Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Бакалаврська кваліфікаційна робота на тему:

ПРОГРАМНИЙ МОДУЛЬ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ТОВАРІВ

Виконав: студент 4 курсу, групи 3КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)

Рудь В. Ю.
(прізвище та ініціали)

Керівник: к. т. н., доцент каф. КН
Арсенюк І. Р.
(прізвище та ініціали)

«04» червня 2025 р.

Рецензент: к. т. н., доцент каф. АІІТ
Кабачій В. В.
(прізвище та ініціали)

«05» червня 2025 р.

Допущено до захисту
Зав. кафедри Яровий А. А.
«06» червня 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра Комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д. т. н. Яровий А. А.
«05» травня 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Рудю Владиславу Юрійовичу

1. Тема роботи: Програмний модуль рекомендаційної системи товарів.
Керівник роботи: Арсенюк Ігор Ростиславович, доцент кафедри КН.
затверджені наказом вищого навчального закладу від «20» березня 2025 року №97
2. Термін подання студентом роботи «30» травня 2025 р.
3. Вихідні дані до роботи:
Вихідні дані до роботи: загальна кількість товарів у базі даних – не менше 500; кількість характеристик товарів – не менше 8; кількість груп товарів – не менше 10; мова програмування – об'єктно-орієнтована; база даних – у форматі PostgreSQL; операційна система – Windows.
4. Зміст текстової частини БКР (перелік питань, які потрібно розробити):
вступ; розробка загальної клієнт-серверної архітектури та структурних модулів системи; розробка алгоритмів авторизації, збереження профілю та історії взаємодії користувача; розробка структури каталогу товарів і алгоритмів пошуку та фільтрації; побудова функціональної схеми рекомендаційної системи; програмна реалізація модуля рекомендаційної системи товарів; висновки; список використаних джерел; додатки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):схема алгоритму формування рекомендацій у гібридній системі; структура клієнт-серверної архітектури системи рекомендацій; схема взаємодії модулів авторизації, профілю та історії користувача; схема структури даних каталогу товарів; функціональна схема роботи рекомендаційної системи товарів; розгортання локального сервісу; головна сторінка інтерфейсу; заповнена форма реєстрації користувача.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Арсенюк І. Р., к. т. н., доцент каф. КН		

7. Дата видачі завдання «05» травня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітки
		початок	закінчення	
1	Аналіз предметної області та постановка задачі рекомендаційної системи товарів	05.05.25	07.05.25	
2	Розробка алгоритмічної структури та архітектури рекомендаційної системи товарів	08.05.25	11.05.25	
3	Програмна реалізація та тестування рекомендаційної системи товарів	12.05.25	15.05.25	
4	Тестування роботи розробленого програмного забезпечення	16.05.25	26.05.25	
5	Розробка інструкції користувача	27.05.25	01.06.25	
6	Оформлення матеріалів до захисту БКР	02.06.25	04.06.25	

Студент _____ Рудь В. Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Арсенюк І. Р
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 92 сторінок формату А4, містить 13 рисунків, 6 таблиць, 4 додатки. Список використаних джерел налічує 22 найменувань.

Бакалаврська кваліфікаційна робота присвячена розробці персоналізованої рекомендаційної системи товарів для онлайн-торгівлі з використанням гібридного підходу щодо рекомендацій товарів. У роботі реалізовано клієнт-серверну архітектуру з підтримкою облікових записів користувачів, збору історії переглядів, а також алгоритму генерації персоналізованих рекомендацій на основі TF-IDF-аналізу описів товарів та поведінкових патернів користувачів. Програмне забезпечення розроблено з використанням мови Python, бібліотек Flask, Pandas та Scikit-learn. Для перевірки якості розробленого модуля було проведено тестування, яке включає перевірку компонентів системи, а також оцінювання точності рекомендацій за допомогою метрик, таких як Precision, Recall та F1.

Ключові слова: рекомендаційна система, гібридна модель рекомендацій, TF-IDF, персоналізація, Python, онлайн-торгівля, Flask.

ABSTRACT

The bachelor's qualification work consists of 92 pages of A4 format, contains 13 figures, 6 tables, 4 appendix. The list of used sources includes 22 items.

The bachelor's qualification work is devoted to the development of a personalized product recommendation system for online shopping using a hybrid approach to product recommendations. The work implements a client-server architecture with support for user accounts, collection of browsing history, as well as an algorithm for generating personalized recommendations based on TF-IDF analysis of product descriptions and user behavioral patterns. The software is developed using the Python language, Flask, Pandas, and Scikit-learn libraries. To verify the quality of the developed module, testing was conducted, which includes checking the system components, as well as evaluating the accuracy of recommendations using metrics such as Precision, Recall, and F1.

Keywords: recommendation system, hybrid recommendation model, TF-IDF, personalization, Python, online shopping, Flask.

ЗМІСТ

ВСТУП 5
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ТОВАРІВ ТА ПОСТАНОВКА ЗАДАЧІ 7
1.1 Аналіз актуальності застосування рекомендаційних систем у сфері онлайн-торгівлі 7
1.2 Огляд основних методів персоналізованих рекомендацій 12
1.3 Огляд найпопулярніших рекомендаційних систем в існуючих онлайн-платформах 16
1.4 Постановка задачі дослідження 19
1.5 Висновок до розділу 1 22
2 ПРОЕКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ТОВАРІВ 24
2.1 Обґрунтування вибору математичної моделі для рекомендаційної системи товарів 24
2.2 Розробка загальної клієнт-серверної архітектури та структурних модулів системи 27
2.3 Розробка алгоритмів авторизації, збереження профілю та історії взаємодії користувача 30
2.4 Розробка структури каталогу товарів і алгоритмів пошуку та фільтрації 34
2.5 Побудова функціональної схеми рекомендаційної системи 37
2.6 Висновок до розділу 2 40
3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ТОВАРІВ 42
3.1 Обґрунтування вибору інструментальних засобів для реалізації програмного модуля 42
3.2 Реалізація модуля рекомендаційної системи товарів 45
3.3 Тестування програмного модуля рекомендаційної системи товарів 55
3.4 Аналіз точності рекомендацій за допомогою метрик оцінювання 60
3.5 Висновок до розділу 3 62
ВИСНОВКИ 65
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 68
ДОДАТОК А (ОБОВ'ЯЗКОВИЙ) ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ 72
ДОДАТОК Б (ОБОВ'ЯЗКОВИЙ) ЛІСТИНГ ПРОГРАМИ 73
ДОДАТОК В (ОБОВ'ЯЗКОВИЙ) ІЛЮСТРАТИВНА ЧАСТИНА 81
ДОДАТОК Г (ДОВІДНИКОВИЙ) ІНСТРУКЦІЯ КОРИСТУВАЧА 87

ВСТУП

Актуальність досліджень. Інформаційні технології в даний час радикально змінюють механізми взаємодії між користувачем та цифровими сервісами. З розвитком електронної комерції, зокрема онлайн-магазинів, платформ для дистрибуції контенту (фільмів, книг, курсів тощо), зростає потреба в ефективних інструментах персоналізації. У цьому контексті рекомендаційні системи відіграють вирішальну роль, адже саме вони дозволяють кожному користувачеві отримувати пропозиції, максимально наближені до його інтересів, звичок і потреб. Завдяки цьому підвищується не лише якість користувацького досвіду, а й ефективність бізнес-процесів: зростає конверсія, лояльність клієнтів і середній чек.

Значна частина сучасних платформ (Amazon, Netflix, YouTube, Spotify тощо) вже активно впровадили ті чи інші форми рекомендаційних алгоритмів, демонструючи істотне зростання користувацької активності та доходів. Проте більшість малих та середніх компаній не мають ресурсу чи експертизи для впровадження повноцінних гібридних рекомендаційних систем. У зв’язку з цим розробка модульного, гнучкого, адаптивного та водночас нескладного у реалізації рекомендаційного рішення, яке можна легко масштабувати чи інтегрувати в різні програмні продукти, є надзвичайно актуальною задачею.

Глобальна цифровізація торгівлі та стрімке зростанням обсягів інформації ускладнюють вибір для споживача і водночас зумовлюють потребу в ефективних механізмах підтримки прийняття рішень. Особливо актуальним є створення адаптивних рекомендаційних систем для малих і середніх підприємств, які не мають змоги впроваджувати складні рішення на рівні Amazon чи Netflix, але потребують аналогічної функціональності для підвищення конкурентоспроможності.

Об’єкт дослідження – процес персоналізованого формування рекомендацій користувачам.

Предмет дослідження – програмне забезпечення рекомендаційної системи товарів.

Мета роботи – розширення функціональних можливостей програмного модуля рекомендаційної системи товарів.

Задачі дослідження:

1) здійснити аналіз сучасних методів, що покладено в основу функціонування рекомендаційних систем;

2) виконати аналітичний огляд існуючих рекомендаційних систем;

3) розробити загальний алгоритм роботи рекомендаційної системи;

4) розробити схему взаємодії модулів авторизації, профілю та історії користувача

5) розробити функціональну схему рекомендаційної системи товарів;

6) реалізувати програмний модуль рекомендаційної системи товарів та здійснити його тестування.

Ця бакалаврська кваліфікаційна робота виконана відповідно до вимог, викладених у методичних вказівках до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп'ютерні науки» [1], затверджених на кафедрі та рекомендованих до використання у ВНТУ.

Апробація роботи.

Робота апробувалась на наступній конференції:

Міжнародна науково-практична Інтернет-конференція студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (мн-2025)». Доповідь: «Програмний модуль рекомендаційної системи товарів» [2].

Публікації: електронні тези конференції Міжнародна науково-практична Інтернет-конференція студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (мн-2025)» [2]; подано заявку на реєстрацію авторського права на твір у формі комп'ютерної програми – «Програмний модуль рекомендаційної системи товарів», номер заявки С202505392 від 09.06.2025 р.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ТОВАРІВ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз актуальності застосування рекомендаційних систем у сфері онлайн-торгівлі

Цифрова економіка, де обсяги інформації стрімко зростають, зростає й роль інтелектуальних технологій, здатних адаптуватися до індивідуальних запитів користувачів. Онлайн-торгівля, як один з найдинамічніших сегментів ринку, зазнає суттєвих трансформацій під впливом персоналізації, що стала ключовим фактором у побудові довготривалих взаємин між продавцем та покупцем. У центрі цих змін знаходяться рекомендаційні системи, які забезпечують відбір найбільш релевантної інформації, товарів або послуг відповідно до уподобань кожного окремого користувача.

