – TMDB, RAWG та Last.fm використовуються на етапі наповнення каталогу і не залучаються під час роботи рекомендаційних алгоритмів. Усі генеративні функції системи виконуються через звернення до Claude Haiku 4.5 (Anthropic API) по захищеному протоколу HTTPS.

Загальну архітектуру системи наведено на рис. 3.1.

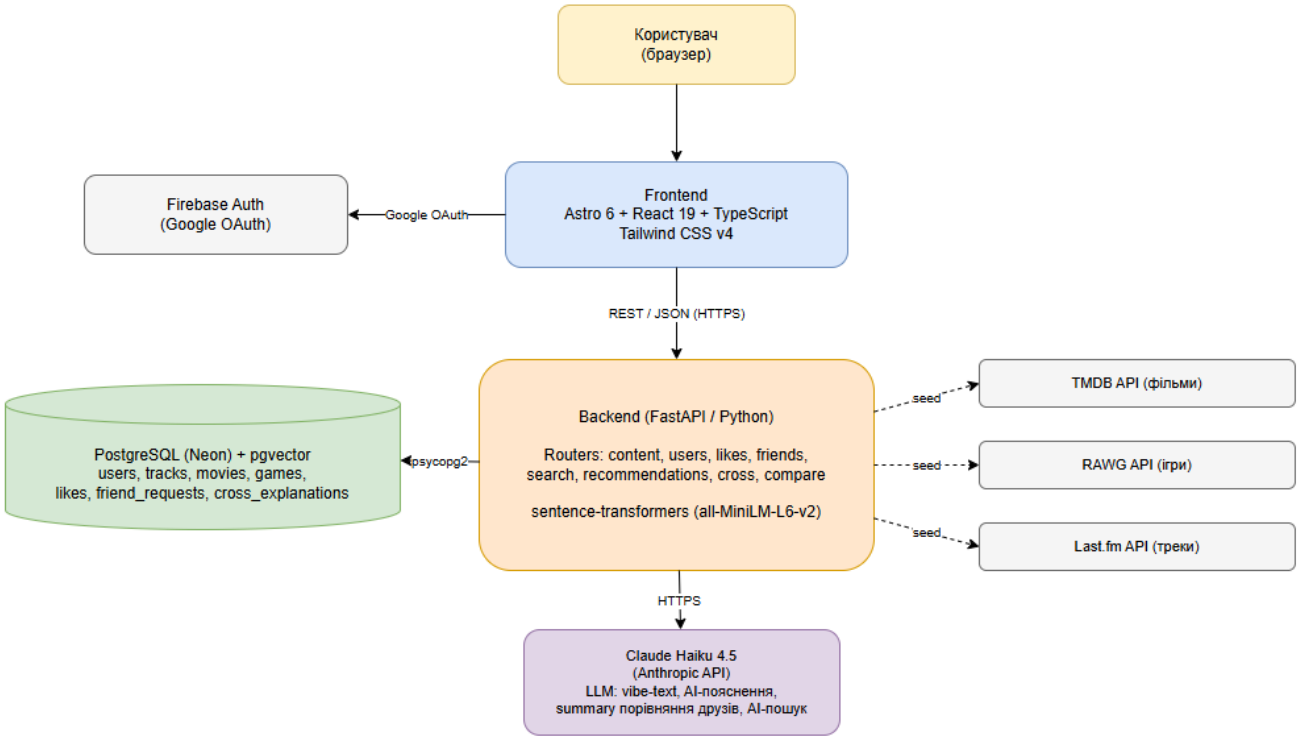


Рисунок 3.1 – Архітектура системи рекомендацій мультимедійного контенту

Наведена архітектура забезпечує чітке розмежування між рівнями системи, що спрощує масштабування та підтримку окремих компонентів незалежно один від одного.

### 3.1.2 Структура бази даних

База даних є центральним сховищем усіх даних системи – контенту, профілів користувачів, вподобань та векторних представлень. Структуру таблиць та зв'язки між ними наведено на рис. 3.2.

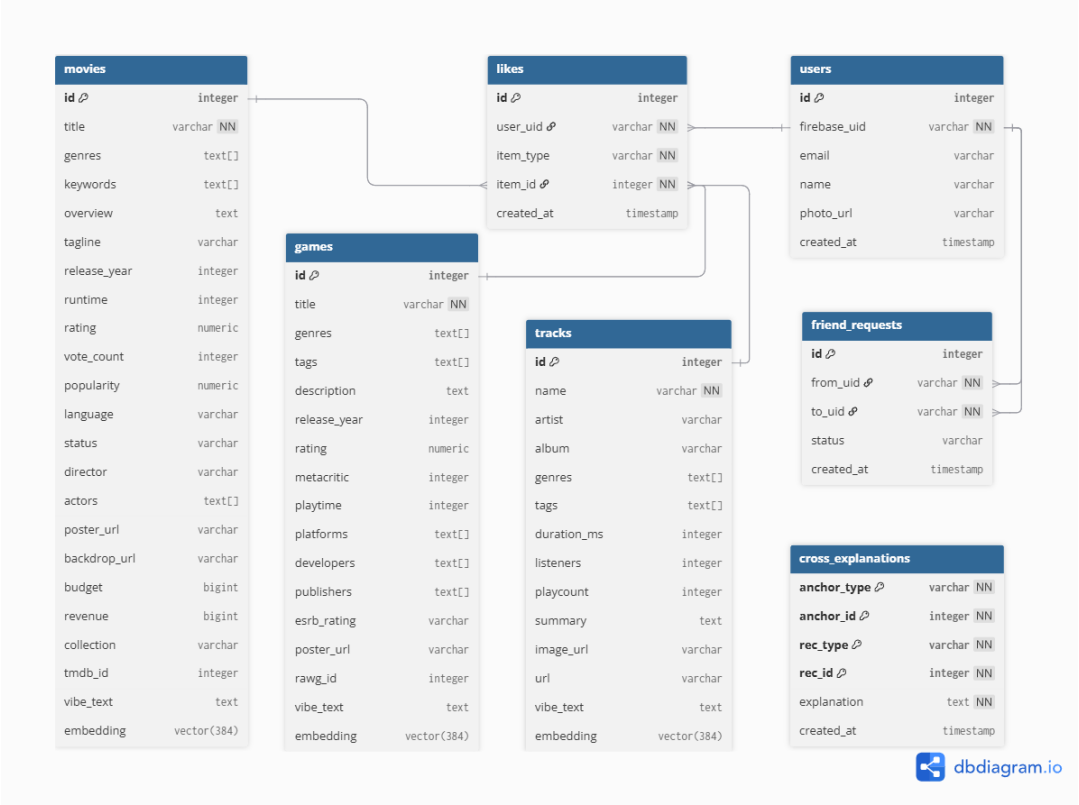


Рисунок 3.2 – ER-діаграма бази даних системи

База даних містить сім таблиць. Таблиця users зберігає профілі користувачів: ідентифікатор Firebase, електронну пошту, ім'я та фото. Паролі у базі даних не зберігаються – автентифікація повністю делегована Firebase Auth.

Таблиці movies, games та tracks містять метадані контенту з зовнішніх API. Окрім стандартних атрибутів кожен елемент має два додаткові поля: нормалізований текстовий опис та 384-вимірний векторний представлення, що використовуються у крос-доменному алгоритмі рекомендацій (п. 2.1.2).

Таблиця likes є поліморфною – поле item_type визначає домен (movie, game або track), а item_id вказує на конкретний елемент. Це дозволяє зберігати

вподобання з усіх трьох доменів в одній таблиці. Таблиця friend_requests зберігає запити на додавання до друзів з полем status. Таблиця cross_explanations є кешем пояснень крос-доменних пар – первинний ключ складається з чотирьох полів що однозначно ідентифікують пару елементів, що виключає повторні звернення до LLM для однієї і тієї ж пари.

Така структура бази даних забезпечує зберігання як реляційних даних, так і векторних представлень в єдиній системі, що спрощує архітектуру та усуває необхідність у окремому векторному сховищі.

### 3.1.3 Вхідні дані системи

Вхідні дані системи поділяються на чотири категорії: каталогові дані з зовнішніх API, дані користувацьких профілів, похідні дані LLM-обробки та дані що вводяться користувачем під час роботи системи. Каталогові дані завантажуються одноразово з трьох публічних API та зберігаються у базі даних (додаток А.1-А.3). Склад каталогу наведено у табл. 3.1.

Таблиця 3.1 – Склад каталогу системи

| **Домен** | **Джерело** | **Кількість елементів** | **Інформація про контент** |
|---|---|---|---|
| Фільми | TMDB API | 100 | Назва, жанри, ключові слова, рейтинг, акторський склад, опис, мова, тривалість часу, режисер, слоган |
| Відеоігри | RAWG API | 113 | Назва, жанри, теги, платформи, рейтинг RAWG, опис, розробники, оцінка Metacritic, видавець |
| Музичні треки | Last.fm API | 100 | Назва, виконавець, альбом, жанри, теги, виконавець, кількість прослуховувань, опис, тривалість часу |

Після завантаження каталогу для кожного елемента генерується нормалізований текстовий опис та векторне представлення відповідно до методу описаного у п. 2.1.2. Ці дані є статичними і не змінюються під час роботи системи.

Динамічні дані формуються в процесі взаємодії користувача з системою. Профіль вподобань будується поступово – кожне вподобання контенту одразу

враховується при наступному обчисленні рекомендацій. Мінімальний профіль для отримання рекомендацій складає одне вподобання у відповідному домені. Для AI-пошуку користувач вводить запит у вільній формі, який обробляється моделлю в момент запиту і не зберігається постійно.

## 3.2 Демонстрація роботи розробленої системи

У цьому підрозділі представлено демонстрацію основних функціональних модулів розробленої системи. Для кожного модуля наведено вид інтерфейсу та описано послідовність взаємодії користувача з системою. Демонстрація охоплює повний користувацький сценарій – від реєстрації та наповнення профілю до отримання однодоменних і крос-доменних рекомендацій, пошуку контенту та соціального порівняння смаків.

### 3.2.1 Авторизація користувача

Робота з системою починається з головної сторінки вебсайту, на якій представлено популярні елементи у вигляді плаваючих карток з елементами трьох доменів – фільмами, музикою та відеоіграми (рис. 3.3).

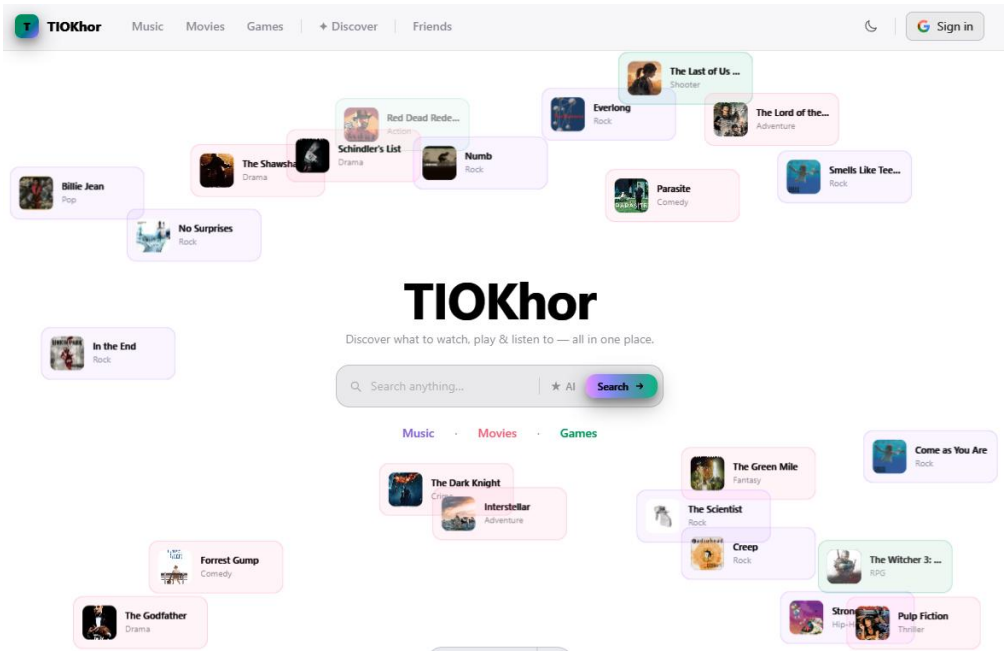


Рисунок 3.3 – Головна сторінка системи

Для авторизації користувачу потрібно натиснути на "Sign in", де користувачу потрібно обрати обліковий запис Google, після вибору користувач буде зареєстрований (рис. 3.4).

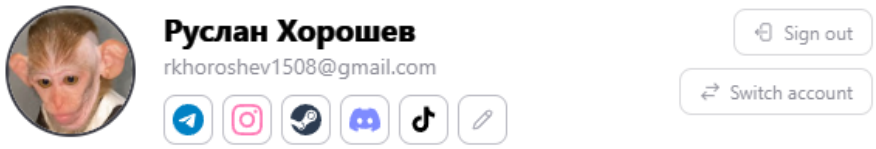


Рисунок 3.4 – Результати відображення користувача після реєстрації

Без авторизації користувач може переглядати каталоги та сторінки окремих елементів, проте функції вподобань, рекомендацій, крос-доменної навігації, а також соціального порівняння смаків стають доступними лише після входу.

### 3.2.2 Каталоги контенту

Система має три окремі каталоги – фільми, музика та відеоігри, доступ до яких реалізовано через відповідні пункти головного меню. На рис. 3.5 представлено каталог відеоігор у режимі сортування "Popular", з активованим фільтром за жанрами.

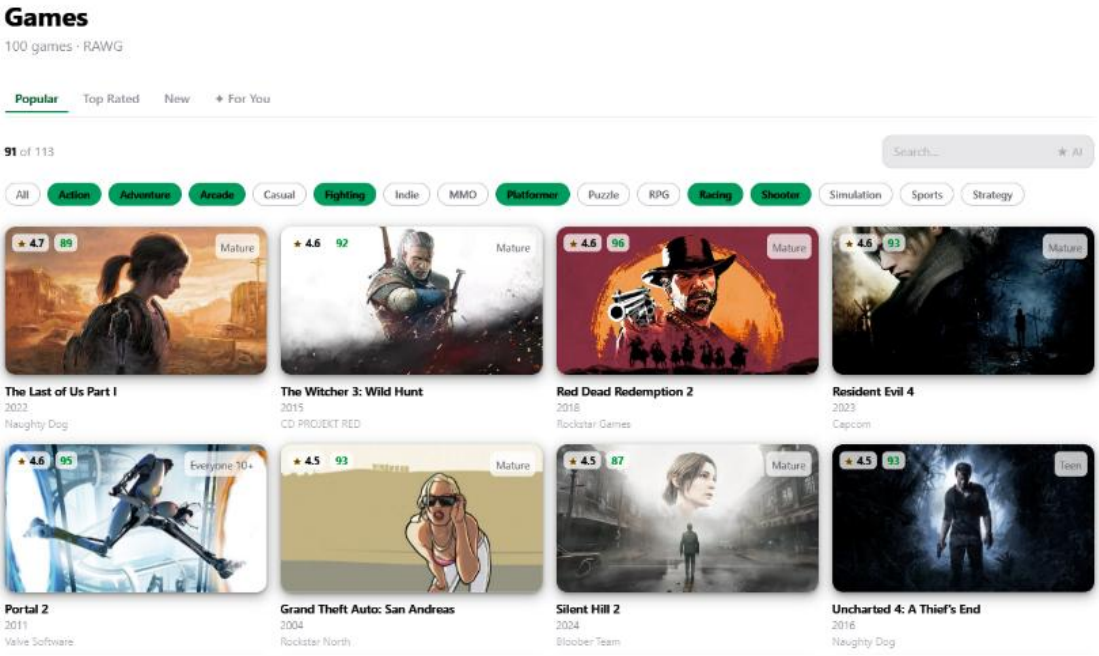


Рисунок 3.5 – Каталог відеоігор з фільтром за жанрами та сортуванням "Popular"

На рис. 3.6 представлено каталог фільмів у режимі сортування "Top Rated" (за середнім рейтингом глядачів).

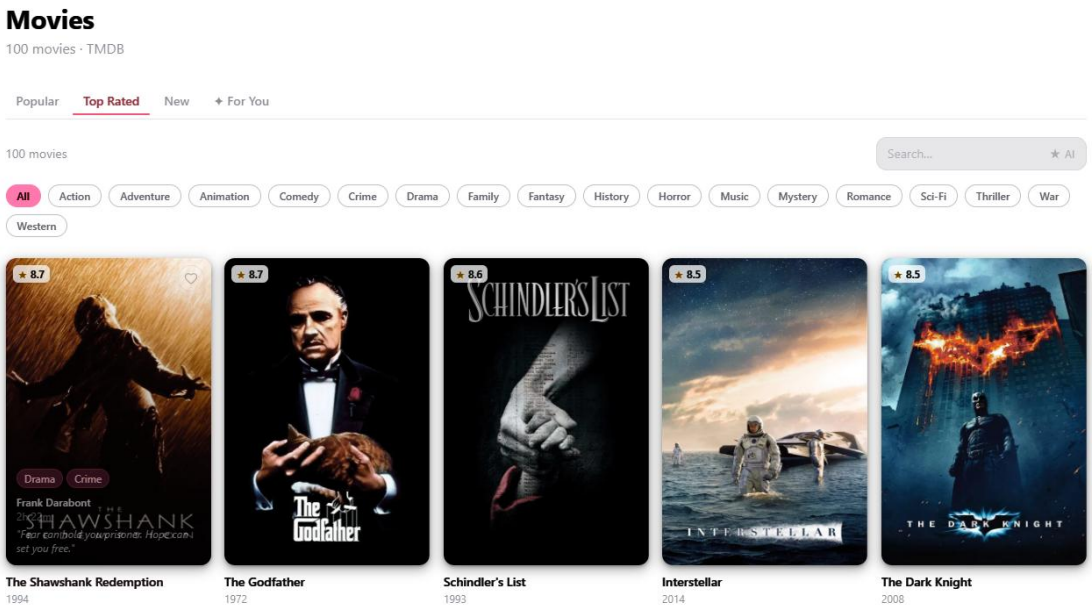


Рисунок 3.6 – Каталог фільмів у режимі сортування "Top Rated"

На рис. 3.7 представлено каталог музики у режимі сортування "New", тобто спочатку показується нова музика.