Онлайн-платформи стикаються з постійним зростанням кількості товарів, що ускладнює процес навігації для кінцевого споживача. За відсутності ефективної системи сортування і рекомендацій користувач змушений витрачати надмірно багато часу на пошук необхідного продукту, що негативно впливає на показники утримання клієнтів та рівень завершених транзакцій. Рекомендаційна система дозволяє автоматизувати процес відбору пропозицій, керуючись історією поведінки користувача, популярністю товарів, релевантністю описів та іншими параметрами. Таким чином, вона не лише підвищує зручність взаємодії, а й виконує функцію «інтелектуального продавця», який знає, що і коли запропонувати [3].

Еволюція електронної комерції призвела до необхідності розширення функціоналу цифрових сервісів, що проявляється у запровадженні складних модулів аналізу даних. Рекомендаційна система, як окрема компонента таких сервісів, вже не розглядається як елемент розкоші, а є базовою функцією, без якої неможливо уявити конкурентоспроможну платформу. Її використання

забезпечує гнучке реагування на запити ринку, адаптацію до поведінкових змін користувачів, а також дозволяє підвищити коефіцієнт конверсії.

За даними звітів McKinsey & Company, до 35% продажів на Amazon генеруються саме через інтегровану рекомендаційну систему. У Spotify та Netflix, системи персоналізованого підбору контенту забезпечують понад 80% споживання медіа. Варто наголосити, що навіть у менш масштабних проєктах, впровадження адаптивних рекомендацій призводить до зростання кількості повторних покупок, зниження рівня відмов та підвищення лояльності користувача [4].

Крім прикладів успішної реалізації, слід звернути увагу й на сучасні тенденції. Зокрема, концепція «zero-click purchase», тобто покупки без активного пошуку, стає реальністю лише за умов функціонування високоточної рекомендаційної системи. Такий рівень персоналізації дозволяє скоротити шлях від перегляду до покупки до одного кліку, що знижує ризик втрати клієнта через складність інтерфейсу чи перевантаженість вибору.

Зростання популярності мобільних платформ та соціальної комерції також створює нові вимоги до рекомендацій. Користувач очікує не лише точних підказок, але й моментального реагування, що потребує інтеграції алгоритмів в реальному часі. Своєю чергою, це викликає попит на прості в реалізації, масштабовані та ефективні з погляду ресурсів моделі, які можна швидко адаптувати під будь-який каталог товарів. У таких умовах саме модульні, гібридні системи рекомендацій, орієнтовані на поєднання поведінкових, контентних та колективних індикаторів, вважаються найоптимальнішими.

Для демонстрації ролі рекомендаційних систем у сфері онлайн-торгівлі можна навести узагальнене порівняння ефективності з/без їх використання (табл.1.1).

Таблиця 1.1 – Порівняння метрик онлайн-торгівлі залежно від наявності рекомендаційної системи

Показник	**Без рекомендацій**	**З рекомендацією**	**Приріст (%)**
Середня кількість переглядів	5,2	12,4	+138%
Коефіцієнт конверсії	1,7%	3,6%	+112%
Середній чек	$18,60	$24,90	+34%
Частка повторних покупок	21%	42%	+100%
Час, витрачений на платформі	3 хв 40 сек	7 хв 25 сек	+102%

Як видно з таблиці, наявність рекомендаційного модуля впливає на всі ключові бізнес-показники: тривалість взаємодії користувача з платформою зростає удвічі, що прямо корелює з підвищенням шансів на покупку. Крім того, збільшується середній чек та відсоток повторних покупок, що є критичним фактором для довгострокової прибутковості платформи.

Не менш важливою є й роль таких систем у формуванні іміджу сервісу. Якщо користувач щоразу отримує актуальні та цікаві для себе пропозиції, це створює враження, що система «розуміє» його потреби. Відповідно, підвищується рівень довіри, знижується ймовірність переходу до конкурентів, а загальна оцінка бренду на ринку покращується.

Сучасні рекомендаційні системи широко впроваджені у провідних платформах електронної комерції та цифрового контенту. Їхні архітектури, методи та ефективність суттєво відрізняються в залежності від типу платформи, цільової аудиторії, обсягу даних та бізнес-мети. У цьому контексті доцільним є детальний розгляд кількох найвідоміших прикладів, які демонструють різні підходи до побудови інтелектуальних систем персоналізації.

Одним з найвідоміших кейсів є Amazon, платформа, що використовує гібридну систему рекомендацій з переважанням item-based колаборативної фільтрації. Алгоритм базується на принципі: «Клієнти, які купили це, також купили…», який дозволяє швидко генерувати релевантні варіанти товарів. Amazon зберігає й аналізує інформацію про кожен клік, перегляд, пошук, придбання, оцінку та навіть час взаємодії з елементами інтерфейсу. Інтеграція

цих сигналів дозволяє будувати багатовимірний вектор поведінки користувача, на основі якого система формує персоналізовані рекомендації. Крім того, Amazon активно використовує кластеризацію користувачів для формування узагальнених шаблонів поведінки, що покращує точність рекомендацій навіть для нових користувачів з мінімальною історією [5].

Не менш цікавою є реалізація рекомендаційної системи в Netflix. На початкових етапах розвитку компанія застосовувала лише рейтингові моделі (наприклад, SVD), однак з часом перейшла до складних ансамблів методів. Сьогодні Netflix використовує не тільки класичну колаборативну фільтрацію, а й глибокі нейронні мережі, які враховують сотні ознак – від жанру, актора та часу доби перегляду до темпу зміни кадрів у фільмі. Особливість системи Netflix полягає у динамічному оновленні рекомендацій залежно від поведінки користувача під час сеансу перегляду: якщо користувач припинив перегляд після 5 хвилин, цей факт одразу враховується у подальших підказках.

Платформа Spotify відзначається складною багаторівневою системою рекомендацій, що включає як глобальне ранжування (на основі схожості профілів користувачів), так і локальне персоналізоване сортування (враховуючи настрій, жанрові вподобання, останні прослуховування). Компанія активно використовує бібліотеку LightFM, Word2Vec-моделі для побудови векторних представлень пісень, а також алгоритми reinforcement learning для формування динамічних плейлистів. Щотижневий плейлист «Discover Weekly», сформований за допомогою змішаної моделі content+collaborative, вважається одним із найуспішніших продуктів компанії, який щотижня залучає мільйони слухачів по всьому світу [6].

Ще одним прикладом ефективної інтеграції рекомендацій є YouTube, де використовується дворівнева система: перший етап – грубе фільтрування та ранжування кандидатів за допомогою нейромережевих моделей (наприклад, wide&deep learning), другий – детальне персоналізоване сортування на основі довгої історії переглядів. YouTube інтегрує сигнали з понад 80 джерел, включаючи швидкість прокручування, паузи, тип пристрою, регіон та навіть

час доби. У результаті система здатна формувати стрічку, яка адаптується в реальному часі до динаміки настрою користувача [7].

Особливої уваги заслуговує й приклад AliExpress, платформи, яка орієнтована на глибоку персоналізацію через обробку сесійних даних. Її рекомендаційна система адаптується до мовного, культурного та поведінкового профілю користувача. В AliExpress реалізовано кастомну систему гібридних рекомендацій, яка використовує машинне навчання для побудови дерев рішень, та глибокі градієнтні бустери для сортування товарів. Для боротьби з «проблемою новизни» платформа додає в систему додаткову евристику, що враховує динаміку нових надходжень та їхню потенційну привабливість [8].

Узагальнимо відмінності між системами у табл.1.2.

Таблиця 1.2 – Приклади реалізації рекомендаційних систем у світових онлайн-сервісах

Платформа	**Алгоритмічна основа**	**Джерела даних**	**Особливість реалізації**
Amazon	Item-based Collaborative Filtering	Покупки, лайки, перегляди	Висока точність без нейронних мереж
Netflix	Ensemble + Deep Learning	Перегляди, зупинки, рейтинги	Динамічні рекомендації у реальному часі
Spotify	Word2Vec + LightFM + RL	Прослуховування, лайки, повтори	Автоматичні плейлисти та зміна рекомендацій за жанром
YouTube	Wide&Deep Learning	Поведінка, кліки, перегляди, час	Контекстна персоналізація через два рівні фільтрації
AliExpress	Decision Trees + Boosting	Клікстрими, країна, пристрій	Підлаштування під локальні особливості

З аналізу видно, що в центрі кожної системи – адаптація до контексту. Якщо Netflix працює з тривалими сесіями перегляду, то YouTube адаптується до мікроповедінки в рамках кількох хвилин. Spotify вміє будувати емоційні профілі слухача, а AliExpress – враховує цінові переваги. Amazon, як

універсальний маркетплейс, навпаки, робить ставку на гігантські масиви даних і простоту рекомендацій.

Ці приклади доводять, що ефективна рекомендаційна система – це не обов'язково складна технологія. Її успіх залежить від точності адаптації до бізнес-моделі, обсягів даних, та особливостей взаємодії користувача із системою. Тому при розробці власного програмного модуля варто брати до уваги не лише наукову обґрунтованість методів, а й практичну доцільність їх застосування в реальному середовищі.

Таким чином, аналіз поточної ситуації вказує на те, що впровадження ефективної рекомендаційної системи є не просто бажаним елементом онлайн-торгівлі, а стратегічною необхідністю. Усе більше компаній приходять до усвідомлення, що без персоналізації неможливо побудувати масштабовану модель зростання. При цьому, відзначається потреба в універсальних рішеннях, які можна швидко інтегрувати, масштабувати, підтримувати і модифікувати в умовах динамічного ринку.

Отже, актуальність розробки модульної рекомендаційної системи товарів, яка враховує поведінкові та контентні характеристики, є беззаперечною. Вона базується на реальних запитах бізнесу, відповідає сучасним трендам у сфері цифрової трансформації та має чітко виражену прикладну цінність як для кінцевого споживача, так і для розробника програмного забезпечення.

1.2 Огляд основних методів персоналізованих рекомендацій

Персоналізація – ключовий елемент сучасних цифрових систем взаємодії з користувачем. У той час як на перших етапах розвитку онлайн-платформ уся комунікація з клієнтом була зведена до фіксованих пропозицій і простих фільтрів за категоріями, сьогоднішня реальність вимагає значно глибшого індивідуального підходу. І саме рекомендаційні системи стали відповіддю на цей виклик, поєднуючи в собі аналітику, штучний інтелект і моделювання поведінки. Поява таких систем є результатом поступового еволюційного

переходу від статичних інтерфейсів до динамічних, адаптивних рішень, здатних навчатися на основі взаємодії з користувачем.

Класичне уявлення про рекомендаційну систему базується на концепції аналізу даних користувача з метою створення персоналізованого набору пропозицій. Цей процес включає декілька ключових етапів: збір інформації про користувача та товари, побудову профілю споживача, обрахунок схожості між профілями чи об'єктами та, зрештою, генерацію списку релевантних результатів. На кожному з цих етапів застосовуються різноманітні методи, які, залежно від принципу дії, можна умовно поділити на три групи: контентно-орієнтовані (CBF), колаборативні (CF) та гібридні.

Контентна фільтрація бере за основу аналіз характеристик товарів, що подобаються користувачу, та пошук схожих за цими ознаками продуктів. Такий підхід вимагає наявності чіткого опису товарів, який можна структурувати та обробити алгоритмічно. Наприклад, система може аналізувати текстовий опис книги, виявляти ключові слова, жанри, стиль, а далі пропонувати інші книги з подібними ознаками. Основою обчислень у більшості моделей є метод TF-IDF, який дозволяє перетворити текст у векторну форму та обчислювати ступінь подібності між об’єктами за допомогою косинусної метрики. Хоча цей підхід працює швидко та не потребує великого масиву поведінкових даних, він має суттєвий недолік – неспроможність вийти за межі вже вподобаних характеристик. Тобто система ніколи не запропонує товар, що принципово відрізняється від попередніх вподобань, навіть якщо він міг би зацікавити користувача [9].

Колаборативна фільтрація, на відміну від контентної, ґрунтується на взаємозв’язках між користувачами або між самими товарами. Тут діє логіка: «якщо інші користувачі, схожі на тебе, вподобали певний товар, то, ймовірно, він сподобається і тобі». Система порівнює матрицю оцінок або вподобань, шукає найбільш подібних користувачів чи товарні пари, і на цій основі формує рекомендації. Існує два основні різновиди: user-based CF (порівняння користувачів) та item-based CF (порівняння товарів). Найпростішою реалізацією

є метод k-ближчих сусідів (KNN), але для великих даних краще працюють методи латентного факторного аналізу – SVD, ALS та інші. Перевагою цього підходу є здатність пропонувати неочевидні товари, які не мають прямої схожості з уподобаннями користувача, але були високо оцінені іншими. Проте колаборативна фільтрація страждає від так званої проблеми холодного старту: вона потребує великої кількості історичних даних і практично не працює для нових користувачів або нових товарів [10].

У відповідь на обмеження кожного з окремих методів було сформовано концепцію гібридних систем. Вони поєднують у собі переваги як контентного, так і колаборативного підходів, компенсуючи недоліки один одного. У гібридній системі оцінка релевантності товару складається з двох або більше компонентів, кожна з яких розраховується окремо, а потім об'єднується у фінальне значення на основі заданих ваг або навчання. Це дозволяє системі, з одного боку, пропонувати товари, які подібні до вже вподобаних, а з іншого – орієнтуватися на поведінку схожих користувачів. Найчастіше такі системи реалізуються через ансамблі моделей або глибокі нейронні мережі з множинними входами. Вони складніші в реалізації, проте мають найкращі показники точності за наявності достатньо великих даних [11].

На додачу до трьох базових підходів, сучасні тенденції вказують на активне впровадження нових методів, зокрема побудованих на основі глибокого навчання. Використання рекурентних або трансформерних архітектур дозволяє враховувати не лише факт уподобання, а й послідовність, час, контекст, емоційний стан користувача, а також зовнішні обставини (наприклад, сезонність, погодні умови тощо). Такі системи, попри складність реалізації, відкривають нові горизонти для побудови так званих «розумних рекомендацій», які не лише пропонують товари, а й формують цілісний цифровий простір навколо інтересів користувача [12].

Для наочного зіставлення основних підходів представимо коротке порівняння ключових характеристик (табл.1.3).

Таблиця 1.3 – Порівняння основних підходів до побудови рекомендаційних систем

Ознака	**Контентна фільтрація**	**Колаборативна фільтрація**	**Гібридна модель**
Джерело даних	Властивості товару	Поведінка інших користувачів	Обидва типи
Нові товари	Працює добре	Не працює	Працює
Нові користувачі	Працює частково	Не працює	Працює частково
Неочевидні рекомендації	Рідко	Часто	Часто
Вимоги до даних	Опис товарів	Велика кількість рейтингів	Великі обсяги з обох типів
Складність реалізації	Низька	Середня	Висока
Швидкість рекомендацій	Висока	Середня	Залежить від реалізації
Масштабованість	Добра	Обмежена	Добра за належної архітектури

Разом із цим залишається актуальним і питання прозорості роботи алгоритмів. Простий підхід TF-IDF або item-based CF легко пояснити користувачеві, що забезпечує довіру. У той час як у складних глибоких моделях пояснити причину тієї чи іншої рекомендації – надзвичайно важко. Саме тому зростає інтерес до методів explainable AI (XAI), які дозволяють виводити причини вибору кожного окремого елемента у списку рекомендацій [13].

Отже, кожен із розглянутих підходів має свої переваги й недоліки. Вибір конкретного методу має визначатися характером даних, цілями платформи, рівнем доступних ресурсів та вимогами до точності, швидкості й масштабованості. У межах цієї кваліфікаційної роботи доцільним є застосування гібридної моделі, яка базується на поєднанні контентного аналізу та простої колаборативної логіки. Такий підхід забезпечить баланс між ефективністю та простотою реалізації, що цілком відповідає вимогам до сучасної рекомендаційної системи для малих та середніх онлайн-платформ.

1.3 Огляд найпопулярніших рекомендаційних систем в існуючих онлайн-платформах

Побудова ефективної рекомендаційної системи не є завданням, що зводиться лише до вибору алгоритму. Її успішна реалізація залежить від багатьох чинників: глибини інтеграції в інтерфейс користувача, швидкості реакції системи, релевантності результатів, збереження приватності даних, адаптації до змінних патернів поведінки, обчислювальних потужностей, масштабованості та багатьох інших технічних і організаційних аспектів. Щоб зрозуміти, як теоретичні підходи реалізуються на практиці, доцільно розглянути конкретні приклади платформ, де рекомендаційні системи стали ключовим компонентом функціонування.

Одним із найвідоміших і найчастіше цитованих кейсів є платформа Amazon, яка використовує рекомендаційні алгоритми ще з початку 2000-х років. В основі їхньої системи лежить item-based collaborative filtering, що було суттєвим відступом від традиційної user-based фільтрації, яка не масштабувалась достатньо ефективно. Основна ідея полягала в тому, щоб шукати схожі товари замість схожих користувачів, що дозволило значно зменшити складність обчислень та прискорити генерацію рекомендацій. Ця система не лише впливає на головну сторінку магазину, а й на кожну сторінку товару, в кошику, в email-розсилках, у push-повідомленнях. Усі ці точки дотику користувача з системою підключені до єдиного механізму персоналізації, який використовує понад сотню сигналів: історію переглядів, попередні покупки, вподобання, час доби, тип пристрою, географічне розташування, швидкість прокручування сторінки тощо. Впровадження рекомендацій на рівні всього користувацького досвіду стало основою стратегії Amazon з побудови довготривалих відносин із клієнтом [14].

Інший приклад – стрімінговий сервіс Netflix, який завдяки своїй рекомендаційній системі не просто утримує глядача, а фактично формує його смакові уподобання. Алгоритм Netflix враховує такі чинники, як історія

переглядів, темп споживання контенту, тривалість сесій, взаємодія з трейлерами, оцінки фільмів, відгуки, а також агреговані дані поведінки інших користувачів із подібними вподобаннями. Система поєднує факторну декомпозицію (SVD), байєсівські мережі, дерево класифікації, а також нейромережі для обробки складних даних, зокрема текстів описів фільмів, заголовків та навіть метаданих (рік, режисер, жанр тощо). Ключовим моментом є використання A/B-тестування для кожної версії інтерфейсу рекомендацій, що дозволяє Netflix точно вимірювати вплив змін на показники утримання та перегляду [15].

Платформа Spotify пішла далі, розглядаючи рекомендації як елемент емоційного супроводу користувача. Вона не просто пропонує схожі треки на основі попередніх прослуховувань, а намагається моделювати емоційний стан користувача та відповідно адаптувати музичні добірки. Наприклад, система може зчитувати час доби, день тижня, тип активності (через підключення до фітнес-додатків), щоб створювати динамічні плейлисти. Найбільш популярна функція – Discover Weekly – генерується з використанням гібридного підходу, що включає Word2Vec-моделі для представлення треків у векторному просторі, колаборативну фільтрацію для побудови користувацьких профілів, а також механізми кластеризації й фільтрації шуму [16].

Цікавий підхід демонструє також YouTube, рекомендаційна система якого працює на двох рівнях: спочатку формується підмножина релевантних відео (retrieval), а потім відбувається їх персоналізоване ранжування (ranking). Алгоритм враховує понад 80 ознак, включаючи середню тривалість перегляду, швидкість взаємодії, тематику попередніх відео, а також контекст перегляду (мобільний чи десктопний пристрій, підключення до Wi-Fi чи мобільного інтернету, місце перебування). Особливістю YouTube є активна робота з негативними сигналами – наприклад, швидкий вихід із відео або відмова від перегляду після рекомендації – що дозволяє системі уникати нав’язування непотрібного контенту [17].

З іншого боку, AliExpress та інші великі торгові платформи орієнтовані на пошук балансу між персоналізацією та комерційною доцільністю. Їхні рекомендації враховують не лише схожість товарів чи користувачів, а й маржинальність, актуальність промоакцій, сезонність, запаси на складі, навіть рекламні кампанії продавців [18].

Щоб продемонструвати відмінності у стратегічних підходах до рекомендацій, доцільно порівняти розглянуті системи за критеріями складності, глибини персоналізації, джерел даних і адаптивності (табл.1.4).