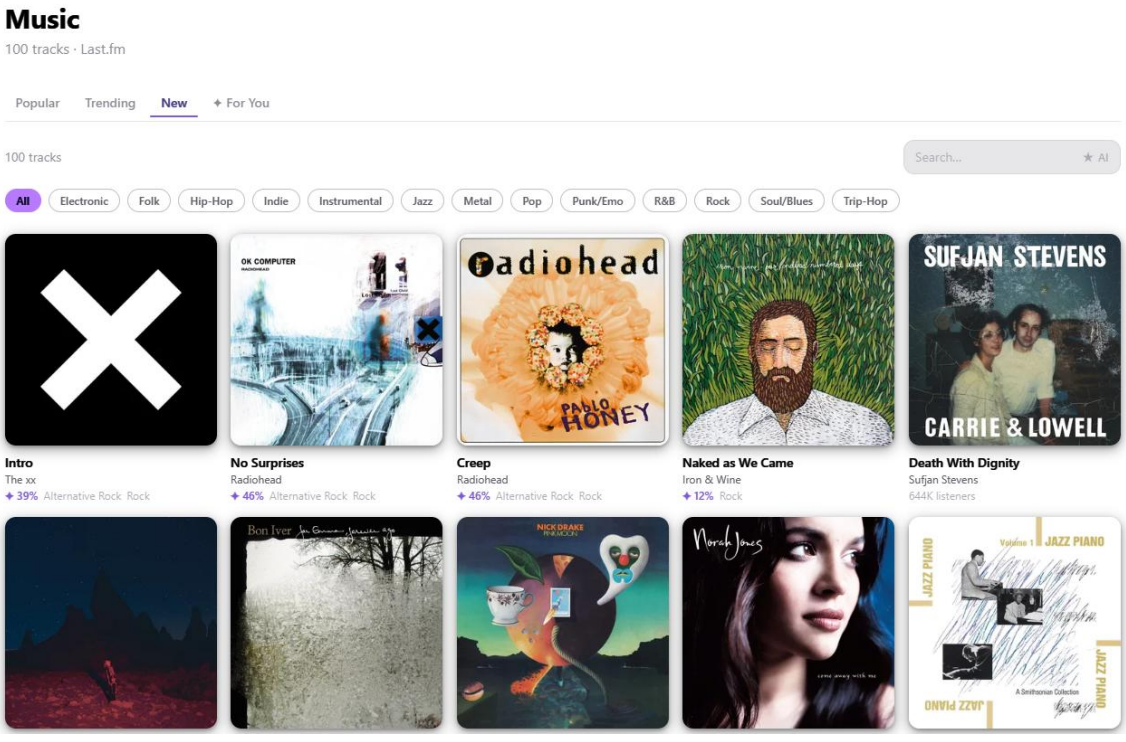


Рисунок 3.7 – Каталог музичних треків

Усі три каталоги підтримують три основні режими сортування:

– Popular сортування за метрикою популярності, які надають відповідні платформи;

– Top Rated сортування за абсолютною оцінкою якості, TMDB Rating для фільмів, Metacritic для ігор, Last.fm кількість прослуховувань музики;

– New сортування за роком випуску контенту.

### 3.2.3 Детальні сторінки елементів

Кожен елемент каталогу має власну детальну сторінку, доступну за кліком на картку у списку. Усі три типи сторінок (фільм, трек, відеогра) побудовані за уніфікованою візуальною структурою з трьома основними блоками: секція з постером та основними метаданими, теговий блок з жанрами та ключовими словами, описовий блок з повним текстовим описом елемента (рис. 3.8).

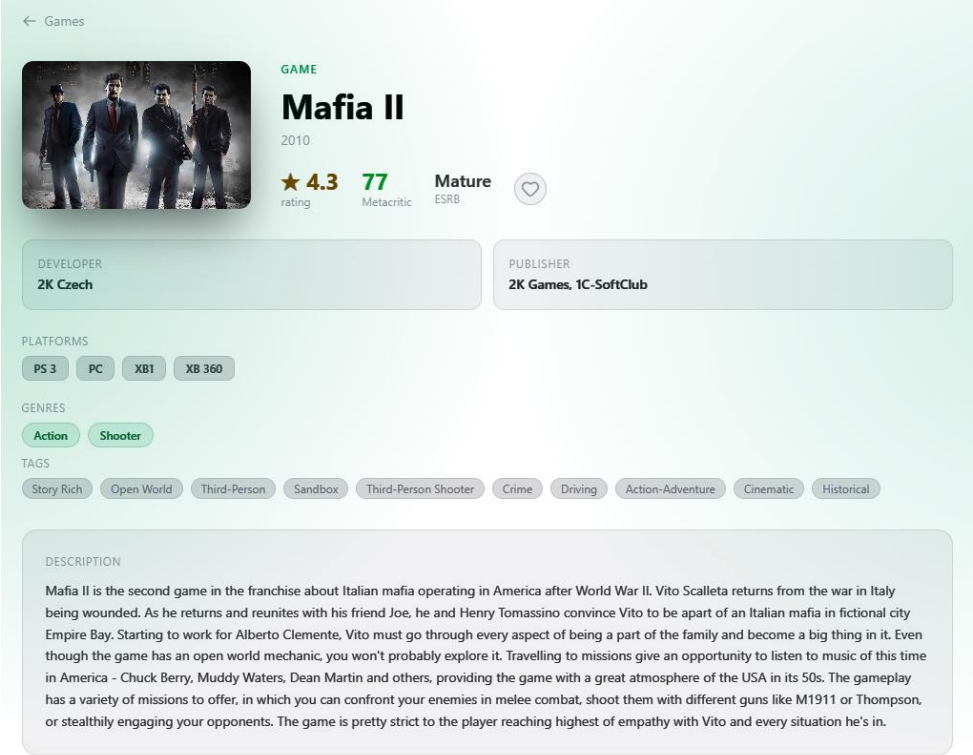


Рисунок 3.8 – Детальна сторінка відеогри

Детальні сторінки треків та відеоігор побудовані за аналогічним патерном, але містять предметно-специфічні поля, що відображають особливості кожного

домену. Внизу сторінки з детальним переглядом елемента розташована секція крос-доменних рекомендацій (Cross-Echoes), яку детально розглянуто у п. 3.2.5.

### 3.2.4 Однодоменні рекомендації

Однодоменні рекомендації реалізовано через окремий елемент меню "For You" на кожній сторінці каталогу. Рекомендації формуються в межах одного домену, тобто під час перегляду каталогу музики враховуються лише музичні вподобання користувача, аналогічно для фільмів та ігор. Алгоритмічно блок реалізує метод контентно-орієнтованої фільтрації з TF-IDF зважуванням (п. 2.1.1) на основі вподобань будується профільний вектор користувача (2.2, додаток Б.2), для кожного невподобаного елемента обчислюється косинусна подібність (2.3) (додаток Б.4), і у блоці виводиться до 50 найрелевантніших елементів (додаток Б.5). На картці кожного треку зліва відображається відсоток схожості з профілем користувача (наприклад, "87 %"), а поряд – до трьох тегів, що мають найвищу спільну вагу між профілем користувача та рекомендованим треком (рис. 3.9, додаток Б.7). Така візуалізація забезпечує пояснюваність рекомендації – користувач бачить не лише числову оцінку схожості, але і конкретні семантичні підстави, через які елемент потрапив до списку.

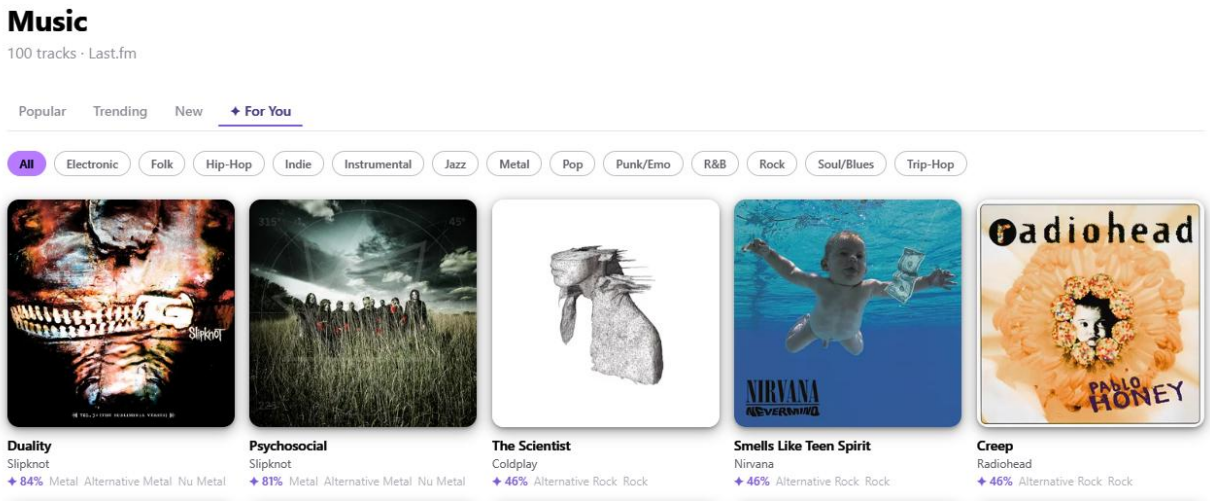


Рисунок 3.9 – Рекомендації музичних треків у блоці "For You"

Список рекомендацій кешується в пам’яті після першого обчислення для конкретного користувача (додаток Б.8). При зміні множини вподобань (додавання нової або видалення існуючої позначки) кеш автоматично інвалідується для даного користувача, і при наступному перегляді "For You" рекомендації перебудовуються з урахуванням оновленого профілю.

### 3.2.5 Крос-доменні рекомендації та AI-пояснення

Крос-доменні рекомендації реалізовано у вигляді секції "Echoes from other domains" внизу кожної детальної сторінки елементу. При перегляді елемента в одному домені система відображає до 15 найближчих елементів у кожному з інших доменів, наприклад, для фільму внизу сторінки з’являється карусель музичних треків та карусель ігор, тематично та емоційно схожих на цей фільм. Алгоритмічно цей механізм реалізує крос-доменну рекомендацію через LLM-нормалізацію та векторний пошук (додаток В.3, В.4, В.7), описану у п. 2.1.2. Приклад блоку крос-доменних рекомендацій на сторінці фільму "The Martian". Кожна картка містить відсоток гібридної подібності (2.6, 2.7, додаток Г.3), а також спільні теги, через які відбулась схожість (додаток Г.7), виділені бірюзовими мітками (рис. 3.10).

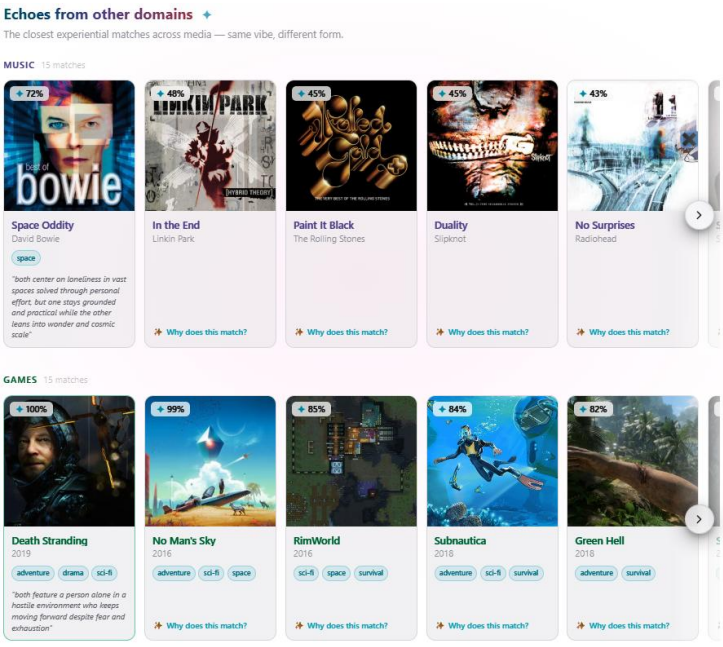


Рисунок 3.10 – Крос-доменні рекомендації на сторінці фільму "The Martian"

На кожній картці присутня кнопка "Why does this match?", при натисканні якої система звертається до моделі Claude Haiku та генерує одне коротке речення англійською, що пояснює смислову схожість елемента з рекомендованим на рівні настрою, енергетики або тематичних абстракцій (додаток Д.1).

### 3.2.6 Інтерактивна навігація крос-доменними рекомендаціями

Окрім крос-доменних рекомендацій на сторінках окремих елементів каталогу (п. 3.2.5), система пропонує спеціалізовану сторінку "Discover", де користувач може розглядати власну бібліотеку для отримання крос-доменних зв’язків, відштовхуючись від уподобань, що збережені у його профілі. Сторінка має вкладки усіх трьох доменів з кількістю вподобань у кожному. Користувач обирає опорний елемент у вигляді картки, при натисканні на яку розгортається список усіх вподобань активного домену з текстовим пошуком, також поряд розміщено кнопку випадкового вибору опорного елемента (рис. 3.11).

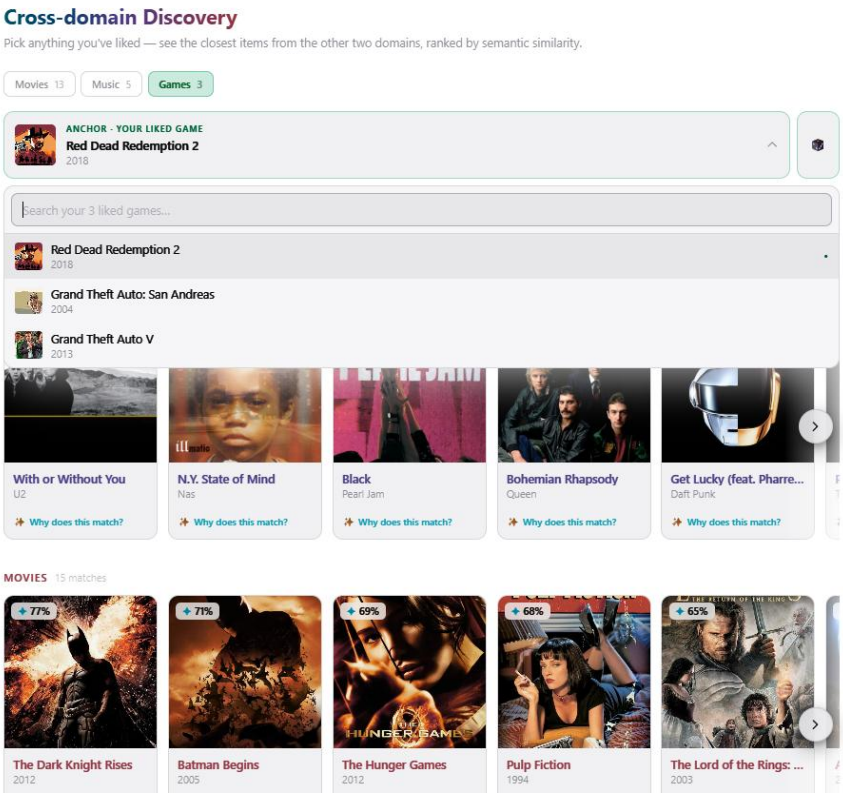


Рисунок 3.11 – Сторінка інтерактивної крос-доменної навігації "Discover"

Після вибору опорного елемента у нижній частині сторінки відображається той самий блок крос-доменних рекомендацій, що і на сторінках детального перегляду (п. 3.2.5) з відсотками подібності, спільними тегами та AI-поясненнями. Зміна опорного елемента або перехід між доменами призводить до миттєвого перерахунку списку без перезавантаження сторінки.

### 3.2.7 Соціальне порівняння смаків користувачів

Система реалізує соціальну взаємодію між користувачами через механізм додавання у друзі та порівняння смаків (додаток Д.2). Управління друзями відбувається на сторінці Friends (рис. 3.12), де користувач може надсилати запити на додавання у друзі, приймати або відхиляти отримані запити, переглядати список поточних друзів та видаляти існуючі зв’язки.