Кожна з платформ надає пріоритет певним характеристикам: Netflix орієнтується на довгострокове утримання, тому інвестує в глибину моделі; Amazon – на швидкість та точність; Spotify – на емоційний контекст; YouTube – на гнучке реагування; AliExpress – на бізнес-ефективність і операційну масштабованість. Це свідчить про те, що побудова рекомендаційної системи завжди вимагає узгодження технічних можливостей із бізнес-цілями.

Отже, аналіз прикладів практичного впровадження дозволяє зробити висновок: універсального алгоритму не існує, однак ефективною є модель, яка адаптована до контексту використання. Саме тому при розробці рекомендаційної системи товарів у межах цієї роботи буде обрано гібридний підхід, що поєднує контентну фільтрацію з елементами колаборативної логіки. Це дозволить забезпечити високу точність рекомендацій, навіть за умов обмеженого обсягу даних, та забезпечити гнучку архітектуру, придатну до масштабування й реального впровадження.

Таблиця 1.4 – Порівняння стратегій реалізації рекомендацій на відомих онлайн-платформах

Платформа	**Методологія**	**Джерела даних**	**Рівень персоналізації**	**Архітектура**
Amazon	item-based CF	історія покупок, кліки, кошик	середній	централізована + кешування

Продовження таблиці 1.4 – Порівняння стратегій реалізації рекомендацій на відомих онлайн-платформах

Платформа	**Методологія**	**Джерела даних**	**Рівень персоналізації**	**Архітектура**
Netflix	гібридна + глибоке навчання	перегляди, сесії, оцінки, метадані	дуже високий	модульна, з ML-оптимізацією
Spotify	гібридна + Word2Vec	треки, повтори, контекст, доба	високий	потокова, з RL-компонентами
YouTube	two-stage NN	перегляди, skip-rate, пристрій	дуже високий	широкомасштабна, на TensorFlow
AliExpress	ML + бізнес-логіка	вподобання, запаси, ціни, логістика	середній	змішана, з евристичними модулями

1.4 Постановка задачі дослідження

У попередніх підрозділах було встановлено, що впровадження рекомендаційної системи у сферу онлайн-торгівлі є не лише бажаним, а фактично необхідним етапом розвитку сучасних платформ. З урахуванням існуючих технологічних тенденцій та досвіду провідних гравців ринку, постає завдання побудови ефективної, адаптивної, масштабованої та водночас технічно досяжної системи персоналізованих рекомендацій. Формулювання задачі розробки такої системи передбачає врахування множини взаємопов’язаних аспектів: від загальної цілі до архітектурних, алгоритмічних і функціональних характеристик. На цьому етапі необхідно визначити цільові орієнтири, обмеження, вхідні та вихідні дані, технічні вимоги до продуктивності, гнучкості, стабільності й інтеграційної сумісності.

Метою побудови системи є розширення функціональності програмного модуля, здатного генерувати персоналізовані рекомендації товарів для конкретного користувача. Ці рекомендації повинні базуватись на змішаній логіці, яка поєднує елементи контентної фільтрації (через аналіз описів товарів та їх властивостей) із базовою колаборативною логікою, що враховує історію дій користувача та вподобання інших. Такий підхід забезпечує широку

адаптацію до умов платформи, де обсяг історичних даних може бути обмеженим, а структура каталогу – динамічною.

Постановка задачі передбачає поетапне уточнення вимог до системи. На загальному рівні функціонування вона має реалізовувати підхід до фіксації дій користувача — від перегляду сторінки товару до додавання його в кошик, позначення вподобання або виставлення оцінки. Зібрані поведінкові дані використовуються для поступового оновлення персонального профілю. Для кожного користувача формується індивідуальний набір характеристик у вигляді векторної репрезентації, яка змінюється відповідно до його активності. Цей вектор слугує основою для створення списку релевантних товарів із поточного каталогу. Кожен товар, у свою чергу, представлений власним вектором, сформованим на основі його опису, категорії, ключових слів, рейтингу, популярності та інших параметрів.

З точки зору даних, система повинна мати доступ до базової інформації про користувача (ідентифікатор, вік, стать, за бажанням – інтереси), а також до журналу дій, який включає перегляди, вподобання, покупки, оцінки та можливі коментарі. Водночас необхідно забезпечити зберігання структури каталогу товарів із прив’язкою до метаінформації, включаючи назву, опис, категорію, атрибути, рейтинг, актуальність, ціну, наявність тощо. Важливою вимогою є також підтримка базових функцій пошуку й фільтрації, оскільки ці механізми є невід’ємною частиною будь-якої торгової системи.

З погляду функціональності, система повинна підтримувати кілька режимів формування рекомендацій: початкові (навіть за відсутності історії користувача), контекстні (на основі останніх дій), глибинні (з урахуванням усієї доступної історії). Рекомендації мають бути релевантними не лише поточному стану користувача, а й логіці самого сервісу – тобто враховувати сезонність, популярність, новизну товару тощо. Кожне формування списку повинно враховувати обмеження щодо кількості результатів, можливість сортування, обмеження за категорією або фільтрами, заданими користувачем.

Окремим аспектом задачі є визначення нефункціональних вимог, які часто мають критичний вплив на працездатність системи. Сюди належать вимоги до швидкодії (формування списку рекомендацій за частки секунди), стабільності (відсутність збоїв навіть при високому навантаженні), масштабованості (можливість обробки великої кількості користувачів одночасно), безпеки (автентифікація, захист даних), прозорості (можливість пояснення рекомендації), а також інтеграційної сумісності з іншими модулями платформи. Для ілюстрації структури таких вимог наведено узагальнену таблицю 1.5.

Таблиця 1.5 – Функціональні та нефункціональні вимоги до системи рекомендацій

Категорія	**Вимога**
Формування рекомендацій	Генерація списку товарів відповідно до профілю користувача
Облік дій користувача	Зберігання переглядів, оцінок, покупок, вподобань
Облік параметрів товару	Аналіз описів, категорій, рейтингу, наявності
Взаємодія з користувачем	Підтримка оцінювання, лайків, коментарів
Можливість масштабування	Робота з великою кількістю товарів та одночасних сесій
Швидкодія	Формування результатів за <500 мс
Безпека	Захист даних користувача, підтримка авторизації
Пояснюваність	Можливість виведення причин рекомендації
Модульність	Розділення на блоки: рекомендації, авторизація, каталог, аналітика
Інтегрованість	Сумісність із зовнішніми інтерфейсами та БД

Таким чином, сформульована задача охоплює широкий спектр аспектів: від математичної моделі до її технічного втілення в конкретному програмному рішенні. Зважаючи на це, вибір гібридної моделі, яка базується на TF-IDF-аналітиці описів товарів та базових елементах колаборативного аналізу оцінок, вважається оптимальним рішенням. Він забезпечує прийнятну точність рекомендацій, можливість адаптації до невеликої історії користувача, невисоку

складність реалізації, а також відкриває шлях до майбутнього масштабування системи.

Подальші етапи реалізації рекомендаційної системи повинні бути зосереджені на виборі інструментальних засобів, моделюванні архітектури, створенні бази даних, реалізації ключових алгоритмів та тестуванні якості отриманих результатів. Описані функціональні вимоги будуть служити критерієм верифікації роботи кожного з етапів розробки.

1.5 Висновок до розділу 1

Проведений аналіз застосування рекомендаційних систем дозволив всебічно обґрунтувати актуальність розробки їх у сфері онлайн-торгівлі. Було встановлено, що в умовах швидкого зростання обсягів цифрового контенту та кількості доступних товарів саме інтелектуальні механізми фільтрації є основним інструментом адаптації інформаційного середовища під індивідуальні запити користувача. Практика провідних цифрових платформ засвідчує, що використання рекомендацій не лише покращує користувацький досвід, а й підвищує ефективність бізнесу за такими показниками, як коефіцієнт конверсії, середній чек, глибина взаємодії та рівень утримання клієнтів.

У ході огляду основних методів до побудови рекомендаційних систем було виявлено, що кожен із них має свої переваги та обмеження. Контентна фільтрація є простою у реалізації та ефективною у ситуаціях з обмеженою історією дій, проте не здатна вийти за межі вже відомих вподобань. Колаборативна фільтрація дозволяє виявляти неочевидні зв’язки між користувачами та товарами, однак потребує значного обсягу даних і погано працює у випадку нових користувачів або товарів. Гібридні моделі, у свою чергу, поєднують переваги обох підходів, забезпечуючи баланс між точністю, адаптивністю та стабільністю, і саме вони розглядаються як найбільш перспективні у сучасних реалізаціях.

Огляд найпопулярніших рекомендаційних систем в існуючих онлайн-платформах, таких як Amazon, Netflix, Spotify, YouTube, AliExpress та інших, продемонстрував широкий спектр архітектурних та алгоритмічних рішень. Кожна з цих компаній адаптує рекомендаційну систему відповідно до своїх цілей, контенту, структури даних і поведінкових особливостей своєї аудиторії. Це підтвердило доцільність гнучкого підходу до проектування, де ключовим фактором успіху є не стільки обраний алгоритм, скільки його інтеграція в загальну логіку платформи.

У підрозділі, присвяченому постановці задач дослідження, було визначено основні функціональні та нефункціональні вимоги до системи. Вони включають облік дій користувача, збереження історії, побудову динамічного профілю, підтримку рекомендацій у різних режимах, забезпечення безпеки та масштабованості. У результаті сформульовано цілісну постановку задачі створення рекомендаційного модуля, який поєднує алгоритмічну складову з архітектурною, інформаційною та інтеграційною.

Таким чином, на основі глибокого теоретичного та практичного аналізу сформовано обґрунтування вибору гібридного підходу для реалізації рекомендаційної системи товарів. Це рішення відповідає сучасним вимогам персоналізації, дозволяє забезпечити адаптацію до поведінки користувача навіть за обмежених даних, а також відкриває перспективи масштабування та модифікації системи в подальших етапах розвитку.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ТОВАРІВ

2.1 Обґрунтування вибору математичної моделі для рекомендаційної системи товарів

Успішне функціонування будь-якої рекомендаційної системи базується на ефективності використовуваного алгоритмічного ядра. Саме алгоритм відповідає за те, які товари буде запропоновано користувачеві, на основі чого будуються ці припущення, як система реагуватиме на нові дані й наскільки релевантними будуть результати. Вибір алгоритму є критично важливим, адже від нього залежить як точність прогнозування, так і продуктивність системи в умовах змінних навантажень та масштабів.

На початковому етапі розробки було визначено, що найоптимальнішою для даного проєкту є гібридна модель, яка поєднує контентну фільтрацію з елементами колаборативного аналізу. Такий підхід дозволяє досягти балансу між точністю і обчислювальною складністю, зберігаючи адаптивність до обмежених обсягів історичних даних. Контентна складова моделі буде базуватися на векторному поданні текстових описів товарів за допомогою методу TF-IDF [19]. Суть цього методу полягає у виявленні найважливіших термінів у тексті шляхом врахування їхньої частоти в одному документі та оберненої частоти появи в усьому корпусі. Формально значення TF-IDF для терміна t в документі d (1.1) визначається як:

$$TFIDF(t,d) = tf(t,d) \cdot \log\left(\frac{N}{df(t)}\right), \quad (2.1)$$

де $tf(t,d)$ – частота терміна у $df(t)$ – кількість документів, у яких зустрічається термін, N – загальна кількість документів.

Кожен товар на основі текстового опису, назви та категорії буде перетворено у вектор ознак, що дозволить застосовувати метрику косинусної подібності (1.2) для визначення рівня схожості між товарами:

$$\text{sim}(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|}, \tag{2.2}$$

де $\vec{a}$ і $\vec{b}$ – векторні представлення двох товарів. Завдяки цьому кожен новий або переглянутий товар автоматично дозволяє формувати список інших схожих товарів, що вже є в системі [20].

Контентна фільтрація добре підходить для роботи з новими товарами, адже не потребує наявності історії оцінок. Водночас її недоліком є те, що вона працює в межах вже заданих характеристик і не здатна генерувати неочікувані рекомендації. Для розширення меж релевантності до системи буде інтегровано простий модуль колаборативного аналізу, реалізований у вигляді евристичної item-based логіки, де враховується співперетин уподобань різних користувачів [21]. Йдеться про фіксацію таких пар товарів, які найчастіше з'являються разом у вподобаннях або оцінках багатьох користувачів. Власне матриця подібностей буде створена на основі обчислення співвідношення числа спільних вподобань до загальної кількості дій (1.3):

$$\text{sim}(i, j) = \frac{|U_i \cap U_j|}{\sqrt{|U_i| \cdot |U_j|}}, \tag{2.3}$$

де U_i – множина користувачів, що вподобали товар i. Такий підхід є обчислювально простим, не вимагає попереднього навчання, і підвищує релевантність рекомендацій у випадках, коли контентна подібність недостатня.

Фінальна формула для оцінки рекомендації містить поєднання двох джерел сигналу. Якщо S_{CBF} – оцінка подібності з контентної частини, а S_{CF} –

оцінка за колаборативною логікою, то об’єднаний рейтинг (1.4) визначається за формулою:

$$S_{core} = \alpha \cdot S_{CBF} + (1 - \alpha) \cdot S_{CF}, \quad (2.4)$$

де $\alpha \in [0,1]$ – вага контентної частини [22]. У межах цієї системи доцільним є встановлення параметра α = 0,6, оскільки контентна частина є більш надійною при обмеженій кількості дій користувача.

Для наочності роботи механізму формування персоналізованих рекомендацій на основі гібридної моделі доцільно розглянути блок-схему, яка відображає основні етапи обробки даних та прийняття рішення системою. Вона демонструє логіку, що поєднує аналіз вмісту товарів з обліком поведінкових патернів користувача. Ця схема представлена на рисунку 2.1.

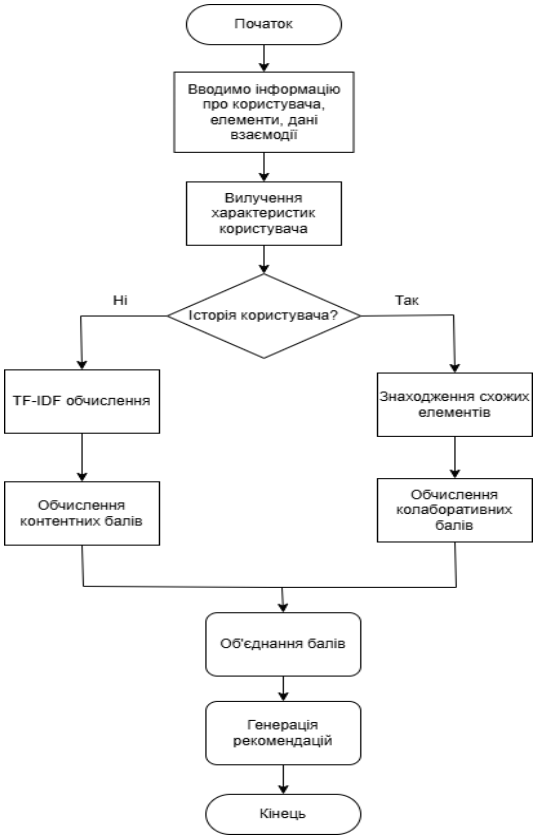

Рисунок 2.1 – Алгоритм формування рекомендацій у гібридній системі

Як видно з рисунка, система функціонує в кілька послідовних етапів. Спочатку відбувається зчитування інформації про користувача, його попередні дії та ознаки товарів. Далі визначається наявність історії взаємодії користувача з системою: якщо вона відсутня, ініціюється процес контентного аналізу з використанням TF-IDF; якщо історія є, активується механізм виявлення подібних товарів або дій інших користувачів. Насамкінець обидва типи результатів комбінуються з урахуванням вагових коефіцієнтів, після чого формується остаточний список персоналізованих рекомендацій. Завдяки цьому підходу система може легко адаптуватися до збільшення кількості користувачів і товарів, ефективно працювати навіть при обмеженій історії взаємодій, а також формувати релевантні рекомендації для нових товарів, які ще не мають відгуків або оцінок.

2.2 Розробка загальної клієнт-серверної архітектури та структурних модулів системи

Формування ефективної програмної системи рекомендацій неможливе без розробки чіткої архітектури, яка б дозволяла організувати взаємодію всіх її компонентів. У сучасних веборієнтованих середовищах найбільш поширеним є клієнт-серверний підхід, що передбачає розподіл логіки між клієнтською частиною, яка відповідає за інтерфейс і первинну обробку взаємодії, та серверною частиною, яка реалізує бізнес-логіку, рекомендаційні алгоритми та роботу з базою даних. Така архітектура забезпечує гнучкість, масштабованість, можливість централізованого оновлення та незалежний розвиток окремих модулів без порушення цілісності всієї системи.

Побудова рекомендаційної системи, орієнтованої на персоналізований відбір товарів, вимагає розділення на низку логічних блоків, кожен з яких відповідає за окрему групу функцій. Серед них основоположними є модуль автентифікації, який реалізує створення облікового запису, вхід користувача, захист доступу; модуль управління профілем, відповідальний за збір та

зберігання персональних характеристик; каталог товарів, який дозволяє здійснювати пошук, фільтрацію, перегляд детальної інформації; блок зберігання історії дій, де фіксуються перегляди, вподобання, оцінки та покупки; нарешті, ядро рекомендаційної системи, яке аналізує накопичену інформацію й формує персональні підказки.

Клієнтська частина системи виконує функцію взаємодії користувача з графічним інтерфейсом, надсилає запити до серверної частини, отримує у відповідь сформовані результати та відображає їх в адаптованому вигляді. Сучасні технології дозволяють реалізувати цю частину за допомогою JavaScript-фреймворків, однак у межах цього проєкту вона буде реалізована у спрощеному варіанті – у вигляді HTML-інтерфейсу або Flask-шаблонів, що передбачає прямий рендеринг на сервері. Такий підхід зменшує складність реалізації, проте не знижує загальної функціональності.

Серверна частина системи реалізує ключові функції: перевірку сесії, обробку форм реєстрації та входу, зберігання та оновлення даних користувача, а також запуск механізму рекомендацій. Для цього сервер має доступ до бази даних, у якій зберігаються всі необхідні таблиці: користувачі, товари, оцінки, перегляди, уподобання, згенеровані рекомендації. Сервер обробляє запити згідно з REST-логікою: кожен запит має визначений маршрут, метод (GET, POST, PUT) і відповідний обробник.

На рівні внутрішньої логіки система має бути розділена на модулі. Кожен модуль виконує ізольовану функцію, що підвищує підтримуваність і дозволяє розробляти окремі компоненти паралельно. Наприклад, модуль рекомендацій містить реалізацію TF-IDF-обробки, матрицю схожості, алгоритм комбінування ваг; модуль фільтрації – пошук товарів за категорією чи рейтингом; модуль профілю – оновлення особистих даних, вподобань, результатів взаємодії. Такий підхід відповідає принципам MVC-моделі, де контролер обробляє логіку, модель взаємодіє з базою даних, а шаблон відповідає за відображення результатів.

Зважаючи на наведене, загальна архітектура системи формалізується як тришарова клієнт-серверна структура з внутрішнім модульним поділом на функціональні компоненти. Це дозволяє забезпечити простоту обслуговування, зрозумілу логіку розгортання та можливість майбутнього масштабування.

Для кращого розуміння внутрішньої організації компонентів системи та їх взаємодії нижче наведено узагальнену схему клієнт-серверної архітектури системи рекомендацій (рис.2.2).