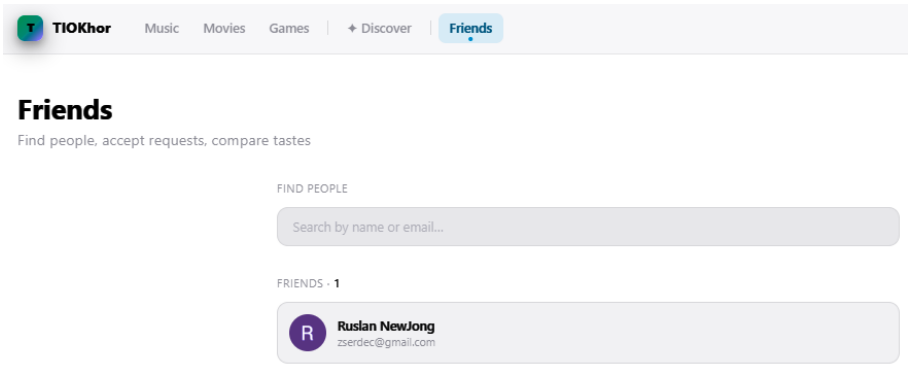


Рисунок 3.12 – Сторінка управління друзями

При перегляді публічного профілю іншого користувача відображається блок порівняння смаків (рис. 3.13). У верхній частині розташований відсоток сумісності, тобто числова метрика схожості тегово-зважених профільних векторів двох користувачів, обчислена через косинусну подібність (п. 2.1.3). Нижче – список спільних вподобань, згрупованих за доменами, з можливістю переходу на сторінку детального перегляду. Поряд є два набори тегів з топ-уподобаннями обох користувачів для швидкого огляду смакових акцентів. Окремий блок містить елементи з бібліотеки іншого користувача, теги яких відсутні у бібліотеці поточного користувача, що відображає те, чим саме смаки відрізняються.

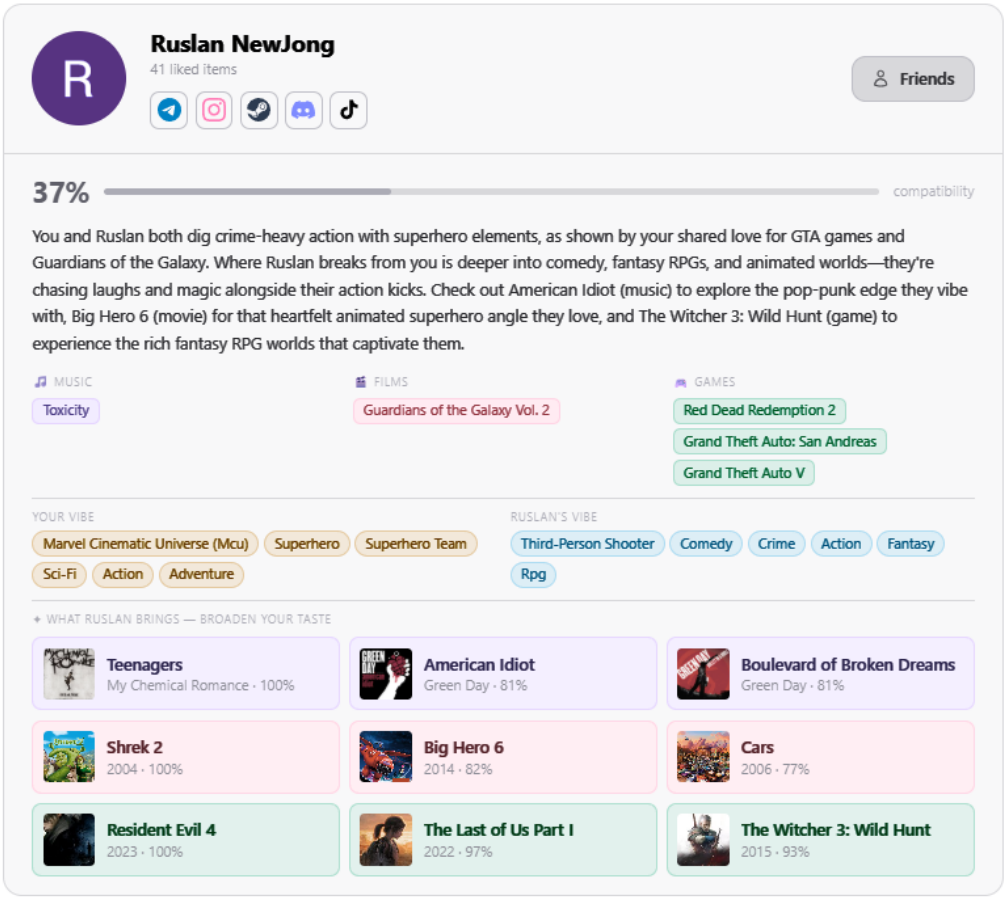


Рисунок 3.13 – Профіль іншого користувача з порівнянням смаків

Над числовими даними розташовано блок LLM-порівняння – короткий текстовий аналіз порівняння довжиною ~80-100 слів, сформований Claude Haiku на основі обчислених метрик. Інформація описує спільні риси смаків двох користувачів, окреслює характер розбіжностей та містить конкретну рекомендацію щодо контенту, який варто спробувати з бібліотеки іншого користувача.

### 3.2.8 Інтелектуальний пошук контенту

Система реалізує пошук контенту з підтримкою трьох послідовних режимів, описаних у п. 2.1.4: підрядкового, семантичного та AI-режиму (додаток Д.3). Пошуковий рядок присутній у заголовку кожної сторінки каталогу та на окремій сторінці результатів пошуку, що дозволяє виконувати запит як у межах одного домену, так і одночасно по всіх трьох.

Стандартний пошук виконується автоматично у двох режимах: підрядковому та семантичному. Підрядковий режим знаходить точні збіги за назвою, виконавцем, режисером, розробником, жанрами та тегами – миттєво, без зовнішніх запитів. Якщо підрядковий пошук знайшов менше трьох результатів у конкретному домені, автоматично активується семантичне доповнення: запит користувача перетворюється у 384-вимірний вектор через sentence-transformers (п. 2.1.2, додаток В.4), після чого через pgvector обираються до трьох найближчих за смисловою близькістю елементів (додаток В.6-В.7). Це дозволяє знаходити контент за фразами, наприклад, "epic late-night drive" або "melancholy rainy day", де точного словесного збігу немає (рис. 3.14).

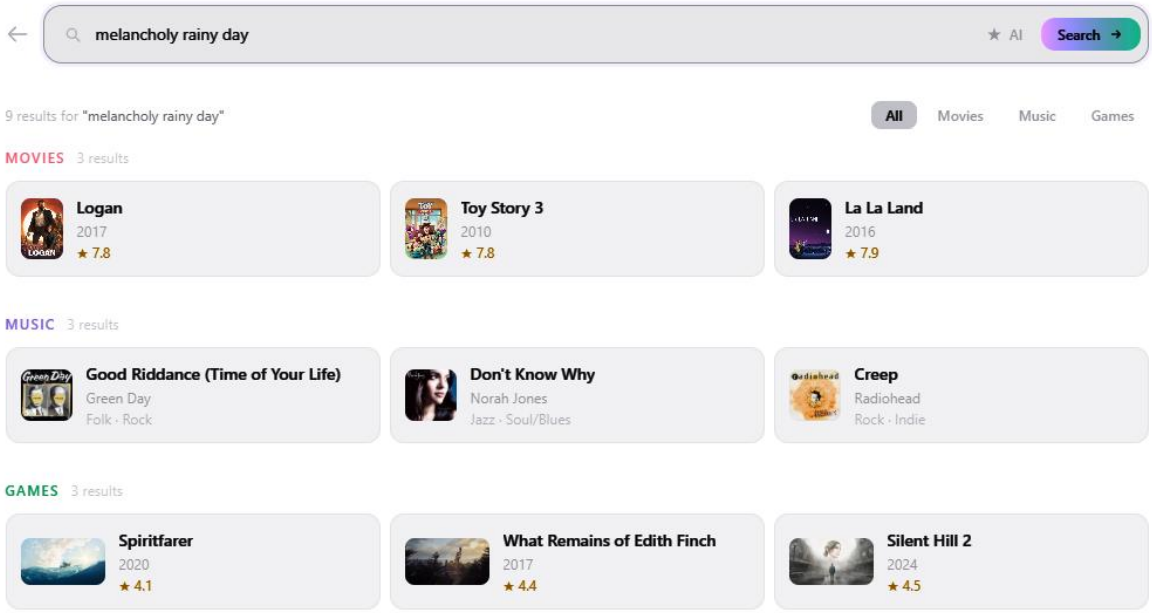


Рисунок 3.14 – Семантичний пошук контенту в трьох доменах одночасно

AI-режим активується кнопкою "AI" поряд із полем введення. У цьому режимі запит передається до Claude Haiku разом зі скороченим описом топ-40 елементів кожного домену (додаток Д.3). Модель аналізує запит, обирає до п’яти найрелевантніших елементів у кожному домені та формує речення-обґрунтування, що відображається над списком результатів (рис. 3.15). Цей режим розрахований на описові запити природною мовою, наприклад, "something feel-good for a Sunday morning", а підрядковий чи семантичний пошук не дають релевантних відповідей.

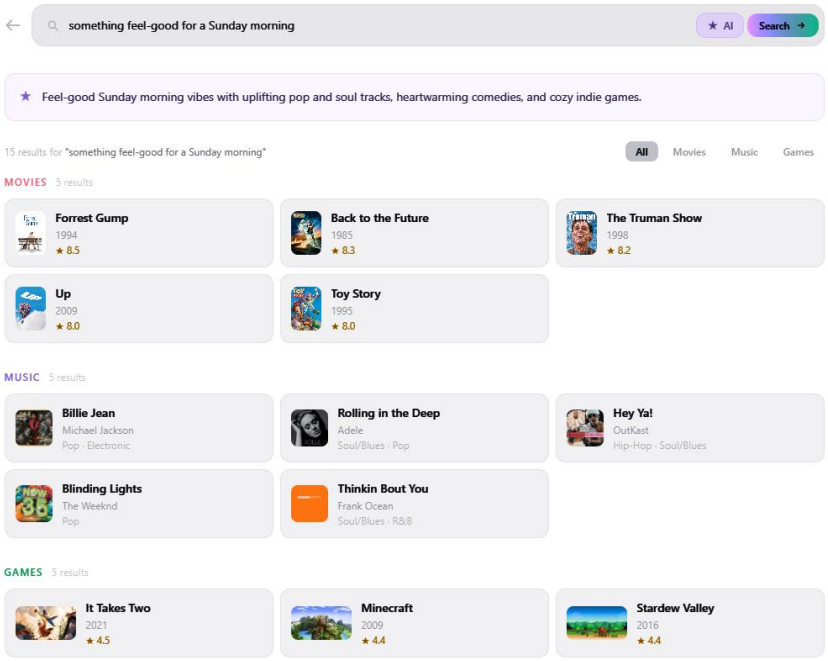


Рисунок 3.15 – Пошук у AI-режимі з природномовним поясненням

Використання великої мовної моделі відбувається виключно за явною ініціативою користувача. У стандартному режимі пошуку звернення до великої мовної моделі не відбувається, що мінімізує витрати токенів та дає миттєвий відгук на запити.

## 3.3 Аналіз отриманих результатів

Розроблена система реалізує два принципово різних типи рекомендацій, однодоменний на основі TF-IDF та крос-доменний на основі семантичного аналізу. Кожен з них потребує окремого експериментального підходу, оскільки має різну природу даних та різні можливості формалізації контрольної вибірки. У даному підрозділі представлено методологію тестування, кількісні результати, якісний аналіз та обмеження розробленої системи.

### 3.3.1 Результати однодоменної рекомендації

Для оцінки якості однодоменної рекомендації застосовано методологію тестування на основі синтетичних користувацьких профілів. Сформовано 15 синтетичних користувачів – по 5 на кожен з трьох доменів, кожен прив’язаний до

конкретного жанру (наприклад, RPG Fan, Drama Fan, Metal Fan). Множина елементів обраного жанру випадковим чином розділяється на дві частини: навчальну вибірку з 7 елементів, що додаються як вподобання користувача, та контрольну вибірку (ground truth), до якої входять усі інші елементи цього ж жанру з каталогу (додаток Е.1). Алгоритм TF-IDF будує профільний вектор користувача за його вподобаннями і повертає ранжований список рекомендацій. Для кожного користувача обчислюються метрики Precision@K, Recall@K та MRR@K при K = 5 і K = 10 (додаток Е.2) – релевантним вважається елемент з контрольної вибірки. Експеримент повторюється з 5 незалежними випадковими посівами (додаток Е.4) для отримання усереднених значень метрик у форматі середнє ± стандартне відхилення (mean ± std).

Обраний розмір навчальної вибірки у 7 елементів є компромісним, він достатній для побудови інформативного профільного вектора з кількома характерними тегами, але залишає мінімум 5 елементів у контрольній вибірці навіть для жанру з найменшою кількістю представників (Hip-Hop, 12 треків). П’ятикратне повторення з різними посівами забезпечує статистичну стабільність, тобто велике стандартне відхилення між посівами сигналізувало б про ненадійність висновків. На виході для кожного алгоритму отримуємо усереднені значення трьох метрик з довірчими інтервалами, що дозволяє статистично обґрунтовано порівнювати реалізований підхід з базовими методами та формулювати висновки про його ефективність.

Зведені результати по трьох алгоритмах наведено у табл. 3.2.

Таблиця 3.2 – Загальна якість однодоменної рекомендації

| Алгоритм | P@5 | R@5 | MRR@5 | P@10 | R@10 | MRR@10 |
|---|---|---|---|---|---|---|
| TF-IDF | 0.933 ± 0.022 | 0.407 ± 0.011 | 0.980 ± 0.025 | 0.851 ± 0.030 | 0.692 ± 0.024 | 0.980 ± 0.025 |
| Popularity | 0.104 ± 0.010 | 0.038 ± 0.007 | 0.210 ± 0.019 | 0.124 ± 0.009 | 0.091 ± 0.012 | 0.244 ± 0.024 |
| Random | 0.165 ± 0.049 | 0.053 ± 0.016 | 0.289 ± 0.035 | 0.172 ± 0.030 | 0.112 ± 0.017 | 0.320 ± 0.027 |

Реалізований алгоритм TF-IDF демонструє значну перевагу над базовими методами (додаток Е.3) по всіх трьох метриках при обох значеннях K. Precision@5

0.933 ± 0.022 означає, що з кожних 5 рекомендованих елементів у середньому 4.7 виявляються релевантними цільовому жанру користувача, тобто понад 93% рекомендацій є точними. При більш широкому K = 10 точність становить 0.851 ± 0.030, що відповідає 8.5 правильних рекомендацій з 10. MRR@10 = 0.980 ± 0.025 свідчить про те, що перший релевантний елемент у списку рекомендацій майже завжди розташований на першій позиції – критично важлива якість для користувацького досвіду, оскільки найкращий результат завжди опиняється вгорі.

У порівнянні з базовими методами реалізований алгоритм демонструє підсилення (lift) у 6.9 раза за Precision@10 та 7.6 раза за Recall@10 порівняно з Popularity, а також у 5.0 та 6.2 раза відповідно у порівнянні з Random. Отримані результати свідчать про те, що алгоритм використовує саме семантичну подібність профілю користувача через IDF-зваження тегів, а не загальну популярність контенту або випадковий збіг. Низькі значення стандартного відхилення (від 0.011 до 0.030) свідчать про стабільність результатів незалежно від того, які саме елементи потрапили до навчальної вибірки, а які – до контрольної.

Порівняння алгоритмів за загальними метриками наведено на рис. 3.16.

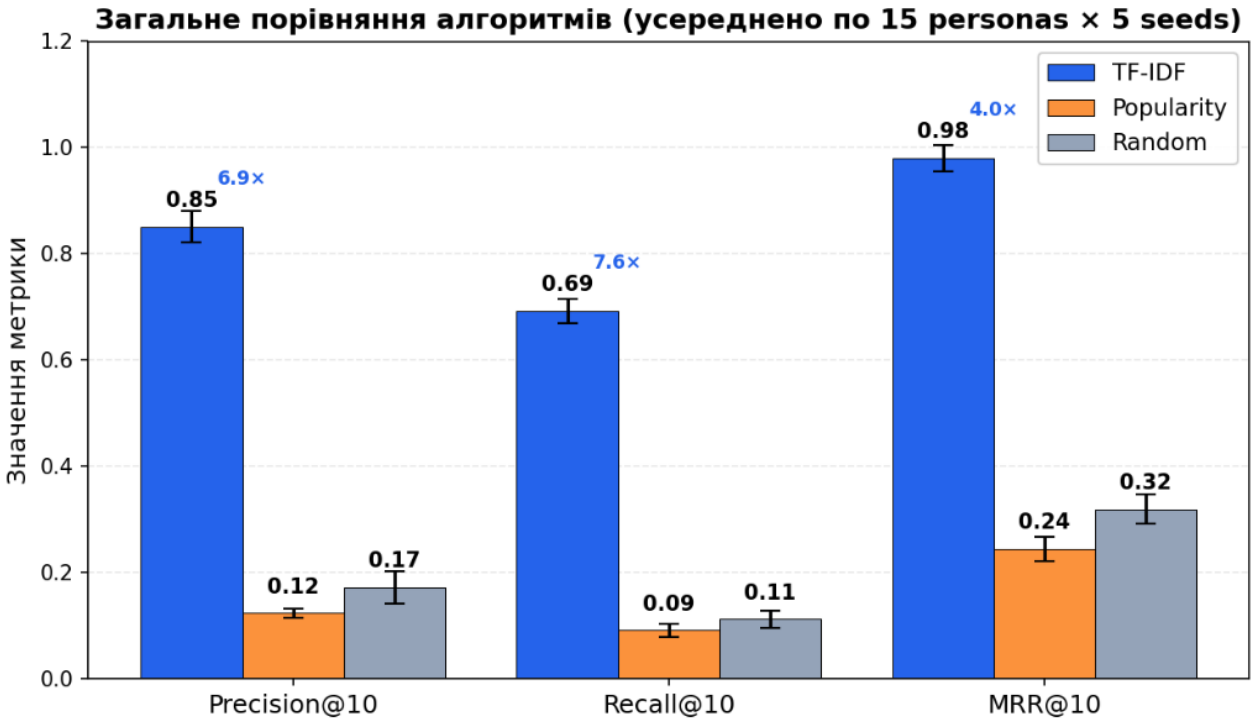


Рисунок 3.16 – Підсилення алгоритму над базовими методами

Для перевірки рівномірності якості алгоритму між доменами проведено детальний аналіз результатів по кожній з трьох категорій контенту (табл. 3.3).