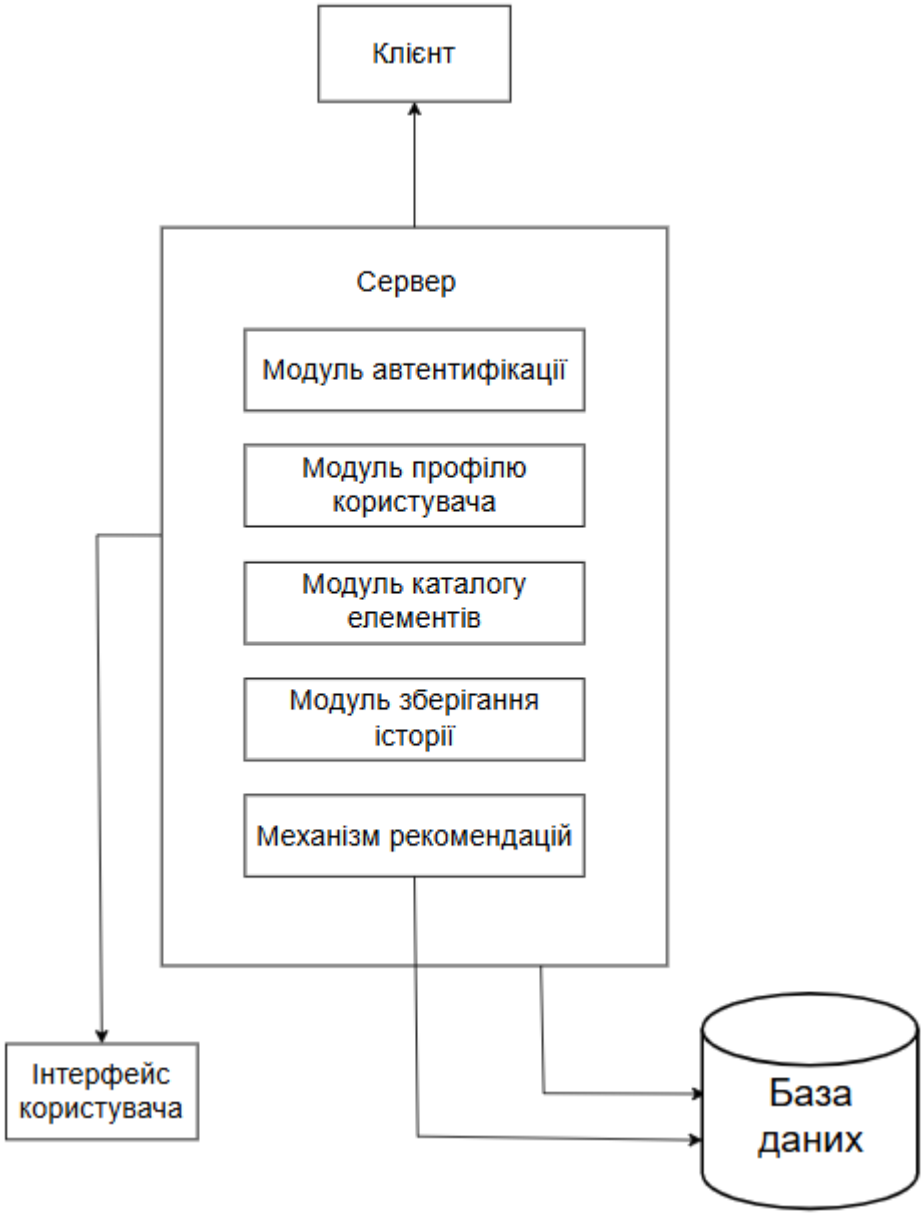

Рисунок 2.2 – Структура клієнт-серверної архітектури системи рекомендацій

З наведеної схеми видно, що центральним елементом є сервер, на якому зосереджено ключові логічні модулі системи. До них належать блок автентифікації, модуль управління профілем користувача, каталог товарів, модуль збереження історії взаємодії, а також ядро рекомендаційної системи. Клієнт взаємодіє із сервером через інтерфейс, отримуючи відповіді у вигляді сформованих рекомендацій або результатів пошуку. Усі дані зберігаються у централізованій базі даних, яка забезпечує стійке збереження інформації й доступ до неї в реальному часі. Така архітектура гарантує модульність, масштабованість і можливість подальшого розвитку системи.

2.3 Розробка алгоритмів авторизації, збереження профілю та історії взаємодії користувача

У будь-якій інформаційній системі, що працює з персоналізованим вмістом, модуль обліку користувача відіграє базову роль. Саме цей компонент забезпечує не лише автентифікацію доступу, а й збір, зберігання та обробку ключових характеристик, які формують основу для подальших дій системи: від побудови профілю до генерування рекомендацій. У системах, що базуються на принципах персоналізації, значення модуля авторизації, профілювання та збереження історії дій неможливо переоцінити. Цей компонент, хоча й не є частиною рекомендаційного алгоритму безпосередньо, забезпечує йому релевантне інформаційне підґрунтя.

Розробка механізмів автентифікації користувачів передбачає впровадження стандартного протоколу створення облікового запису з валідацією унікальності логіна або адреси електронної пошти, захищеного зберігання пароля (за допомогою хешування, наприклад, через алгоритм bcrypt або hashlib.pbkdf2_hmac), а також сесійної ідентифікації через токени або cookie. В контексті системи, реалізованої на Python з використанням фреймворку Flask, доцільним є застосування бібліотеки Flask-Login або власного рішення на основі JWT (JSON Web Tokens), яке дозволяє

підтримувати безпечну та масштабовану авторизацію без потреби у серверному збереженні сесій.

Після реєстрації користувач повинен мати можливість увійти в систему з будь-якого пристрою, що передбачає реалізацію механізмів автентифікації, які перевіряють відповідність введених даних до записів у базі даних, та створення сесії доступу. Відповідно до політики безпеки, токени доступу повинні мати обмежений строк дії та перегенеровуватись у разі повторного входу. Кожна сесія повинна логуватись у внутрішній системі моніторингу, що дозволяє запобігати несанкціонованому доступу та вести журнал активності.

Наступним кроком є розробка структури профілю користувача. Профіль виконує роль не лише сховища основних ідентифікаційних даних, таких як ім’я, вік, стать, мова інтерфейсу, а й служить підґрунтям для формування векторного представлення користувача у системі. Це представлення може базуватись як на вручну введених параметрах (наприклад, вказані інтереси), так і на результатах аналізу динамічної поведінки – історії переглядів, пошуків, оцінок. У реалізації системи доцільно зберігати профіль користувача у вигляді JSON-структури, що дозволяє легко адаптуватися до змін у структурі даних і додавати нові поля без потреби реорганізації бази.

Важливо зазначити, що профіль користувача не є статичним. Система повинна підтримувати механізм його динамічного оновлення. Наприклад, якщо користувач кілька разів переглядає товари з певної категорії, ця інформація автоматично повинна впливати на вагу відповідного параметра у векторі вподобань. Таким чином, профіль поступово адаптується до реальних інтересів користувача, навіть якщо спочатку вони були вказані неточно або взагалі не були задані.

Окреме місце у структурі системи займає журнал історії взаємодії користувача. Цей компонент фіксує кожну значущу дію користувача: перегляд сторінки товару, натискання на кнопку «додати до вподобаних», оцінку (у вигляді рейтингу або лайку), здійснення покупки, коментар, пошуковий запит тощо. Дані такого журналу мають бути максимально деталізованими,

включаючи дату і час події, ідентифікатор користувача, ідентифікатор товару, тип дії, а також технічні параметри – IP-адресу, тип пристрою, канал взаємодії (мобільний додаток, веб, API). Усе це дозволяє згодом не лише будувати більш точні рекомендації, а й проводити глибокий аналіз поведінки користувачів.

Для забезпечення ефективної обробки великої кількості подій журнал історії повинен бути реалізований на рівні окремої таблиці бази даних із відповідною індексацією. Крім того, доцільно реалізувати механізм агрегування – тобто перетворення «сирих» подій у агреговані сигнали, що підсумовують частоту, тривалість або інтенсивність певного типу дій по відношенню до кожного товару. Наприклад, якщо користувач переглядав товар тричі протягом двох днів, це може бути інтерпретовано як підвищений інтерес і використано для підсилення релевантності товару в рекомендаційній видачі.

Історія дій також є ключовим елементом для генерації поведінкових метрик, які формують ядро користувацького профілю в системі колаборативної фільтрації. Власне ці дані використовуються для побудови векторів користувачів у просторі подібності, де основою слугують спільні оцінки або перетин уподобань. Таким чином, історія взаємодії – це не лише «логування», а фактично джерело інтелекту для системи, що дозволяє їй «вчитися» з поведінки реальних людей.

Ще одним важливим моментом є синхронізація між модулями: авторизації, профілю та історії. Наприклад, після входу користувача система одразу має оновити його останню активність, перевірити зміни у профілі, ініціювати оновлення персонального рейтингу товарів. Для цього реалізовується так званий middleware – проміжний рівень, що обробляє взаємодію між запитами, базою та логікою рекомендацій.

У структурному вигляді взаємодія між цими модулями представлена у вигляді схеми, яка демонструє послідовність дій: від введення логіна до збереження оцінки чи перегляду (рис.2.3).

Як видно зі схеми, процес розпочинається з введення облікових даних. У разі успішної перевірки облікових даних виконується автентифікація,

оновлення профілю та подальша реєстрація дій користувача в системі. Кожна взаємодія – оцінка, перегляд або будь-яка інша форма активності – фіксується в окремому модулі історії, який забезпечує подальше використання цих даних для формування персоналізованих рекомендацій. Підхід гарантує безперервну синхронізацію між модулями та повну адаптацію системи до поточної поведінки користувача.