Таблиця 3.3 – Якість однодоменної рекомендації по доменах при K = 10

| Домен | Алгоритм | Precision@10 | Recall@10 | MRR@10 |
|---|---|---|---|---|
| **Ігри** | TF-IDF | 0.856 ± 0.073 | 0.621 ± 0.064 | 0.970 ± 0.060 |
| | Popularity | 0.060 ± 0.018 | 0.042 ± 0.015 | 0.203 ± 0.093 |
| | Random | 0.144 ± 0.054 | 0.097 ± 0.036 | 0.315 ± 0.149 |
| **Фільми** | TF-IDF | 0.836 ± 0.034 | 0.568 ± 0.016 | 0.970 ± 0.060 |
| | Popularity | 0.232 ± 0.030 | 0.139 ± 0.020 | 0.448 ± 0.052 |
| | Random | 0.248 ± 0.035 | 0.123 ± 0.014 | 0.419 ± 0.034 |
| **Треки** | TF-IDF | 0.860 ± 0.000 | 0.889 ± 0.002 | 1.000 ± 0.000 |
| | Popularity | 0.080 ± 0.022 | 0.093 ± 0.027 | 0.083 ± 0.017 |
| | Random | 0.124 ± 0.053 | 0.117 ± 0.045 | 0.225 ± 0.095 |

Графічне порівняння алгоритмів по доменах представлено на рис. 3.17.

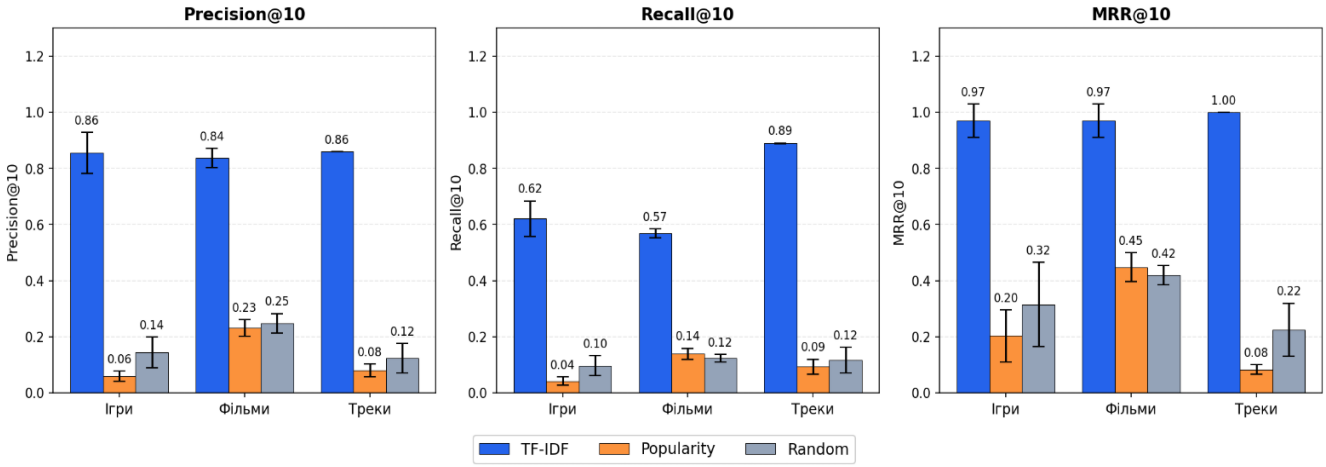


Рисунок 3.17 – Порівняння алгоритмів за метриками Precision@10, Recall@10 та MRR@10 по доменах

Результати по доменах підтверджують стабільність якості алгоритму (додатки Б.1-Б.8) незалежно від типу контенту. Найвищі показники зафіксовано у домені треків, оскільки MRR@10 = 1.000 з нульовим відхиленням (досконале ранжування – перший релевантний трек завжди на першій позиції) та Recall@10 = 0.889 (покриває близько 89% релевантних треків у топ-10). У домені

ігор Precision@10 = 0.856, у домені фільмів 0.836. Невелика варіативність між доменами пояснюється насиченістю тегової розмітки: треки мають більш чітку розмітку завдяки user-generated тегам Last.fm. Низькі значення стандартного відхилення (від 0.000 до 0.073) у всіх трьох доменах додатково підтверджують надійність результатів.

Отримані результати підтверджують ефективність TF-IDF підходу для однодоменної задачі рекомендацій. Алгоритм коректно рекомендує понад 80% елементів цільового жанру з майже досконалим ранжуванням найрелевантнішого елемента на першій позиції та підсиленням у 5-8 разів над базовими методами, що доводить використання саме семантики тегів профілю користувача.

### 3.3.2 Результати крос-доменної рекомендації

Крос-доменна задача рекомендації відрізняється від однодоменної відсутністю розміченого еталону – неможливо однозначно визначити, які саме елементи з домену музики мають відповідати конкретним фільмам або відеоіграм. Тому замість класичних метрик Precision/Recall з контрольною вибіркою застосовано порівняння реалізованого алгоритму з випадковим базовим рівнем (random baseline) на однакових наборах опорних елементів.

Для кожної з шести спрямованих доменних пар (track-movie, track-game, movie-track, movie-game, game-track, game-movie) випадковим чином обирається 25 опорних елементів з відповідного домену. Для кожного опорного елемента обчислюються три варіанти результатів:

– реалізований гібридний алгоритм (2.4–2.6, додаток Г.1);

– семантичний варіант без тегового бонусу (2.4) для оцінки внеску бонусу окремо;

– 25 випадкових елементів іншого домену як шумовий поріг.

Для кожного варіанту обчислюється середня семантична схожість, середня кількість спільних тегів та частка рекомендованих елементів зі щонайменше одним спільним тегом – після чого результати усереднюються по 5 незалежних

випадкових посівах (додатки Е5-Е.6). Основним показником якості є lift як співвідношення значення метрики hybrid до значення random – чим вищий lift, тим сильніше алгоритм відрізняється від випадкового підбору.

Загальні результати, усереднені по шести доменних парах та 5 посівах, наведено у табл. 3.4.

Таблиця 3.4 – Загальна якість крос-доменної рекомендації

| Алгоритм | Семантична схожість | Спільні теги | Частка з перетином тегів |
|---|---|---|---|
| **Hybrid** | 0.723 ± 0.007 | 0.647 ± 0.055 | 34.8% ± 3.1% |
| **Semantic** | 0.723 ± 0.007 | 0.343 ± 0.047 | 22.7% ± 2.7% |
| **Random** | 0.599 ± 0.010 | 0.245 ± 0.031 | 17.1% ± 2.2% |

Порівняння алгоритмів з показниками підсилення наведено на рис. 3.18.

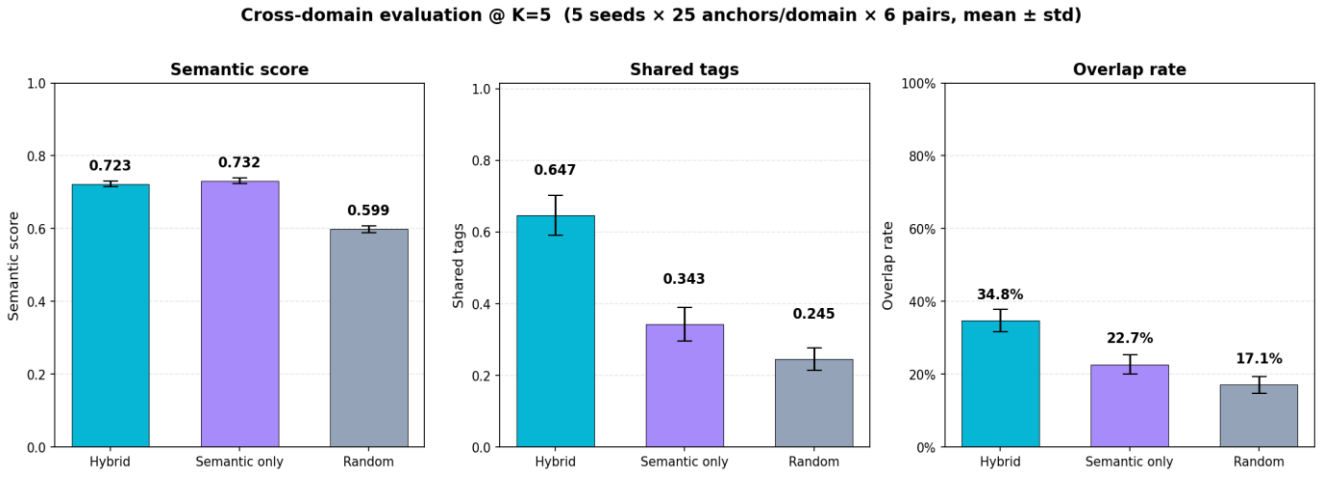


Рисунок 3.18 – Підсилення крос-доменного алгоритму над базовими методами

Реалізований гібридний алгоритм (додаток Г.4-Г.6) демонструє суттєву перевагу над випадковим підбором за всіма трьома метриками. Найвищі показники підсилення зафіксовано за метриками тегового перетину (додаток Г.2), середня кількість спільних тегів у рекомендованих елементах у 2.6 рази вища за випадковий рівень (0.647 проти 0.245), а частка елементів зі щонайменше одним спільним тегом у 2.0 рази вища (34.8% проти 17.1%). Це означає, що алгоритм цілеспрямовано знаходить тематично пов’язані елементи у двох інших доменах, а не повертає випадковий контент.

Метрика семантичної схожості також демонструє статистично значущу перевагу гібридного алгоритму над random baseline (0.723 проти 0.599, lift 1.2 раза), хоча її абсолютна різниця менша за різницю у тегових метриках. Це пояснюється архітектурою LLM-нормалізованих векторних представлень, тому що всі описи елементів каталогу генеруються Claude Haiku у спільний словник досвіду сприйняття, через це навіть випадкові пари елементів мають базовий рівень семантичної схожості близько 0.6. У такому щільному просторі різниця у 0.12 між сигналом (hybrid 0.72) та шумом (random 0.60) є значущою, оскільки відображає реальну семантичну відповідність між опорним та рекомендованими елементами каталогу.

Практично важливим висновком є порівняння гібридного варіанту з чисто семантичним (без тегового бонусу). Метрика семантичної схожості для двох варіантів майже однакова (0.723 проти 0.732), що свідчить про те, що додавання тегового бонусу не погіршує семантичної якості. Проте за тегово-орієнтованими метриками гібридний варіант показує майже удвічі вищі результати, 0.647 проти 0.343 за середньою кількістю спільних тегів та 34.8% проти 22.7% за часткою елементів з перетином. Це підтверджує, що компонент тегового бонусу (2.5) реально працює як тематичний переранжувач (додаток Г.5) поверх семантичного базису, не псуючи його якості, але суттєво підвищуючи структурну релевантність рекомендацій.

Для перевірки рівномірності якості алгоритму у різних напрямках крос-доменного зіставлення проведено детальний аналіз результатів по кожній з шести спрямованих пар (табл. 3.5).

Таблиця 3.5 – Якість крос-доменної рекомендації за спрямованими доменними парами

| **Пара напрямків** | **Алгоритм** | **Семантична схожість** | **Спільні теги** |
|---|---|---|---|
| track - movie | Hybrid | 0.732 ± 0.006 | 0.051 ± 0.014 |
| | Random | 0.611 ± 0.007 | 0.005 ± 0.004 |

Кінець таблиці 3.5

| Пара напрямків | Алгоритм | Семантична схожість | Спільні теги |
|---|---|---|---|
| track - game | Hybrid | 0.725 ± 0.004 | 0.104 ± 0.043 |
| | Random | 0.574 ± 0.014 | 0.032 ± 0.021 |
| movie - track | Hybrid | 0.724 ± 0.006 | 0.056 ± 0.023 |
| | Random | 0.603 ± 0.003 | 0.005 ± 0.006 |
| movie - game | Hybrid | 0.729 ± 0.007 | 1.766 ± 0.098 |
| | Random | 0.609 ± 0.012 | 0.659 ± 0.067 |
| game - track | Hybrid | 0.704 ± 0.013 | 0.168 ± 0.037 |
| | Random | 0.578 ± 0.013 | 0.018 ± 0.017 |
| game - movie | Hybrid | 0.722 ± 0.009 | 1.734 ± 0.117 |
| | Random | 0.617 ± 0.009 | 0.752 ± 0.074 |

Графічне порівняння за метрикою семантичної схожості представлено на рис. 3.19.

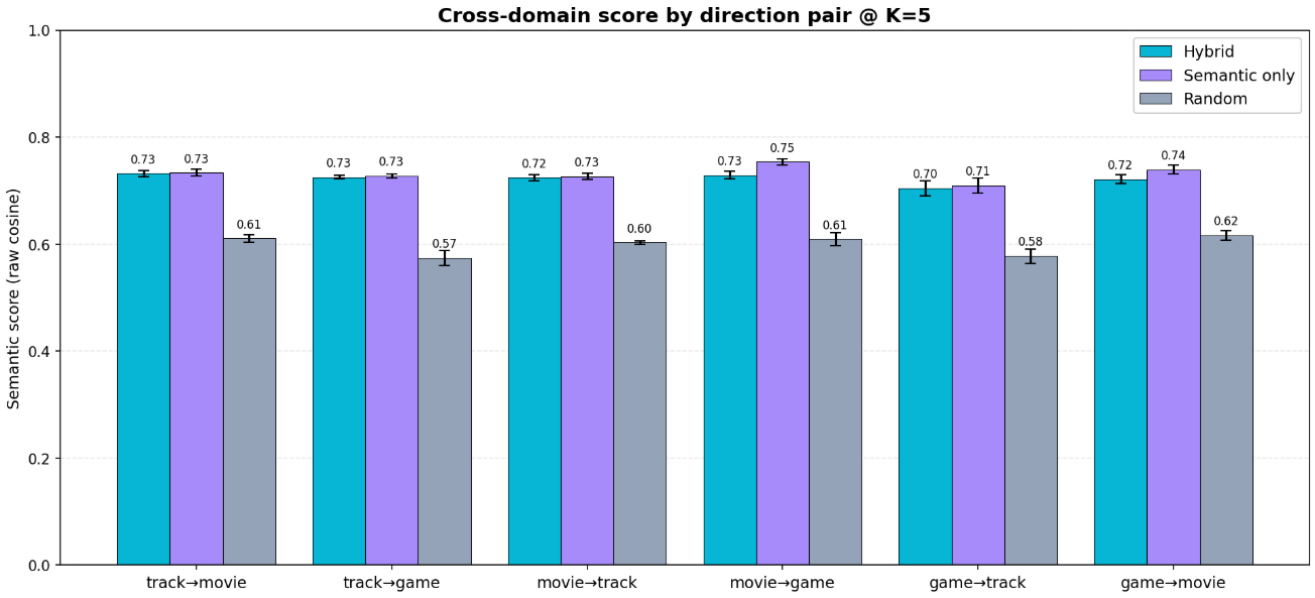


Рисунок 3.19 – Семантична схожість крос-доменних рекомендацій по парах напрямків

Слід зазначити, що абсолютні значення тегового перетину не порівнюються між парами через відмінність словників – порівняння коректне лише в межах однієї пари між hybrid та random. Гібридний алгоритм демонструє стабільну перевагу над випадковим базовим рівнем за всіма шістьма напрямками. Семантична схожість

становить 0.704–0.732 для hybrid проти 0.574–0.617 для random. Найвищі абсолютні значення тегового перетину зафіксовано у напрямках movie-game та game-movie (1.766 та 1.734 спільних тегів відповідно), що пояснюється спільним словником жанрів у цих доменах ("Action", "Adventure", "Sci-Fi" зустрічаються і у фільмах, і у відеоіграх). У напрямках за участю музичних треків теговий перетин практично відсутній (0.051–0.104), оскільки музичні жанри не мають прямих відповідників серед жанрів фільмів та ігор – саме тут LLM-нормалізація є єдиним механізмом знаходження зв’язків, і lift за семантичною схожістю у цих парах досягає максимуму (1.20–1.26). Це розмежовує роль двох компонентів, тобто тегового бонусу – для доменів зі спільним словником, LLM-семантики – для доменів без нього.

Отримані числові результати підтверджують ефективність запропонованого vibe-pipeline (додаток В.1-В.2, Г.6) для крос-доменної рекомендації. Алгоритм демонструє статистично значуще підсилення над випадковим базовим рівнем по всіх трьох метриках та у всіх шести напрямках доменних пар. Гібридне ранжування з тегового бонусом покращує структурну релевантність рекомендацій майже удвічі у порівнянні з чисто семантичним варіантом, не погіршуючи семантичної якості. Це доводить, що запропонована архітектурна комбінація LLM-нормалізації описів та гібридного pgvector-ранжування є ефективним рішенням для задачі крос-доменної рекомендації мультимедійного контенту.

### 3.3.3 Якісний аналіз крос-доменних рекомендацій

Кількісні метрики, представлені у п. 3.3.2, демонструють статистичну ефективність крос-доменного алгоритму, проте не передають характеру семантичних зв’язків, які він фактично знаходить. Для оцінки практичної релевантності системи розглянуто опорні елементи, що репрезентують типові сценарії її роботи, наприклад, сильний збіг у межах однієї тематичної франшизи, концептуальна паралель між принципово різними доменами, відповідність за енергетикою та темпом, а також слабкий збіг як приклад чесної самооцінки системи

(додаток Е.5). Усі наведені числові значення отримано безпосередньо з реалізованої системи на момент тестування.

Сильний збіг у межах франшизи. Як опорний елемент обрано фільм "The Dark Knight" (жанри Action, Crime, Thriller). Алгоритм визначив топ-3 найближчі відеоігри з наступними калібровими відсотками схожості (2.7). Це Batman: Arkham Knight (100%), Batman: Arkham City (89%) та Phasmophobia (87%). Першу рекомендацію забезпечує поєднання високої семантичної схожості (0.739) та чотирьох спільних тегів (action, dark, knight, superhero), що відображає очевидний зв’язок у межах франшизи Бетмена. Третя рекомендація демонструє цікавий крос-жанровий перехід – горор-гра Phasmophobia потрапила до топ-3 за фільмом про action/crime завдяки спільному настрою тривоги, темному міському оточенню та поступовому нарощуванню напруги. AI-пояснення системи для цієї пари: "both build tension through slow planning that explodes into chaos, though one is urban crime while the other is supernatural horror" – точно фіксує спільну рису та різницю згідно з підказкою моделі для пари з помірним рівнем схожості (п. 2.1.4).

Концептуальна крос-доменна паралель. Опорним елементом обрано відеогру "The Witcher 3: Wild Hunt" (Action, RPG) для перевірки зворотного напрямку рекомендацій (game-movie). Топ-3 фільмів, визначених алгоритмом: The Lord of the Rings: The Two Towers (100%, спільні теги action, fantasy, magic), The Lord of the Rings: The Return of the King (97%) та The Lord of the Rings: The Fellowship of the Ring (94%) – повна трилогія посідає верхні три позиції. Цей результат заслуговує окремої уваги, оскільки система самостійно відновила тематичний зв’язок між фентезі-RPG та класичною кінотрилогією, без жодних прямих посилань між цими творами у вихідних метаданих TMDB та RAWG. AI-пояснення для топ-1 "both oscillate between quiet moments and explosive confrontation, blending mythic wonder with gritty human struggle across long, immersive journeys" – характеризує спільний наратив подорожі без згадування назв творів. Цей приклад є ключовим аргументом на користь LLM-нормалізації як методу побудови крос-доменних зв’язків (п. 2.1.2).

Відповідність за енергетикою та темпом. Опорним елементом обрано "DOOM Eternal" (Action, Shooter) – гру з виразним характером інтенсивної кінетики та постапокаліптичним настроєм. Топ-3 фільмів: Mad Max: Fury Road (93%, спільні теги action, sci-fi), Avengers: Infinity War (80%) та Iron Man 2 (79%). Найвища рекомендація демонструє точну відповідність за енергетикою та постапокаліптичною атмосферою, незважаючи на принципово різний наратив. AI-пояснення для першої рекомендації "both push through with relentless fast aggression and dark humor while fighting against impossible odds". Цей приклад підтверджує, що алгоритм оперує не лише поверхневими жанровими тегами, а й більш тонкими атрибутами – темпом, інтенсивністю, домінантною емоцією – закодованими у нормалізованому vibe-text.

Слабкий збіг та поведінка системи. Як приклад чесної поведінки системи у випадку слабкого зв’язку розглянуто пісню "Black" гурту Pearl Jam (Rock) інтимну меланхолійну композицію про втрату. Топ-3 крос-доменних рекомендацій у домені відеоігор: Spiritfarer (61%), Undertale (56%) та Silent Hill 2 (51%). Жодна рекомендація не перевищила 75%, що активувало інший режим генерації AI-пояснень з конструкцією "share X, but..." згідно з логікою розгалуження за рівнем схожості (п. 2.1.4). AI-пояснення для Spiritfarer: "share slow sadness and moments of tender goodbye, but one stays isolated while the other brings warmth and acceptance", система самостійно сигналізує користувачу, що тематичний зв’язок присутній, але не є повним, через явне зазначення розбіжності. Це підтверджує, що калібровка відсотка (2.7) та адаптивний запит LLM спільно забезпечують диференційований рівень впевненості рекомендацій, не перетворюючи систему на джерело завищених оцінок схожості.

Узагальнені результати якісного аналізу наведено у табл. 3.6, що містить топ-3 крос-доменних рекомендацій для усіх чотирьох розглянутих опорних елементів.

Таблиця 3.6 – Якісні приклади крос-доменних рекомендацій

| **Опорний елемент (домен)** | **Топ-3 рекомендованих елементів (домен)** | **Схожість %** | **Спільні теги** |
|---|---|---|---|
| The Dark Knight(movie) | Batman: Arkham Knight (game) | 100 | action, dark, knight, superhero |
| | Batman: Arkham City (game) | 89 | action, dark, superhero |
| | Phasmophobia (game) | 87 | action, dark, thriller |
| The Witcher 3: Wild Hunt (game) | The Lord of the Rings: The Two Towers (movie) | 100 | action, fantasy, magic |
| | The Lord of the Rings: The Return of the King (movie) | 97 | action, fantasy, magic |
| | The Lord of the Rings: The Fellowship of the Ring (movie) | 94 | action, fantasy, magic |
| DOOM Eternal (game) | Mad Max: Fury Road (movie) | 93 | action, sci-fi |
| | Avengers: Infinity War (movie) | 80 | action, sci-fi |
| | Iron Man 2 (movie) | 79 | action, sci-fi |
| Black (track, Pearl Jam) | Spiritfarer (game) | 61 | – |
| | Undertale (game) | 56 | – |
| | Silent Hill 2 (game) | 51 | – |

Узагальнений висновок з якісного аналізу полягає у виявленні трьох характерних режимів роботи розробленої системи. По-перше, алгоритм коректно ідентифікує очевидні франшизні зв’язки (The Dark Knight – Batman: Arkham Knight), де високий семантичний сигнал підкріплений значним теговим перетином, тобто гібридного ранжування (2.6) досягає максимуму через одночасну роботу обох компонентів. По-друге, система здатна знаходити концептуальні паралелі між принципово різними доменами (The Witcher 3 – трилогія The Lord of the Rings) там, де перетин жанрів обмежений, ключову роль відіграє виключно LLM-нормалізована семантика, що проєктує описи у спільний experiential простір (п. 2.1.2). По-третє, система чесно фіксує слабкі зв’язки (Pearl Jam – Spiritfarer), де калібровка (2.7) не дозволяє відсотку перевищити природну межу, а адаптивний запит LLM сигналізує користувачу про неповний збіг через явне виокремлення розбіжностей. Така диференційована поведінка є важливою для довіри користувача до рекомендаційної системи, оскільки усуває ризик систематичного завищення

оцінок схожості, що є типовою проблемою рекомендаційних систем без калібровки відображуваних значень.

### 3.3.4 Обмеження реалізованої системи рекомендацій

Запропонована система реалізує повний цикл функцій, передбачених постановкою задачі (п. 1.3), та демонструє значущі результати у задачах однодоменної та крос-доменної рекомендації (п. 3.3.1, п. 3.3.2). Проте отримані результати слід інтерпретувати з урахуванням ряду обмежень, що визначають межі застосовності системи у поточному стані та формують напрямки подальших досліджень.

Прототипний обсяг каталогу. Система оперує каталогом з 313 елементів (табл. 3.1), що є прототипним обсягом для забезпечення контрольованих умов оцінки. Реалізовані алгоритми мають лінійну часову складність для TF-IDF фільтрації та логарифмічну для векторного пошуку через pgvector, тому збільшення каталогу до сотень тисяч елементів є інженерною задачею і не змінює методологію оцінки. Підсилення над базовими методами (lift) є інваріантним до розміру каталогу, оскільки обчислюється як відношення метрик нашого алгоритму до базових – обидва значення масштабуються пропорційно при зміні обсягу даних. При збільшенні каталогу очікується незначне зниження Precision через зростання конкуренції між елементами, однак ранжування за косинусною подібністю зберігає відносну стабільність, що підтверджується стабільністю MRR@10 на рівні $0.987 \pm 0.016$ за 5 випадкових прогонів (п. 3.3.1) – стандартне відхилення близько 2% свідчить про збіжність алгоритму.

Методологія експериментальної оцінки. Тестування проведено на синтетичних користувацьких профілях, що є стандартним підходом для академічної оцінки рекомендаційних систем [10, 14] та дозволяє контрольовано вимірювати якість алгоритму з чітко визначеною контрольною вибіркою. Альтернативне A/B-тестування на реальних користувачах потребує запущеного сервісу з тривалим накопиченням трафіку. Для крос-доменної задачі додатковим

обмеженням є відсутність розміченого еталону, тому у п. 3.3.2 застосовано порівняння з random baseline як альтернативу класичним метрикам точності.

Холодний старт. На відміну від CF-систем, content-based підхід не має проблеми холодного старту для нового контенту – кожен елемент каталогу одразу отримує vibe-text та векторне представлення і може бути рекомендований незалежно від кількості взаємодій. Проте для абсолютно нового користувача без жодного вподобання персоналізовані рекомендації залишаються неможливими, що є фундаментальним обмеженням саме контентно-орієнтованого підходу. Таким користувачам система пропонує сортовані каталоги у режимах Popular, Top Rated та New (п. 3.2.2) та інтелектуальний пошук природною мовою (п. 3.2.8). Перше вподобання активує повноцінну персоналізацію.

Якість LLM-генерованих описів. Ефективність крос-доменного механізму безпосередньо залежить від якості нормалізованих vibe-text та AI-пояснень, що генеруються моделлю Claude Haiku 4.5 за заданим системним запитом. У роботі ця якість підтверджена опосередковано через досягнуті крос-доменні метрики (п. 3.3.2) та якісні приклади рекомендацій (п. 3.3.3). При зміні мовної моделі або системного запита характеристики простору вкладень можуть змінитися, що потребуватиме повторної валідації алгоритму на новій конфігурації.

Гіперпараметри гібридного ранжування. Коефіцієнти тегового бонусу (2.5) та поріг лінійного калібрування відсотка (2.7) обґрунтовано з огляду на статистичні властивості поточного каталогу. Поріг 0.5 узгоджено з виміряним шумовим рівнем семантичної подібності випадкових пар (≈0.59), а максимальний бонус тегу 0.35 відповідає ширині корисного діапазону [0.6, 1.0]. При значному розширенні каталогу розподіл косинусних значень випадкових пар може зміститися, що змінить як шумовий поріг, так і ширину корисного діапазону. Тому систематична оптимізація цих гіперпараметрів через пошук по сітці параметрів на промисловому датасеті є природним напрямком подальшої роботи.

Виявлені обмеження не зменшують методологічної цінності розробленої системи – вони визначають межі коректної інтерпретації отриманих результатів та формують напрямки подальшого розвитку.

**Висновки до розділу 3**

У третьому розділі реалізовано та експериментально перевірено інтелектуальну вебсистему рекомендацій мультимедійного контенту. Спроєктовано трирівневу архітектуру з клієнтською частиною на Astro та React, серверною частиною на FastAPI та базою даних PostgreSQL з підтримкою векторного пошуку через pgvector. На основі цієї архітектури побудовано вісім функціональних модулів системи, тобто авторизацію, перегляд каталогів фільмів, музики та відеоігор, детальні сторінки елементів, однодоменні рекомендації, крос-доменні рекомендації з природномовними поясненнями, інтерактивну навігацію бібліотекою користувача, соціальне порівняння смаків між користувачами та інтелектуальний пошук природною мовою.

Експериментальна оцінка підтвердила ефективність обраного методологічного підходу. Однодоменна рекомендація на основі TF-IDF показала високу точність та стабільність у всіх трьох доменах, суттєво перевищивши базові методи популярності та випадкового вибору. Крос-доменна рекомендація на основі LLM-нормалізації описів та векторного пошуку довела свою здатність знаходити змістовні зв’язки між принципово різними типами медіа – як очевидні франшизні відповідності, так і концептуальні паралелі за настроєм, темпом та тематикою. Якісний аналіз конкретних прикладів продемонстрував, що система коректно диференціює рівень впевненості рекомендацій та чесно сигналізує користувачу про слабкі зв’язки замість штучного завищення оцінок схожості. Виявлені обмеження стосуються переважно обсягу каталогу та методології оцінки, що формує природні напрямки подальшого розвитку системи. Отримані результати підтверджують, що розроблена система відповідає поставленим у п. 1.3 завданням та забезпечує цілісне рішення для крос-доменних рекомендацій мультимедійного контенту.

# 4 ПРАКТИЧНА ВЕРИФІКАЦІЯ РОЗРОБЛЕНОЇ СИСТЕМИ

У попередніх розділах роботи було проведено аналіз предметної області, обґрунтовано вибір методів рекомендацій, описано архітектуру системи та її основні модулі, а також представлено результати кількісного оцінювання якості алгоритмів. На відміну від п. 3.2, де систему розглянуто з технічної точки зору, цей розділ зосереджений на досвіді кінцевого споживача. Наведено керівництво з типовими сценаріями використання, а також результати тестування на функціональному, інтеграційному та продуктивному рівнях.

## 4.1 Керівництво користувача

Даний підрозділ має забезпечити користувача зрозумілими та послідовними інструкціями, що дозволять йому всебічно використовувати увесь функціонал системи належним чином. Спочатку наведено системні вимоги та процес первинного налаштування, після цього представлено послідовний опис ключових сценаріїв використання. На завершення надано пояснення щодо коректної інтерпретації показників, які відображаються системою. Такий порядок дозволяє користувачу як вперше ознайомитись з можливостями застосунку, так і повертатися до окремих пунктів керівництва за потреби.

### 4.1.1 Системні вимоги та початок роботи з системою

Для роботи з системою користувачу не потрібно встановлювати додаткове програмне забезпечення. Достатньо сучасного веб-браузера (Google Chrome, Mozilla Firefox, Microsoft Edge або Safari актуальних версій), стабільного підключення до мережі Інтернет та облікового запису Google, який використовується для авторизації через сервіс Firebase Authentication. Інтерфейс системи адаптований до різних розмірів екрану та коректно відображається на моніторах комп’ютерів, планшетах та мобільних телефонах.

Коли користувач потрапляє на головну сторінку вебзастосунку його зустрічає літаючі картки різних типів медіа та кнопки входу у свій профіль. Натиснувши кнопку "Sign in with Google", користувач активує стандартне діалогове вікно Google, де він може надати згоду на передачу базової інформації свого профілю, такої як ім’я, електронна адреса та фото. Якщо автентифікація проходить успішно, система автоматично створює новий запис у власній базі даних (у таблиці users), за умови, що такого користувача ще не було зареєстровано. Немає потреби заповнювати будь-які додаткові форми для реєстрації.

Після першого входу користувача персоналізовані рекомендації будуть недоступні. Це пояснюється нестачею інформації про вподобання користувача, що є класичною проблемою "холодного старту", детальніше описаною в розділі 1.2. Користувачеві пропонується розпочати з ознайомлення з каталогом і вибору вподобань щонайменше в одній з трьох категорій: фільми, ігри або музика.

### 4.1.2 Основні сценарії використання

Перегляд каталогу та додавання вподобань. Навігаційне меню містить розділи каталогу "Movies", "Games" та "Tracks". У меню навігації є секції з матеріалами: "Movies", "Games" та "Tracks". Для кожного розділу представлена сітка карток, де міститься зображення, найменування, ключова інформація (рік випуску, категорії, оцінка) та значок серця у правому верхньому кутку картки при наведенні, натискаючи на серце, відповідний елемент потрапляє до персонального списку вподобань користувача, значок змінює вигляд з порожнього серця на заповнене. База даних фіксує новий запис у таблиці, яка має назву "likes". Повторне натискання на серце видалить вподобання. Каталоги надають можливість здійснювати пошук за назвою, семантикою або запитом до штучного інтелекту, а також фільтрувати результати за жанрами, використовуючи бічну панель. Для отримання рекомендацій, що точно відповідатимуть інтересам користувача, рекомендується вказати щонайменше п’ять вподобань у відповідному типу медіа.

Отримання персональних рекомендацій. Після додавання вподобань користувач переходить на сторінку "Discover", де відображаються три горизонтальні стрічки рекомендацій, окремо для фільмів, ігор та музики. Кожна стрічка містить десять елементів, упорядкованих за спаданням оцінки релевантності, обчисленої методом TF-IDF з косинусною мірою подібності (2.1–2.3). У випадку, коли користувач ще не вказав жодних вподобань у конкретному домені, замість стрічки рекомендацій буде відображено відповідне повідомлення. Це повідомлення міститиме пропозицію додати вподобання у каталозі цього домену.