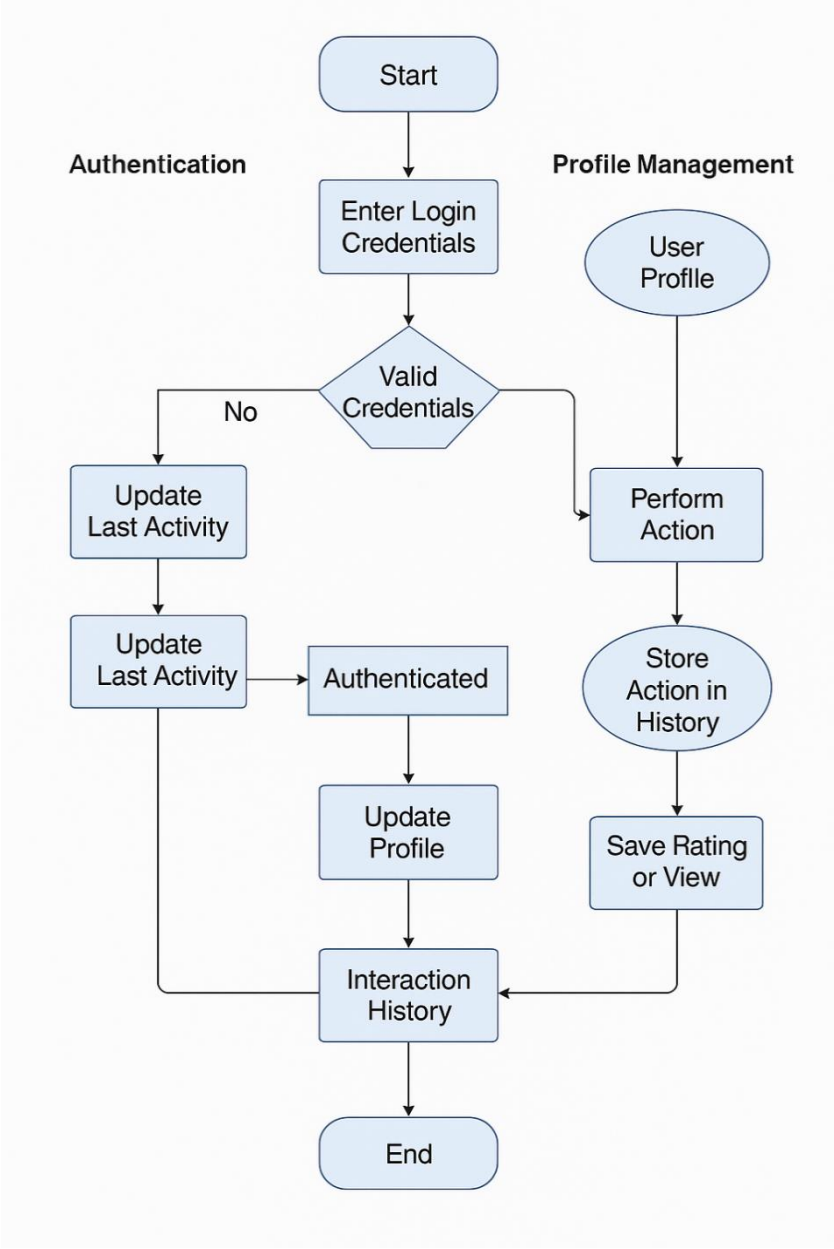

Рисунок 2.3 – Схема взаємодії модулів авторизації, профілю та історії користувача

Такий підхід гарантує безперервну синхронізацію між модулями та повну адаптацію системи до поточної поведінки користувача.

2.4 Розробка структури каталогу товарів і алгоритмів пошуку та фільтрації

Функціонування будь-якої системи електронної торгівлі або контентної платформи неможливе без чітко організованого механізму представлення товарів. Саме каталог виступає структурним елементом, що забезпечує організацію, пошук, фільтрацію, сортування й відображення одиниць продукції. Для системи, що реалізує рекомендаційний модуль, каталог має додаткову функцію – виступати джерелом ознак для формування векторного представлення товарів у контентному аналізі. Таким чином, правильне моделювання структури каталогу товарів і логіки доступу до нього є критичним не лише з погляду юзабіліті, а й для забезпечення точності рекомендаційного алгоритму.

Структура каталогу в нашому випадку будується навколо базової сутності – товару. Кожен товар повинен мати унікальний ідентифікатор, назву, короткий і повний опис, категорію, підкатегорію, рейтинг, ціну, а також метаінформацію, таку як дата додавання, статус наявності, кількість переглядів. Крім того, кожна товарна одиниця може бути зв’язана з візуальним зображенням, що вимагає окремої таблиці з відносним посиланням або URL. З технічної точки зору доцільно реалізовувати цю структуру у вигляді реляційної моделі, де основна таблиця Items містить ключову інформацію, а пов’язані таблиці (наприклад, Categories, Images, Tags) забезпечують гнучкість і нормалізацію даних.

Для реалізації пошуку необхідно передбачити індексацію ключових полів, зокрема name, description, category_id. Базовий пошук за назвою може бути реалізований за допомогою оператора LIKE або ILIKE у SQL, однак для підвищення продуктивності та якості результатів доцільно застосовувати механізми повнотекстового пошуку. Наприклад, використання PostgreSQL

дозволяє створити tsvector-індекси, які забезпечують швидкий і точний пошук за великими текстовими полями, включаючи морфологічний аналіз і ранжування результатів.

Фільтрація є іншим ключовим механізмом, який дозволяє користувачеві обмежити вибірку товарів згідно із заданими критеріями. У типових сценаріях користувач фільтрує товари за категорією, діапазоном цін, рейтингом, новизною або популярністю. З боку бази даних це вимагає реалізації запитів із динамічно змінними умовами, що генеруються інтерфейсом. Такі запити повинні бути оптимізовані, наприклад, шляхом попередньої агрегації даних або кешування результатів для популярних комбінацій фільтрів. При реалізації механізму фільтрації важливо дотримуватись принципу комутабельності, що означає можливість застосування фільтрів у будь-якому порядку без втрати результатів або їх некоректного сортування.

Категоризація товарів забезпечує структуризацію каталогу та дозволяє створювати ієрархічну навігацію. У реалізації системи необхідно передбачити не лише наявність категорії як атрибута, а й підтримку вкладених категорій через рекурсивну структуру або модель батько-дитина. Наприклад, категорія «Книги» може містити підкатегорії «Фантастика», «Бізнес», «Навчальна література» тощо. Це дозволяє будувати динамічне дерево категорій, що відображається у навігаційній панелі користувача та забезпечує інтуїтивне переміщення по розділах.

Окремо варто зазначити, що атрибути товарів, які використовуються в контентному аналізі, повинні бути стандартизовані. Наприклад, якщо поле description використовується для обчислення TF-IDF, то опис має бути очищений від HTML-тегів, спеціальних символів, та приведений до нижнього регістру. Аналогічно – категорії мають бути представлені у вигляді лейблів або індексів, які можна легко перетворити у числові вектори або one-hot-представлення. Це дозволяє зменшити похибки у процесі векторизації та підвищити точність рекомендацій.

Також система повинна підтримувати сортування результатів пошуку або фільтрації за заданим критерієм: за рейтингом, ціною, датою додавання. Для цього сервер повинен мати можливість обробляти параметри сортування, які задаються користувачем у запиті, та динамічно формувати SQL-конструкцію ORDER BY. У поєднанні з пагінацією це дозволяє підтримувати високу продуктивність і зручність інтерфейсу навіть за великої кількості товарів у каталозі.

З технічного погляду логіка пошуку і фільтрації має бути інкапсульована у відповідному модулі. У Python-реалізації це може бути окремий клас або сервіс, який приймає параметри від клієнта, формує запит до бази даних, обробляє результати (наприклад, форматує значення цін, округлює рейтинг, додає зображення) та повертає підготовлений об’єкт JSON. Таким чином, забезпечується логічне розділення між рівнем представлення і рівнем бізнес-логіки.

На рисунку 2.4 представлена узагальнена ER-схему, яка візуально демонструє структуру таблиць Items, Categories, Ratings, Tags, Images, а також ключі зв’язку між ними.

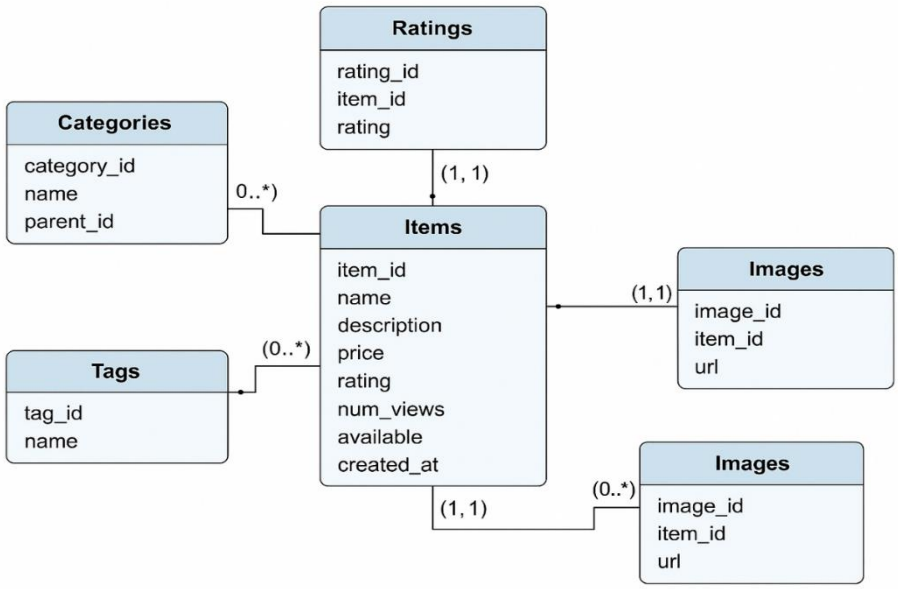

Рисунок 2.4 – Схема структури даних каталогу товарів

На схемі центральне місце займає таблиця Items, яка містить опис товару, включаючи назву, опис, ціну, рейтинг, кількість переглядів та статус доступності. Зв’язки з іншими таблицями дозволяють організувати ієрархію категорій, приєднувати зображення, призначати теги та зберігати оцінки.

Завдяки такій структурі забезпечується не лише ефективна фільтрація та пошук, а й можливість побудови точних векторних представлень товарів для контентного аналізу. Це є важливою передумовою для коректного функціонування рекомендаційної системи.

2.5 Побудова функціональної схеми рекомендаційної системи

Після розробки окремих компонентів рекомендаційної системи – алгоритмічного ядра, архітектури клієнт-серверної взаємодії, модуля автентифікації, історії взаємодії, каталогу товарів та алгоритмів пошуку – необхідним є створення функціональної моделі, яка би узагальнювала всі ці елементи в єдину цілісну структуру. Така схема дозволяє не лише візуалізувати логіку роботи системи, але й забезпечує на етапі реалізації правильне узгодження компонентів, мінімізацію дублювання функцій і контроль за потоками даних. Функціональна схема розкриває системну організацію програмного продукту та визначає, як саме інформація переміщується між його частинами.

Концептуальна функціональна модель охоплює весь життєвий цикл взаємодії користувача з системою: від входу до сервісу до формування та перегляду персоналізованих рекомендацій. У самому початку користувач звертається до інтерфейсу програми, який у спрощеному випадку реалізований у вигляді HTML-сторінки або шаблону з підтримкою форм. Цей інтерфейс дозволяє користувачу виконати вхід або зареєструватися. У разі коректної автентифікації ініціюється створення сесії, що включає передачу унікального токена, який супроводжує подальші запити.