Дослідження крос-доменних зв’язків. При натисканні на будь-який елемент у списку рекомендацій або каталогу відкриває детальну сторінку елемента. У нижній частині сторінки розташовано блок "Cross-domain echoes", у якому представлені рекомендації з інших двох доменів, семантично пов’язані з поточним елементом. Поряд із кожною крос-доменною рекомендацією відображається відсоток подібності (2.7) та коротке пояснення, яке згенероване мовною моделлю Claude Haiku та створюється лише за явним запитом користувача, при натисканні на елемент, що оптимізує використання токенів та зменшує навантаження на зовнішній API.

AI-пошук природною мовою. На головній сторінці присутнє поле "Smart search", у яке користувач може ввести запит звичайною людською мовою, наприклад "темне фентезі з моральним вибором" або "спокійна музика для роботи". Система передає запит до Claude Haiku, який інтерпретує його семантично та формує перелік релевантних елементів із каталогу всіх трьох доменів. На відміну від класичного пошуку за назвою, AI-пошук розуміє жанрові, емоційні та тематичні характеристики запиту.

Соціальні функції. Розділ "Friends" надає можливість пошуку інших користувачів за іменем або поштою та надіслати запит на додавання у друзі. Переглядати профіль можна як і користувача який є в друзях, так і будь-якого користувача. На сторінці, де порівнюються два профілі, відображається відсоток

сумісності смаків, список спільно вподобаних елементів, короткий AI-опис спільних рис і відмінностей користувачів, яка згенерована мовною моделлю на основі обох профілів, а також відображено спільні емоційні, тегові та жанрові вподобання.

### 4.1.3 Інтерпретація показників системи

Усі відсотки подібності, що відображаються у крос-доменних рекомендаціях та порівнянні з друзями, проходять через процедуру калібрування (2.7), тому показані значення відповідають реальному рівню збігу. Згідно з налаштованими порогами системи, значення 75% і вище свідчать про сильний змістовний зв’язок між елементами, діапазон 55–74% про помірний зв’язок, нижче 55% про слабкий, але все одно виявлений зв’язок. Така градація узгоджена із трьома гілками запиту для генерації пояснень, що дозволяє AI-коментарю чесно відображати ступінь близькості елементів, від впевненого твердження про спільну рису до обережного формулювання "share X, but…".

У випадках, коли інформація щодо певного компонента є неповною, наприклад, недостатньо вподобань у певному домені, відповідний блок не відображається або виводиться пояснювальне повідомлення, яке допомагає користувачу одразу зорієнтуватися, що треба зробити. Усі помилки взаємодії з зовнішніми сервісами перехоплюються на сервері та повертають користувачу зрозумілий текст без розкриття технічних деталей реалізації.

## 4.2 Тестування системи

Тестування програмної реалізації проводилось у кілька етапів. Перший етап включав функціональне тестування, що перевіряє відповідність реалізованих можливостей вимогам, сформульованим у розділі 1. Другий етап це інтеграційне тестування, що підтверджує коректну взаємодію між клієнтською частиною, серверним API, базою даних PostgreSQL та зовнішніми сервісами. Третій етап це тестування продуктивності, що оцінює час відповіді ключових компонентів.

### 4.2.1 Функціональне тестування

Перевірка працездатності виконувалось через проходження розроблених сценаріїв тестування. Вони охоплюють основні шляхи взаємодії користувача із системою. Перелік охоплює реєстрацію та автентифікацію, роботу з каталогом, додавання та видалення вподобань, отримання однодоменних і крос-доменних рекомендацій, генерацію AI-пояснень, AI-пошук, а також соціальні функції. Окремими тест-кейсами перевірено обробку граничних ситуацій, зокрема, відсутність вподобань, щоб переконатися у відсутності аварійних завершень роботи. Результати наведено у таблиці 4.1.

Таблиця 4.1 – Перелік функціональних тест-кейсів

| № | Опис тесту | Очікуваний результат | Статус |
|---|---|---|---|
| TC-01 | Реєстрація нового користувача через Google OAuth | Створено запис у таблиці users, відкрито головну сторінку | Pass |
| TC-02 | Повторний вхід існуючого користувача | Завантажено профіль, відображено персональні дані | Pass |
| TC-03 | Перегляд каталогу фільмів | Відображено сітку з 100 фільмів та можливістю фільтрації | Pass |
| TC-04 | Додавання вподобання | Створено запис у таблиці likes, кнопка змінює стан | Pass |
| TC-05 | Видалення вподобання | Запис видалено з бази даних, кнопка повертається у вихідний стан | Pass |
| TC-06 | Отримання однодоменних рекомендацій | Виведено 10 елементів, упорядкованих за релевантністю | Pass |
| TC-07 | Крос-доменні рекомендації для фільму | Виведено пов’язані ігри та треки з відсотком подібності | Pass |
| TC-08 | Генерація AI-пояснення для крос-доменної пари | Виведено текстове пояснення, кешовано у базу даних | Pass |
| TC-09 | AI-пошук природною мовою | Виведено релевантні результати з трьох доменів | Pass |
| TC-10 | Надсилання запиту в друзі | Створено запис типу pending, отримувач бачить повідомлення | Pass |
| TC-11 | Прийняття запиту в друзі та перегляд порівняння | Виведено відсоток сумісності та AI-аналітику | Pass |
| TC-12 | Обробка відсутності вподобань | Виведено інформаційне повідомлення без помилок | Pass |

Усі тест-кейси пройдено успішно. Це підтверджує, що функціональні вимоги до системи, сформульовані у підрозділі 1.3 повністю реалізовані.

### 4.2.2 Інтеграційне тестування

Інтеграційне тестування перевіряє взаємодію між компонентами системи, оскільки розподілена архітектура має кілька потенційно вразливих ланок зв'язку. Окремо було перевірено стійкість системи до типових помилок, наприклад, недоступність зовнішнього API, перевищення ліміту запитів, відсутність мережі на стороні клієнта. У всіх випадках система коректно повідомляє користувача про проблему без розкриття внутрішніх деталей реалізації та без аварійного завершення роботи. Результати наведено у таблиці 4.2.

Таблиця 4.2 – Результати інтеграційного тестування

| **Точка інтеграції** | **Перевірка** | **Статус** |
|---|---|---|
| Frontend ↔ FastAPI | Усі REST-запити повертають коректний JSON та статус-коди | Pass |
| FastAPI ↔ PostgreSQL | CRUD-операції з таблицями users, likes, friendships виконуються коректно | Pass |
| FastAPI ↔ pgvector | Пошук найближчих сусідів за косинусною мірою повертає очікувані результати | Pass |
| FastAPI ↔ Firebase Auth | Перевірка ID-токенів проходить успішно, неавторизовані запити відхиляються | Pass |
| FastAPI ↔ Claude API | LM-виклики повертають валідні відповіді, помилки коректно перехоплюються | Pass |
| FastAPI ↔ TMDB, RAWG, Last.fm | Дані для каталогу завантажуються та оновлюються без втрат | Pass |

Усі шість ключових точок інтеграції пройшли перевірку без зауважень, що підтверджує коректність обміну даними між компонентами розподіленої архітектури. Це дозволяє стверджувати, що збій або затримка в будь-якій з ланок не призводить до некоректного стану системи в цілому.

### 4.2.3 Тестування продуктивності

Тестування продуктивності виконувалось шляхом замірів часу відповіді ключових кінцевих точок сервера за умов локального розгортання клієнта та з'єднання з хмарною базою даних Neon (PostgreSQL 17 з розширенням pgvector). Заміри проводилися утилітою curl, кожне значення є середнім арифметичним за десятьма послідовними викликами. Для кінцевих точок, що мають однотипну реалізацію у різних доменах (фільми, ігри, треки), у таблиці наведено усереднений показник для всіх трьох доменів. Результати наведено у таблиці 4.3.

Таблиця 4.3 – Середній час відповіді кінцевих точок API системи

| № | Кінцева точка | Опис | Середній час, мс |
|---|---|---|---|
| 1 | GET/{domain} | Завантаження каталогу | 299 |
| 2 | GET/{domain}/{id} | Деталі окремого елемента | 572 |
| 3 | GET/tags/overview | Перелік тегів усіх трьох доменів | 561 |
| 4 | GET/users/{uid} | Профіль користувача | 571 |
| 5 | GET/users/search | Пошук користувача за іменем | 738 |
| 6 | GET/likes/{uid} | Усі вподобання користувача | 573 |
| 7 | GET/likes/{uid}/{type}/{id} | Наявність конкретного вподобання | 650 |
| 8 | GET/friend/{uid}/… | Соціальні запити | 576 |
| 9 | GET/recommendations/{uid}/{domain} | TF-IDF рекомендації | 253 |
| 10 | GET/cross/{type}/{id} | Крос-доменні рекомендації | 854 |
| 11 | POST/cross/explain | AI-пояснення крос-доменної пари | 1069 |
| 12 | POST/search | Звичайний / AI-пошук природною мовою | 1281/2584 |
| 13 | GET/compare{u1}/{u2}/insight | Порівняння двох профілів з AI-аналітикою | 4302 |

Виходячи з отриманих даних, визначаємо, що операції без звернення до зовнішніх AI-сервісів виконуються у межах 250–740 мс, що є прийнятним значенням для вебзастосунку та забезпечує відчуття миттєвого відгуку інтерфейсу.

Найшвидшими є виклики однодоменних рекомендацій (≈250 мс), застосування кешування SQL-запитів зменшує час, тому що каталоги завантажуються лише один раз і повторно використовуються для обчислень. Крос-доменні рекомендації виконуються повільніше (≈850 мс), оскільки додатковий пошук найближчих сусідів у векторному просторі pgvector. Операції, що використовують мовну модель Claude Haiku (AI-пошук, AI-пояснення, AI-аналітика порівняння профілів), мають помітно вищу затримку (1.0–4.3 с), адже такі запити залежать від часу генерації відповіді на стороні зовнішнього API.

### 4.2.4 Навантажувальне тестування

Навантажувальне тестування проводилось з метою оцінити поведінку системи при одночасній роботі декількох користувачів та визначити пропускну здатність ключових кінцевих точок. Тестування виконувалось з використанням бібліотеки aiohttp, що дозволяє виконувати асинхронні HTTP-запити з контрольованим рівнем паралелізму. Для кожного рівня одночасних з'єднань виконувалось 200 запитів, після чого обчислювалися пропускна здатність (RPS – requests per second) та середній час відповіді. Результати тестування декількох запитів наведено у таблиці 4.4.

Таблиця 4.4 – Результати навантажувального тестування

| **Concurrency** | **GET /movies** | | **GET /recommendations** | | **GET /cross/movie** | |
|---|---|---|---|---|---|---|
| | RPS | mean, mc | RPS | mean, mc | RPS | mean, mc |
| 1 | 7.9 | 126 | 230 | 4 | 1.5 | 657 |
| 5 | 77 | 64 | 747 | 5 | 7.5 | 660 |
| 10 | 74 | 132 | 706 | 10 | 14.5 | 672 |
| 25 | 74 | 313 | 727 | 24 | 34 | 694 |
| 50 | 74 | 602 | 719 | 47 | 49 | 906 |
| 100 | 76 | 991 | 652 | 108 | 49.5 | 1584 |

На всіх рівнях навантаження жоден запит не завершився помилкою. Найвищу пропускну здатність забезпечують однодоменні рекомендації (до 747 RPS), після

першого виклику дані каталогу залишаються в оперативній пам’яті, тож повторні обчислення TF-IDF не звертаються до бази даних. Завантаження каталогу досягає плато ≈ 75 RPS при C ≥ 5 (обмеження пулу з’єднань бази даних), а крос-доменні рекомендації ≈ 50 RPS через комбінацію векторного пошуку та зчитування кешованих AI-пояснень. При C = 100 формується черга запитів, але всі вони обслуговуються без втрат, що є достатнім показником для очікуваного навантаження.

### 4.3 Безпека системи

Архітектура безпеки розробленої системи охоплює декілька рівнів. Вона регламентує автентифікацію, валідує вхідні дані та фільтрує джерела запитів. Нижче розглянуто технічну реалізацію цих механізмів та результати їх випробування реальними атаками.

Ідентифікацію користувачів виконує Firebase Authentication через провайдер Google OAuth 2.0. З підписаного ідентифікатор токена JWT (JSON Web Token) сервер витягує унікальний firebase_uid. Далі система використовує його як первинний ключ у базі даних. Застосунок не зберігає і не обробляє користувацьких паролів. Таке архітектурне рішення усуває масив вразливостей, характерних для самописних систем керування обліковими записами.

Взаємодія з базою даних спирається на параметризовані запити: СУБД отримує значення ізольовано від SQL-команди, що блокує SQL-ін’єкції. Клієнтський React автоматично екранує рядки перед вставкою в об’єктну модель документа, унеможливлюючи XSS-атаки. Сервер контролює трафік через політики CORS, відкидаючи невідомі домени на стадії OPTIONS. Чутливі дані винесено у файл .env поза системою контролю версій. API додатково захищено моделями Pydantic. Вони перевіряють структуру запиту та типи змінних. Некоректний JSON відхиляється (HTTP 422), тому погані дані не доходять до бізнес-логіки.

Дієздатність цих механізмів перевірено на практиці. Система пройшла серію live-тестів, що імітують основні класи вразливостей з актуального переліку

OWASP (Open Web Application Security Project) Top 10. Результати зведено у таблицю 4.5.

Таблиця 4.5 – Результати тестування безпеки

| № | Вектор атаки | Запит | Фактичний результат |
|---|---|---|---|
| S-01 | SQL-ін’єкція у параметр пошуку | GET /users/search?q=' OR '1'='1 | HTTP 200, порожній список, бази даних не зачеплено |
| S-02 | SQL-ін’єкція у параметр пошуку | GET /likes/test_rpg_fan';DROP TABLE users;-- | HTTP 200, таблиця users цілісна |
| S-03 | Відсутнє обов’язкове поле | POST /search без поля q | HTTP 422, «Field required» |
| S-04 | Неправильний тип поля | POST /cross/explain з score: "abc" | HTTP 422, «Input should be a valid integer» |
| S-05 | CORS з недозволеного джерела | OPTIONS /movies з Origin: http://evil.com | HTTP 400, заголовок Access-Control-Allow-Origin відсутній |
| S-06 | CORS з дозволеного джерела | OPTIONS /movies з Origin: http://localhost:4321 | HTTP 200, видано Access-Control-Allow-Origin |
| S-07 | Запит до неіснуючого користувача | GET /likes/this_user_does_not_exist_xyz | HTTP 200, порожні списки без розкриття деталей |

## Висновки до розділу 4

У розділі надано керівництво для користувача, яка розкриває ключові взаємодії з системою, розглянувши основні сценаріїв взаємодії з системою – від реєстрації за допомогою Google OAuth до використання соціальних функцій порівняння смаків. Проведене тестування підтвердило коректність роботи реалізації, усі дванадцять функціональних тест-кейсів і шість інтеграційних точок успішно пройдено, час відповіді тринадцяти основних запитів перебуває у межах, прийнятних для вебзастосунку, а навантажувальне тестування показало стабільну роботу при паралелізмі до ста одночасних з’єднань без втрат запитів. Сім перевірених векторів атаки на безпеку від SQL-ін’єкцій до некоректних CORS-запитів оброблено системою згідно з очікуваннями. У сукупності з результатами алгоритмічного оцінювання, наведеними в третьому розділі, свідчать про те, що розроблена система є функціонально повною, продуктивною та готовою до практичного використання й подальшого розвитку.

# ВИСНОВКИ

У кваліфікаційній роботі розроблено інтелектуальну вебсистему персоналізованих рекомендацій мультимедійного контенту, що об’єднує фільми, музичні треки та відеоігри в єдину екосистему на основі накопиченого профілю користувача. Усі завдання, визначені в постановці задачі, виконано в повному обсязі.

У ході дослідження існуючих методів побудови рекомендаційних систем проаналізовано підходи контентно-орієнтованої фільтрації, методи семантичного аналізу на основі векторних представлень, а також напрямок крос-доменних рекомендацій. Встановлено, що провідні платформи – Netflix, Spotify, Steam – реалізують рекомендації виключно у межах одного домену, а найближчий загальнодоступний мультимедійний агрегатор Tastedive не зберігає накопиченого профілю та позбавлений соціального компоненту. Виявлена прогалина підтвердила актуальність і обґрунтованість обраного напрямку.

Сформовано підхід до крос-доменного зіставлення контенту на основі конвеєра семантичної нормалізації (vibe-pipeline), тобто велика мовна модель генерує для кожного елемента каталогу нейтральний текстовий опис у спільному словнику настрою та атмосфери без прив’язки до типу медіа, який перетворюється на 384-вимірне векторне вкладення та зберігається через розширення pgvector. Гібридне ранжування поєднує семантичну косинусну подібність з тег-бонусом на основі спільних жанрів і тегів, що дозволяє знаходити тематично близький контент між різними типами медіа навіть за відсутності спільних метаданих.

Архітектуру системи спроєктовано у вигляді трирівневого вебзастосунку. Підсистема збору даних інтегрує зовнішні API – TMDB, RAWG та Last.fm і формує каталог. Блок профілювання будує IDF-зважений профіль користувача на основі вподобань і оновлює його при додаванні нових вподобань. Ядро рекомендацій реалізує однодоменну фільтрацію на основі TF-IDF, крос-доменну рекомендацію

через конвеєр семантичної нормалізації та порівняння профілів між двома користувачами.

Інтеграцію великої мовної моделі реалізовано у кількох точках. Модель генерує семантичні описи елементів при наповненні каталогу, формує природномовні пояснення крос-доменних рекомендацій із трьома стратегіями підказок залежно від рівня схожості, а також робить аналітичний опис при порівняння профілів користувачів. Для оптимізації витрат реалізовано кешування відповідей моделі у базі даних. Функцію соціального порівняння побудовано на розрахунку косинусної подібності між зваженими профілями двох користувачів з виділенням характерних вподобань кожного та лінійним перетворенням значень подібності у шкалу відсотку.

Програмний продукт реалізовано за допомогою FastAPI і PostgreSQL 17 з pgvector та Astro 6 і React 19 з авторизацією через Firebase Auth. Проведено функціональне тестування 12 тест-кейсів та інтеграційне тестування шести компонентів системи – усі пройдено успішно. Навантажувальне тестування при 100 одночасних з’єднаннях не виявило жодної помилки. Якість рекомендацій оцінено за метриками Precision@K, Recall@K та MRR@K на 15 синтетичних персонах із п’ятьма випадковими seed. Однодоменний TF-IDF алгоритм досяг Precision@10 $0.859 \pm 0.017$, що перевищує алгоритм популярності у 6.78 рази. Гібридний крос-доменний алгоритм підвищив тегову релевантність у 2.62 рази порівняно з випадковим базисом при збереженні семантичної якості, що підтверджено аналізом вкладу компонент алгоритму.

Отримані результати свідчать про ефективність поєднання TF-IDF, семантичних вкладень та великих мовних моделей для побудови мультимедійної рекомендаційної системи. Розроблена система може використовуватися як самостійна платформа або як основа для подальших досліджень у напрямку крос-доменної персоналізації з розширенням каталогу та залученням реальних даних про поведінку користувачів.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Nicoomanesh A. Evolution of Recommendation Algorithms, Part I: Fundamentals, History Overview, Core and Classical. *Medium* : вебсайт. 2024. URL: https://medium.com/@anicomanesh/evolution-of-recommendation-algorithms-part-i-fundamentals-and-classical-recommendation-bb1c0bce78a9 (дата звернення: 01.02.2026).

2. Entertainment Media Market Size, Share & Trends Report – 2035. *SNS Insider* : вебсайт. 2026. URL: https://www.snsinsider.com/reports/entertainment-media-market-9760 (дата звернення: 03.02.2026).

3. IMDb Top Data Contributors for 2025 (and Happy New Year 2026 from IMDb). *IMDb Community* : вебсайт. 2025. URL: https://community-imdb.sprinklr.com/conversations/data-issues-policy-discussions/imdb-top-data-contributors-for-2025-and-happy-new-year-2026-from-imdb/695629bcd907dc059b7ba512 (дата звернення: 05.02.2026).

4. Spotify Statistics. User Demographics, Daily Usage & Revenue. *Our Own Brand* : вебсайт. 2026. URL: https://ourownbrand.co/spotify-statistics/ (дата звернення: 07.02.2026).

5. Steam Game Release Summary by Year. *SteamDB* : вебсайт. 2026. URL: https://steamdb.info/stats/releases/ (дата звернення: 09.02.2026).

6. Spotify Statistics. *SoundCampaign* : вебсайт. 2026. URL: https://soundcamps.com/blog/spotify-statistics/ (дата звернення: 11.02.2026).

7. Spotify Usage Statistics. *Spaceloud* : вебсайт. 2026. URL: https://www.spaceloud.com/blog/best-playlist-promotion-services (дата звернення: 11.02.2026).

8. Media & Entertainment Industry Statistics. Growth Facts. *SQ Magazine* : вебсайт. 2026. URL: https://sqmagazine.co.uk/media-and-entertainment-industry-statistics/ (дата звернення: 12.02.2026).

9. Digital Media Trends. *Deloitte Insights* : вебсайт. 2026. URL: https://www.deloitte.com/us/en/insights/industry/technology/digital-media-trends-consumption-habits-survey.html (дата звернення: 13.02.2026).

10. Ricci F., Rokach L., Shapira B. Recommender Systems Handbook. 3rd ed. New York : Springer US. 2022. 1060 p. DOI: 10.1007/978-1-0716-2197-4.

11. A Brief History of Recommender Systems / Z. Dong et al. *ArXiv*. 2022. DOI: 10.48550/arXiv.2209.01860.

12. Konstan J., Terveen L. Human-Centered Recommender Systems: Origins, Advances, Challenges, and Opportunities. *AI Magazine*. 2021. Vol. 42, № 3. P. 31–42. DOI: 10.1609/aimag.v42i3.18142.

13. Rich E. User Modeling via Stereotypes. *Cognitive Science*. 1979. Vol. 3, № 4. P. 329–354. URL: https://www.cs.utexas.edu/~ear/CogSci.pdf (дата звернення: 23.02.2026).

14. Aggarwal C. C. Recommender Systems: The Textbook. Cham : Springer International Publishing, 2016. DOI: 10.1007/978-3-319-29659-3.

15. On Generative Agents in Recommendation / J. Zhang et al. *ArXiv*. 2023. DOI: 10.48550/arXiv.2310.10108

16. Hsiao K.-J., Feng Y., Lamkhede S. Foundation Model for Personalized Recommendation. Netflix Technology Blog : вебсайт. 2025. URL: https://netflixtechblog.com/foundation-model-for-personalized-recommendation-1a0bd8e02d39 (дата звернення: 01.03.2026).

17. Burke R. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*. 2002. Vol. 12, № 4. P. 331–370. DOI: 10.1023/A:1021240730564.

18. Discover Weekly: How Spotify is Changing the Way We Consume Music. *Harvard Business School Technology and Operations Management* : вебсайт. 2018. URL: https://d3.harvard.edu/platform-rctom/submission/discover-weekly-how-spotify-is-changing-the-way-we-consume-music/ (дата звернення: 04.03.2026).

19. Semantic Search For Recommendation Systems. *Meegle* : вебсайт. 2026. URL: https://www.meegle.com/en_us/topics/semantic-search/semantic-search-for-recommendation-systems (дата звернення: 07.03.2026).

20. CD-LLMCARS: Cross Domain Fine-Tuned Large Language Model for Context-Aware Recommendation System / Cheema A. A. et al. *IEEE Open Journal of the Computer Society*. 2025. Vol. 1. URL: https://www.computer.org/csdl/journal/oj/2025/01/10771726/22fgyqh6mas (дата звернення: 09.03.2026).

21. Balancing Global and Local Interests in Cross-Domain Recommendation Systems / Zhao Y. et al. *Multimedia Systems*. 2025. DOI: 10.1007/s00530-025-01969-1.

22. Koren Y., Bell R., Volinsky C. Matrix Factorization Techniques for Recommender Systems. *Computer (IEEE)*. 2009. Vol. 42, № 8. P. 30–37. DOI: 10.1109/MC.2009.263.

23. Shahbazi Z., Jalali R., Shahbazi Z. Enhancing Recommendation Systems with Real-Time Adaptive Learning and Multi-Domain Knowledge Graphs. *Big Data and Cognitive Computing*. 2025. Vol. 9, № 5. Art. 124. DOI: 10.3390/bdcc9050124.

24. Herimanto H., Samosir K., Ginting F. A Comparative Analysis of Content-Based Filtering and TF-IDF Approaches for Enhancing Sports Recommendation Systems. *Innovation in Research of Informatics.* 2024. Vol. 6, № 2. P. 90–97. URL: https://jurnal.unsil.ac.id/index.php/innovatics/article/view/12404 (дата звернення: 12.03.2026).

25. Cross-Modal Attention Network with Dual Graph Learning in Multimodal Recommendation / Dai J. et al. *ArXiv.* 2026. DOI: 10.48550/arXiv.2601.11151.

26. A Survey on Cross-domain Recommendation: Taxonomies, Methods, and Future Directions / Zang T. et al. *ACM Transactions on Information Systems*. 2022. Vol. 41, № 2. P. 1–39. DOI: 10.1145/3548455.

27. Reimers N., Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *Proceedings of the 2019 Conference on Empirical Methods in*

*Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, 2019. P. 3982–3992. DOI: 10.18653/v1/D19-1410.

28. A Survey on Large Language Models for Recommendation / Wu L. et al. *World Wide Web*. 2024. Vol. 27. Art. 60. DOI: 10.1007/s11280-024-01291-2.

29. Leskovec J., Rajaraman A., Ullman J. D. Mining of Massive Datasets. 3rd ed. *Cambridge* : Cambridge University Press, 2020. 565 p.

30. Array Programming with NumPy / Harris C. R. et al. *Nature*. 2020. Vol. 585. P. 357–362. DOI: 10.1038/s41586-020-2649-2.

31. Ramírez S. FastAPI: Modern, Fast Web Framework for Building APIs with Python : вебсайт. 2024. URL: https://fastapi.tiangolo.com (дата звернення: 16.03.2026).

32. PostgreSQL 17 Documentation. PostgreSQL Global Development Group : вебсайт. 2024. URL: https://www.postgresql.org/docs/17/ (дата звернення: 19.03.2026).

33. React: A JavaScript Library for Building User Interfaces. Meta Open Source : вебсайт. 2024. URL: https://react.dev (дата звернення: 21.03.2026).

34. Astro Documentation. Astro Technology Inc : вебсайт. 2024. URL: https://docs.astro.build (дата звернення: 23.03.2026).

35. API Documentation. TMDB (The Movie Database) : вебсайт. 2025. URL: https://developer.themoviedb.org (дата звернення: 25.03.2026).

36. API Documentation. RAWG Video Games Database : вебсайт. 2025. URL: https://rawg.io/apidocs (дата звернення: 27.03.2026).

37. Scikit-learn: Machine Learning in Python / Pedregosa F. et al. *Journal of Machine Learning Research*. 2011. Vol. 12. P. 2825–2830.

# ДОДАТОК А

## Лістинг коду збору вхідних даних із зовнішніх API

### А.1 Збір фільмів із TMDB API

```
BASE    = "https://api.themoviedb.org/3"
IMG     = "https://image.tmdb.org/t/p/w500"
HEADERS = {"Authorization": f"Bearer {TMDB_TOKEN}"}

def get(path, **params):
    r = httpx.get(f"{BASE}{path}", headers=HEADERS,
                  params={"language": "en-US", **params})
    return r.json()

def fetch_movie(tmdb_id):
    d = get(f"/movie/{tmdb_id}", append_to_response="credits,keywords")
    crew     = d.get("credits", {}).get("crew", [])
    director = next((p["name"] for p in crew if p["job"] == "Director"), "")
    return {
        "tmdb_id":      d["id"],
        "title":        d["title"],
        "genres":       [g["name"] for g in d.get("genres", [])],
        "keywords":     [k["name"] for k in d["keywords"]["keywords"][:15]],
        "overview":     d.get("overview", ""),
        "tagline":      d.get("tagline", ""),
        "release_year": int(d.get("release_date", "0")[:4] or 0),
        "rating":       round(float(d.get("vote_average") or 0), 1),
        "director":     director,
        "actors":       [p["name"] for p in d["credits"]["cast"][:6]],
        "poster_url":   f"{IMG}{d['poster_path']}",
    }
```

### А.2 Збір відеоігор із RAWG API

```
BASE = "https://api.rawg.io/api"
def rget(path, **params):

    r = httpx.get(f"{BASE}{path}", params={"key": RAWG_KEY, **params})
    return r.json()

def search_game(name):
    data = rget("/games", search=name, search_precise=True, page_size=1)
    results = data.get("results", [])
    return results[0]["id"] if results else None

def fetch_game(rawg_id):
    d = rget(f"/games/{rawg_id}")
    return {
        "rawg_id":      d["id"],
        "title":        d["name"],
```

```
        "genres":       [g["name"] for g in d.get("genres", [])],
        "tags":         [t["name"] for t in d.get("tags", [])
                         if t.get("language") == "eng"],
        "description":  strip_html(d.get("description", "")),
        "release_year": int((d.get("released") or "0")[:4] or 0),
        "rating":       round(float(d.get("rating") or 0), 2),
        "metacritic":   d.get("metacritic") or 0,
        "platforms":    [p["platform"]["name"] for p in d.get("platforms") or []],
        "developers":   [x["name"] for x in d.get("developers") or []],
        "poster_url":   d.get("background_image") or "",
    }
```

## А.3 Збір музичних треків із Last.FM API

```
BASE = "https://ws.audioscrobbler.com/2.0"

def lfm(method, **params):
    r = httpx.get(BASE, params={"method": method, "api_key": LASTFM_KEY,
                                "format": "json", **params})
    return r.json()

def fetch_track(artist, title):
    t = lfm("track.getInfo", artist=artist, track=title)["track"]
    image = next((img["#text"] for img in reversed(t["album"]["image"])
                  if img["#text"]), "")
    return {
        "name":      t["name"],
        "artist":    t["artist"]["name"],
        "album":     t.get("album", {}).get("title", ""),
        "tags":      [tag["name"] for tag in t["toptags"]["tag"]],
        "summary":   strip_html(t.get("wiki", {}).get("summary", "")),
        "listeners": int(t.get("listeners") or 0),
        "playcount": int(t.get("playcount") or 0),
        "image_url": image,
        "url":       t.get("url", ""),
    }
```

# ДОДАТОК Б

## Лістинг коду однодоменного алгоритму TF-IDF з косинусною подібністю

### Б.1 Обчислення IDF та формування словника тегів

```
def build_idf(items, tag_fields, min_df=1):
    doc_freq = {}
    for item in items:
        seen = set()
        for field in tag_fields:
            for tag in (item.get(field) or []):
                key = tag.lower()
                if key not in seen:
                    doc_freq[key] = doc_freq.get(key, 0) + 1
                    seen.add(key)

    vocab = {key: i for i, (key, freq) in enumerate(doc_freq.items())
             if freq >= min_df}

    N = len(items)
    idf = {tag: math.log(N / (freq + 1)) for tag, freq in doc_freq.items()}
    return vocab, idf
```

### Б.2 Побудова вектора користувача

```
def build_user_vector(liked_items, tag_fields, vocab, idf):
    u = np.zeros(len(vocab))
    for item in liked_items:
        for field in tag_fields:
            for tag in (item.get(field) or []):
                key = tag.lower()
                if key in vocab:
                    u[vocab[key]] += idf[key]
    return u
```

### Б.3 Побудова матриці елементів каталогу

```
def build_item_matrix(candidates, tag_fields, vocab):
    M = np.zeros((len(candidates), len(vocab)))
    for i, item in enumerate(candidates):
        for field in tag_fields:
            for tag in (item.get(field) or []):
                key = tag.lower()
                if key in vocab:
                    M[i, vocab[key]] = 1.0
    return M
```

### Б.4 Косинусна подібність та формування топ-К рекомендацій

```
def recommend_top_k(user_vec, item_matrix, candidates, k=10):
    scores = cosine_similarity(user_vec.reshape(1, -1), item_matrix)[0]
    top_indices = np.argsort(scores)[::-1][:k]
    return [
        {"item": candidates[i], "score": round(float(scores[i]) * 100)}
        for i in top_indices if scores[i] > 0
```

```
    ]
```

## Б.5 Інтеграція функцій у єдиний алгоритм

```
def tf_idf_recommend(uid, items, liked_ids, tag_fields, min_df=1, k=10):
    vocab, idf = build_idf(items, tag_fields, min_df)
    liked = [it for it in items if it["id"] in liked_ids]
    candidates = [it for it in items if it["id"] not in liked_ids]

    user_vec = build_user_vector(liked, tag_fields, vocab, idf)
    item_matrix = build_item_matrix(candidates, tag_fields, vocab)
    return recommend_top_k(user_vec, item_matrix, candidates, k)
```

## Б.6 Отримання вподобань користувача з бази даних

```
def get_liked_ids(uid, item_type):
    conn = get_conn()
    cur = conn.cursor()
    cur.execute(
        "SELECT item_id FROM likes WHERE user_uid=%s AND item_type=%s",
        (uid, item_type)
    )
    return {row["item_id"] for row in cur.fetchall()}
```

## Б.7 Визначення тегів збігу для пояснення рекомендацій

```
def get_matched_tags(item, tag_fields, vocab, user_vec, idx_to_tag, top_n=3):
    item_tag_indices = {
        vocab[tag.lower()]
        for field in tag_fields
        for tag in (item.get(field) or [])
        if tag.lower() in vocab
    }
    matched = sorted(
        [i for i in item_tag_indices if user_vec[i] > 0],
        key=lambda i: -user_vec[i]
    )[:top_n]
    return [idx_to_tag[i].title() for i in matched]
```

## Б.8 Виклик алгоритму для фільмів через REST API

```
@router.get("/{uid}/movies")
def recommend_movies(uid):
    items = cached_query("movies",
                         "SELECT * FROM movies ORDER BY rating DESC")
    liked = get_liked_ids(uid, "movie")
    return tf_idf_recommend(uid, items, liked,
                            tag_fields=["genres", "keywords"], min_df=2)
```

# ДОДАТОК В

## Лістинг коду LLM-нормалізації та побудови векторних вкладень

### В.1 Системний запит для LLM-нормалізації описів

```
VIBE_SYSTEM = """You are a semantic normalizer for a multimedia recommendation system.

Given a music track, movie, or video game's metadata, write a SINGLE PARAGRAPH
(60-90 words) describing its EXPERIENTIAL essence:
- Mood and emotional tone (melancholic / exhilarating / brooding / playful / etc.)
- Energy and tempo (slow, medium, fast — intense or relaxed)
- Atmosphere (gritty, dreamlike, cinematic, nostalgic, urban, otherworldly)
- What moment or activity it suits (focus work, late-night drive, workout, etc.)
- Thematic abstractions (freedom, isolation, rebellion, longing, escapism, chaos)

CRITICAL RULES:
- Use DOMAIN-AGNOSTIC vocabulary. Avoid medium-specific terms like
  "gameplay", "soundtrack", "cinematography", "narrative arc", "level design".
- DO NOT mention the title, franchise, artist, or release year.
- DO NOT summarize plot or describe mechanics.
- Focus ONLY on what the item FEELS LIKE and WHEN it fits."""
```

### В.2 Формування короткого опису елемента для подачі у LLM

```
def build_summary(item, item_type):
    bits = []
    if item_type == "movie":
        bits.append(f"Title: {item['title']}.")
        bits.append(f"Genres: {', '.join(item.get('genres') or [])}.")
        bits.append(f"Keywords: {', '.join((item.get('keywords') or [])[:12])}.")
        if item.get("tagline"):
            bits.append(f'Tagline: "{item["tagline"]}".')
        bits.append(f"Plot: {(item.get('overview') or '')[:400]}")
    elif item_type == "game":
        bits.append(f"Title: {item['title']}.")
        tags = (item.get("genres") or []) + (item.get("tags") or [])
        bits.append(f"Tags: {', '.join(tags[:12])}.")
        bits.append(f"About: {(item.get('description') or '')[:400]}")
    elif item_type == "track":
        bits.append(f"Title: {item['name']}. Artist: {item['artist']}.")
        tags = (item.get("genres") or []) + (item.get("tags") or [])
        bits.append(f"Tags: {', '.join(tags[:12])}.")
        bits.append(f"About: {(item.get('summary') or '')[:400]}")
    return " ".join(bits)
```

### В.3 Генерація vibe-text через Claude Haiku

```
def generate_vibe(item, item_type):
    summary = build_summary(item, item_type)
    msg = llm.messages.create(
        model="claude-haiku-4-5-20251001",
        max_tokens=220,
        system=VIBE_SYSTEM,
        messages=[{"role": "user", "content": summary}],
    )
    return msg.content[0].text.strip()
```

## В.4 Побудова векторного вкладення через sentence-transformers

```
embed_model = SentenceTransformer("sentence-transformers/all-MiniLM-L6-v2")

def build_embedding(vibe_text):
    vec = embed_model.encode(vibe_text, normalize_embeddings=True)
    return "[" + ",".join(f"{x:.6f}" for x in vec) + "]"
```

## В.5 Загальний цикл побудови вкладень для всього каталогу

```
def process_catalog():
    for table, item_type in [("movies", "movie"),
                             ("games",  "game"),
                             ("tracks", "track")]:
        for item in fetch_all(table):
            vibe = generate_vibe(item, item_type)
            emb  = build_embedding(vibe)
            update_db(table, item["id"], vibe, emb)
```

## В.6 Збереження векторного вкладення у базі даних через pgvector

```
def save_to_db(table, item_id, vibe_text, embedding_vec):
    conn = get_conn()
    cur = conn.cursor()
    cur.execute(
        f"""
        UPDATE {table}
        SET vibe_text = %s,
            embedding = %s::vector
        WHERE id = %s
        """,
        (vibe_text, embedding_vec, item_id)
    )
    conn.commit()
```

## В.7 Пошук найближчих сусідів у векторному просторі через pgvector

```
def find_nearest_neighbors(anchor_vec, other_table, k=5):
    conn = get_conn()
    cur = conn.cursor()
    cur.execute(
        f"""
        SELECT *,
               1 - (embedding <=> %s::vector) AS semantic_score
        FROM {other_table}
        WHERE embedding IS NOT NULL
        ORDER BY embedding <=> %s::vector
        LIMIT %s
        """,
        (anchor_vec, anchor_vec, k)
    )
    return [dict(row) for row in cur.fetchall()]
```

# ДОДАТОК Г

## Лістинг коду крос-доменної рекомендації з гібридним ранжуванням

### Г.1 Параметри алгоритму гібридного ранжування

```
SHARED_TAG_BONUS = 0.07   # внесок одного спільного тегу
MAX_TAG_BONUS    = 0.35   # межа сумарного тегового бонусу
POOL_MULT        = 4      # розмір пулу = K * POOL_MULT
DISPLAY_BASELINE = 0.50   # поріг калібрування
DISPLAY_SCALE    = 2.0    # коефіцієнт лінійного перетворення

TABLE = {"track": "tracks", "movie": "movies", "game": "games"}
TAG_FIELDS = {
    "track": ("genres", "tags"),
    "movie": ("genres", "keywords"),
    "game":  ("genres", "tags"),
}
```

### Г.2 Збір тегів елемента для обчислення спільного перетину

```
def collect_tags(row, item_type):
    tags = set()
    for field in TAG_FIELDS[item_type]:
        for value in row.get(field) or []:
            if value:
                tags.add(str(value).lower())
    tags |= title_tokens(row.get("title") or row.get("name"))
    return tags
```

### Г.3 Калібрування відображуваного значення подібності

```
def calibrate(raw_score):
    scaled = (raw_score - DISPLAY_BASELINE) * DISPLAY_SCALE
    bounded = max(0.0, min(1.0, scaled))
    return round(bounded * 100)
```

### Г.4 Отримання пулу семантичних кандидатів через pgvector

```
def fetch_semantic_pool(cur, anchor_vec, other_table, k):
    cur.execute(f"""
        SELECT *,
               1 - (embedding <=> %s::vector) AS semantic_score
        FROM {other_table}
        WHERE embedding IS NOT NULL
        ORDER BY embedding <=> %s::vector
        LIMIT %s
    """, (anchor_vec, anchor_vec, k * POOL_MULT))
    return [dict(row) for row in cur.fetchall()]
```

### Г.5 Гібридне ранжування пулу кандидатів

```
def hybrid_rerank(anchor, anchor_type, pool, other_type):
    anchor_tags = collect_tags(anchor, anchor_type)
    scored = []
    for item in pool:
```

```
        sem = float(item["semantic_score"])
        item_tags = collect_tags(item, other_type)
        shared = anchor_tags & item_tags
        bonus = min(MAX_TAG_BONUS, len(shared) * SHARED_TAG_BONUS)
        hybrid = min(1.0, sem + bonus)
        scored.append({
            "item":           item,
            "hybrid_score":   hybrid,
            "semantic_score": sem,
            "shared_tags":    sorted(shared)[:3],
            "displayed":      calibrate(hybrid),
        })
    scored.sort(key=lambda x: -x["hybrid_score"])
    return scored
```

## Г.6 Загальний алгоритм крос-доменної рекомендації

```
def cross_domain_recommend(anchor_type, anchor_id, k=5):
    conn = get_conn()
    cur = conn.cursor()
    cur.execute(f"SELECT * FROM {TABLE[anchor_type]} WHERE id = %s", (anchor_id,))
    anchor = dict(cur.fetchone())
    anchor_vec = anchor["embedding"]

    result = {}
    for other_type in TABLE:
        if other_type == anchor_type:
            continue
        pool = fetch_semantic_pool(cur, anchor_vec, TABLE[other_type], k)
        ranked = hybrid_rerank(anchor, anchor_type, pool, other_type)
        result[other_type] = ranked[:k]
    return result
```

## Г.7 Виділення значущих лексем з назви елемента

```
TITLE_STOPWORDS = {
    "the", "and", "for", "with", "from", "into", "your", "this", "that",
    "edition", "remaster", "remastered", "definitive", "ultimate",
    "complete", "deluxe", "anniversary", "collection", "trilogy",
}

TOKEN_REGEX = re.compile(r"[\w\-]+", re.UNICODE)


def title_tokens(title):
    if not title:
        return set()
    return {
        token for token in TOKEN_REGEX.findall(title.lower())
        if len(token) >= 4 and token not in TITLE_STOPWORDS
    }
```

# ДОДАТОК Д

## Лістинг коду AI-функцій LLM

### Д.1 AI-пояснення крос-домених пар з адаптивним запитом

```
def adaptive_instruction(score):
    if score >= 75:
        return ('Strong match — lead confidently with the shared trait. '
                'Format: "both [share/feel/have] X".')
    if score >= 55:
        return ('Medium match — describe what they share, then briefly '
                'contrast. Format: "share X, but differ in Y".')
    return ('Weak match — start with "loosely connected". Mention the thin '
            'shared element, then explicitly contrast difference.')

def generate_explanation(anchor_vibe, rec_vibe, score):
    instruction = adaptive_instruction(score)
    prompt = (
        f"Pair [score {score}%]: {instruction}\n"
        f"  A: {anchor_vibe[:600]}\n"
        f"  B: {rec_vibe[:600]}\n\n"
        "Write ONE short sentence (max 18 words, no period) explaining "
        "the relationship between A and B in plain English. "
        "Avoid literary metaphors and domain-specific terms."
    )
    msg = llm.messages.create(
        model="claude-haiku-4-5-20251001",
        max_tokens=80,
        messages=[{"role": "user", "content": prompt}],
    )
    return msg.content[0].text.strip()
def cache_explanation(cur, anchor_type, anchor_id, rec_type, rec_id, text):
    cur.execute("""
        INSERT INTO cross_explanations
            (anchor_type, anchor_id, rec_type, rec_id, explanation)
        VALUES (%s, %s, %s, %s, %s)
        ON CONFLICT (anchor_type, anchor_id, rec_type, rec_id) DO NOTHING
    """, (anchor_type, anchor_id, rec_type, rec_id, text))
```

### Д.2 AI-аналітика порівняння смаків двох користувачів

```
def generate_compare_insight(name1, name2, tags1, tags2, common_titles,
                             distinctive_picks, compatibility):
    shared = ", ".join(common_titles) if common_titles else "nothing in common yet"
    picks  = ", ".join(distinctive_picks) if distinctive_picks else "no picks"
    prompt = (
        f"Two users on a multimedia recommendation platform.\n\n"
        f"{name1} — taste: {', '.join(tags1)}.\n"
        f"{name2} — taste: {', '.join(tags2)}.\n\n"
        f"Shared liked content: {shared}.\n"
        f"Taste compatibility: {compatibility}%.\n"
        f"Distinctive picks from {name2}: {picks}.\n\n"
        f"Write EXACTLY 3 short sentences addressed to {name1} in second "
        f"person (~80-100 words):\n"
        f"  1. What you and {name2} share.\n"
```

```
        f"  2. Where {name2}'s taste diverges from yours.\n"
        f"  3. Concrete suggestions to broaden your taste."
    )
    msg = llm.messages.create(
        model="claude-haiku-4-5-20251001",
        max_tokens=280,
        messages=[{"role": "user", "content": prompt}],
    )
    return msg.content[0].text.strip()
```

## Д.3 AI-пошук природною мовою

```
def ai_search(query, tracks, movies, games):
    catalog = {
        "t": [{"i": t["id"], "n": t["name"],  "g": t.get("genres", [])[:3]}
              for t in tracks[:40]],
        "m": [{"i": m["id"], "n": m["title"], "g": m.get("genres", [])[:3]}
              for m in movies[:40]],
        "g": [{"i": g["id"], "n": g["title"], "g": g.get("genres", [])[:3]}
              for g in games[:40]],
    }

    msg = llm.messages.create(
        model="claude-haiku-4-5-20251001",
        max_tokens=400,
        system="You are a recommendation assistant. Respond with raw JSON only.",
        messages=[{"role": "user", "content":
            f'User wants: "{query}"\n\n'
            f'From the catalog below pick up to 5 best-matching items per category.\n'
            f'Respond with this exact JSON:\n'
            f'{{"e":"one sentence about what you found",'
            f'"t":[track ids],"m":[movie ids],"g":[game ids]}}\n\n'
            f'Catalog: {json.dumps(catalog)}'
        }]
    )

    return json.loads(msg.content[0].text.strip())

def resolve_ai_search(ai_response, tracks, movies, games):
    tid = set(ai_response.get("t", []))
    mid = set(ai_response.get("m", []))
    gid = set(ai_response.get("g", []))
    return {
        "tracks":      [t for t in tracks if t["id"] in tid],
        "movies":      [m for m in movies if m["id"] in mid],
        "games":       [g for g in games  if g["id"] in gid],
        "explanation": ai_response.get("e"),
    }
```

# ДОДАТОК Е

## Лістинг коду експериментальної оцінки рекомендацій

### Е.1 Створення синтетичних користувачів для однодоменної оцінки

```
PERSONAS = [
    ("test_rpg_fan",     "game",  "RPG"), ("test_shooter_fan", "game",  "Shooter"),
    ("test_drama_fan",   "movie", "Drama"), ("test_scifi_fan",   "movie", "Sci-Fi"),
    ("test_metal_fan",   "track", "Metal"),  # ... 15 personas загалом
]
def create_test_profile(uid, item_type, genre, sample_size, seed):
    rng = random.Random(seed)
    cur.execute(f"SELECT id FROM {TABLE[item_type]} WHERE %s = ANY(genres)", (genre,))
    all_ids = [r["id"] for r in cur.fetchall()]
    rng.shuffle(all_ids)
    training     = set(all_ids[:sample_size])
    ground_truth = set(all_ids[sample_size:])
    for item_id in training:
        cur.execute("""INSERT INTO likes (user_uid, item_type, item_id)
            VALUES (%s, %s, %s)""", (uid, item_type, item_id))
    return training, ground_truth
```

### Е.2 Метрики оцінки якості ранжування

```
def precision_at_k(rec_ids, ground_truth, k):
    hits = sum(1 for x in rec_ids[:k] if x in ground_truth)
    return hits / k

def recall_at_k(rec_ids, ground_truth, k):
    hits = sum(1 for x in rec_ids[:k] if x in ground_truth)
    return hits / len(ground_truth) if ground_truth else 0.0

def mrr_at_k(rec_ids, ground_truth, k):
    for i, item_id in enumerate(rec_ids[:k]):
        if item_id in ground_truth:
            return 1.0 / (i + 1)
    return 0.0
```

### Е.3 Базові алгоритми для порівняння

```
def rec_popularity(items, training_set, item_type):
    key = (lambda x: -(x.get("listeners") or 0)) if item_type == "track" \
          else (lambda x: -(x.get("rating") or 0))
    pool = [i for i in items if i["id"] not in training_set]
    return [i["id"] for i in sorted(pool, key=key)]

def rec_random(items, training_set, seed):
    rng = random.Random(seed)
    pool = [i for i in items if i["id"] not in training_set]
    rng.shuffle(pool)
    return [i["id"] for i in pool]
```

### Е.4 Цикл оцінки з кількома випадковими seed

```
SEEDS = [42, 100, 200, 300, 400]; K_VALUES = [5, 10]

def evaluate_single_domain(personas, items_by_type):
    all_runs = []
    for seed in SEEDS:
        run_results = []
        for uid, item_type, genre in personas:
            training, ground_truth = create_test_profile(uid, item_type, genre,
                                                         SAMPLE_SIZE, seed)
            recs_tfidf = rec_tfidf(uid, item_type, items_by_type[item_type])
            recs_pop   = rec_popularity(items_by_type[item_type], training, item_type)
            recs_rand  = rec_random(items_by_type[item_type], training, seed)
            for algo, recs in [("tfidf", recs_tfidf), ("pop",   recs_pop),
                               ("rand",  recs_rand)]:
                for k in K_VALUES:
                    run_results.append({
                        "algo":      algo, "k":         k,
                        "precision": precision_at_k(recs, ground_truth, k),
                        "recall":    recall_at_k(recs, ground_truth, k),
                        "mrr":       mrr_at_k(recs, ground_truth, k),
                    })
        all_runs.append(run_results)
    return aggregate_mean_std(all_runs)
```

### Е.5 Метрики крос-доменної оцінки

```
def avg_semantic_score(top_k):
    return sum(item["semantic_score"] for item in top_k) / len(top_k)
def avg_shared_tags(top_k, anchor_tags):
    return sum(len(item["tags"] & anchor_tags) for item in top_k) / len(top_k)
def overlap_rate(top_k, anchor_tags):
    return sum(1 for item in top_k
               if len(item["tags"] & anchor_tags) > 0) / len(top_k)
```

### Е.6 Цикл крос-доменної оцінки з покомпонентним порівнянням алгоритму

```
DOMAIN_PAIRS = [
    ("track", "movie"), ("track", "game"), ("movie", "track"),
    ("movie", "game"), ("game",  "track"), ("game",  "movie"),
]
N_ANCHORS_PER_DOMAIN = 25
def evaluate_cross_domain(items_by_type):
    all_runs = []
    for seed in SEEDS:
        rng = random.Random(seed)
        for anchor_type, other_type in DOMAIN_PAIRS:
            anchors = rng.sample(items_by_type[anchor_type], N_ANCHORS_PER_DOMAIN)
            for anchor in anchors:
                top_hybrid   = cross_rec_hybrid(anchor, other_type)
                top_semantic = cross_rec_semantic_only(anchor, other_type)
                top_random   = cross_rec_random(anchor, other_type, rng)
                all_runs.append({
                    "hybrid":   compute_metrics(top_hybrid,   anchor),
                    "semantic": compute_metrics(top_semantic, anchor),
                    "random":   compute_metrics(top_random,   anchor),
                })
    return aggregate_mean_std(all_runs)
```