Після входу користувач отримує доступ до головної сторінки каталогу. На цьому етапі клієнтський модуль ініціює запит на отримання списку товарів. Запит передається на сервер, де спеціалізований модуль фільтрації формує відповідь, враховуючи активні параметри сортування, категорії, рейтингу тощо. Якщо користувач обирає конкретний товар для перегляду, генерується детальна сторінка, де крім опису товару можуть бути запропоновані пов'язані або подібні товари – або на основі контентної схожості, або на основі оцінок інших користувачів.

Ключовим елементом у цій взаємодії є механізм збору поведінкових даних. Кожен перегляд, оцінка або вподобання, як і додавання товару в кошик чи завершення покупки, фіксується в окремому журналі. Цей журнал історії взаємодії є не лише інструментом логування, а й живим джерелом даних для модуля формування рекомендацій. Саме на основі цієї інформації модуль рекомендацій періодично генерує оновлені списки товарів, які можуть бути актуальними для конкретного користувача. Залежно від частоти активності та обсягу дій, ці списки можуть формуватися в реальному часі або бути збереженими у тимчасовому кеші та оновлюватися раз на певний період.

Формування рекомендаційного списку є результатом узгодженої роботи декількох підсистем. З одного боку, аналізується профіль користувача: його інтереси, вподобання, структура взаємодії, частотність переглядів. З іншого – виконується векторний аналіз товарів на основі TF-IDF-представлення та обчислюється косинусна подібність до тих товарів, з якими користувач взаємодіяв найбільше. У паралельному потоці система здійснює облік схожих вподобань інших користувачів і додає рекомендації на основі колаборативної логіки. Обидва результати зважуються та агрегуються в єдиний список, який ранжується за відповідним коефіцієнтом релевантності.

Рекомендовані товари відображаються у персональній стрічці користувача, яка структурована на кілька блоків: «Рекомендовано для вас», «На основі переглядів», «Популярне», «Схожі на вподобані». Ці блоки також

формуються з урахуванням поточного стану користувача, активності за останні сеанси та загальних трендів у поведінці системи.

Функціональна схема відображає всі основні потоки даних між клієнтською частиною, модулями серверної логіки, базою даних та алгоритмами. Вона демонструє, як інформація передається, трансформується, зберігається та використовується для досягнення ключової мети – точного і релевантного формування персоналізованих рекомендацій. Завдяки такій схемі можна відслідковувати, на якому етапі виникає потенційне навантаження, які частини системи є критичними з точки зору продуктивності, а також які з модулів можуть бути оптимізовані або винесені в окремі сервіси (рис.2.5).

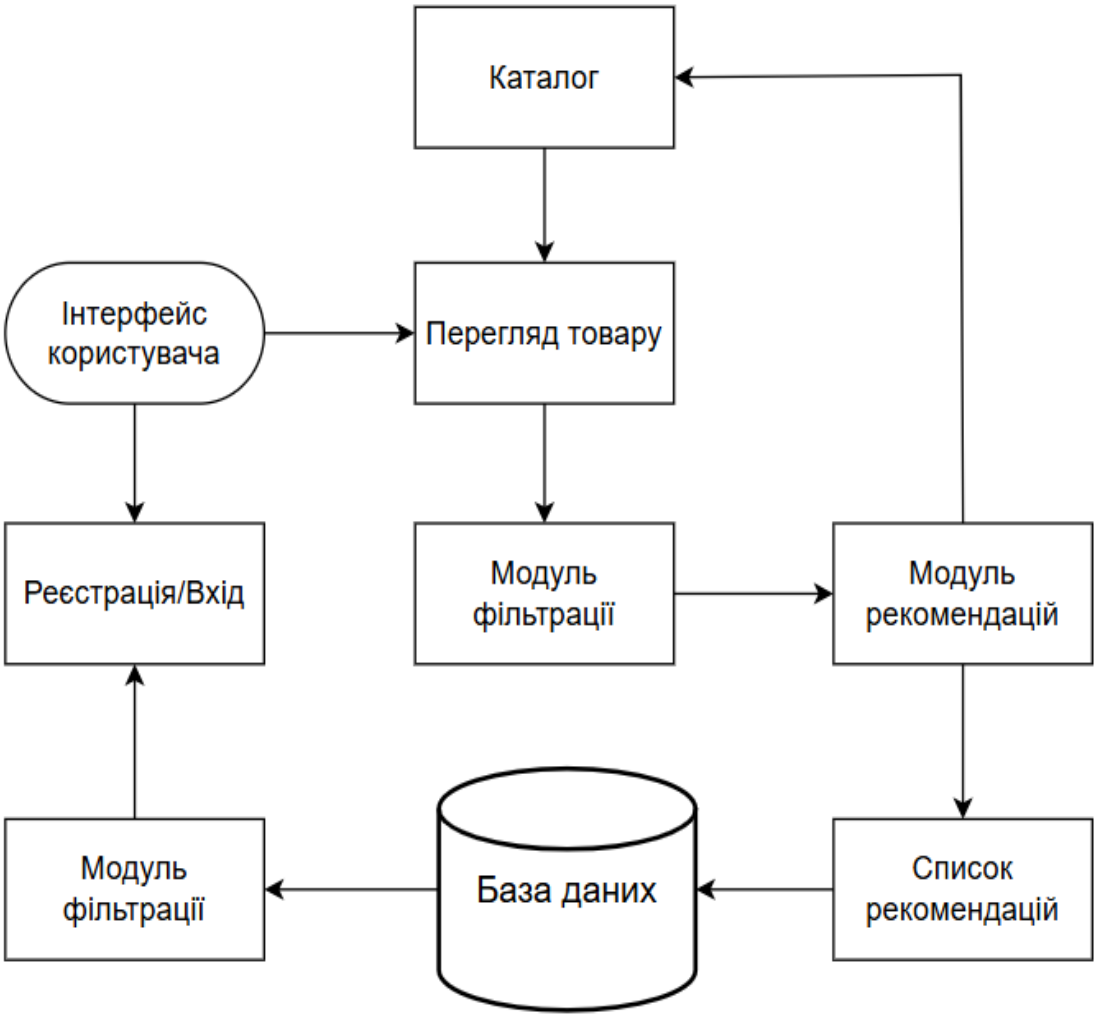

Рисунок 2.5 – Функціональна схема роботи рекомендаційної системи товарів

На схемі простежується повний цикл роботи системи. Користувач здійснює вхід або реєстрацію, після чого взаємодіє з каталогом товарів. Усі дії передаються через модуль фільтрації до бази даних, а також до рекомендаційного модуля, який формує перелік персоналізованих товарів. Результати повертаються на клієнтську частину у вигляді стрічки пропозицій, адаптованої до профілю та поведінки конкретного користувача. Така схема забезпечує узгоджену роботу всіх компонентів системи та дозволяє ефективно реалізувати персоналізацію в умовах динамічного оновлення даних.

2.6 Висновок до розділу 2

У цьому розділі було здійснено розробку архітектурно-алгоритмічної моделі рекомендаційної системи товарів, яка є основою для подальшої реалізації повноцінного програмного модуля. Послідовний виклад і технічне структурування всіх компонентів дозволило досягти логічної цілісності проєкту та забезпечити взаємозв’язок між окремими модулями в межах єдиної клієнт-серверної платформи.

На першому етапі було проведено глибокий аналіз існуючих математичних моделей формування рекомендацій, що дозволило обґрунтувати вибір гібридної моделі як найоптимальнішого рішення для умов обмежених даних та необхідності високої релевантності результатів. Поєднання контентної фільтрації (на основі TF-IDF і косинусної подібності) з елементами колаборативного аналізу забезпечує адаптивність системи до поведінкових змін користувача і надає можливість підвищення точності без ускладнення архітектури.

Розроблена клієнт-серверна архітектура з модульним поділом дозволила логічно впорядкувати всі компоненти системи: авторизацію, профілювання, роботу з історією, каталогом, пошуком, фільтрацією та генерацією рекомендацій. Така побудова є масштабованою, підтримуваною і придатною до

розширення, що особливо важливо в умовах швидкого розвитку бізнесу або зміни вимог з боку користувачів.

Особливу увагу приділено модулю авторизації та профілю користувача, оскільки саме на цьому рівні відбувається формування векторного представлення користувача, що є основою для всього подальшого аналізу. Також впроваджено механізм ведення історії взаємодії з системою, що не лише дозволяє відтворювати минулі дії, але й використовується як джерело даних для рекомендаційного ядра.

Було сформовано оптимальну структуру каталогу товарів, де кожна одиниця зберігає повний набір атрибутів, потрібних як для візуалізації, так і для векторизації під час контентного аналізу. Впроваджено логіку пошуку, фільтрації та сортування, що дає змогу користувачеві швидко знайти потрібний товар або звузити вибірку відповідно до власних уподобань.

На завершення було побудовано узагальнену функціональну схему рекомендаційної системи, яка наочно демонструє повний цикл взаємодії між усіма модулями та етапами роботи програми – від автентифікації до персоналізованої рекомендації. Така схема слугує не лише візуальним підсумком, а й технічним орієнтиром для подальшої реалізації коду.

Отже, у розділі 2 було запропоновано цілісну архітектурну й алгоритмічну основу, яка у наступному розділі буде використана для реалізації програмного модуля, проведення його тестування, аналізу результатів і визначення показників ефективності.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ТОВАРІВ

3.1 Обґрунтування вибору інструментальних засобів для реалізації програмного модуля

Процес реалізації рекомендаційної системи товарів передбачає використання широкого спектру інструментальних засобів, кожен з яких відповідає за певний аспект роботи системи: від побудови алгоритмів машинного навчання до реалізації веб-середовища та обробки запитів користувача. Вибір конкретних технологій має бути зумовлений як вимогами функціональності, так і характеристиками зручності розгортання, продуктивності, масштабованості та доступності супроводу.

У центрі реалізації системи знаходиться мова програмування Python, яка протягом останнього десятиліття стала стандартом де-факто для побудови інтелектуальних систем, прототипування та обробки даних. Її універсальність, багатий набір бібліотек, активна спільнота та висока інтеграційна здатність роблять Python найкращим вибором для реалізації як серверної логіки, так і компонентів штучного інтелекту в системі. Python дозволяє поєднати реалізацію класичної серверної архітектури з модулями обчислення подібності, фільтрації, обробки тексту, формування рекомендацій і паралельно реалізувати інтерфейс користувача навіть у рамках одного проєкту.

Для побудови веб-сервера та реалізації клієнт-серверної взаємодії було обрано Flask – мікрофреймворк для Python, який ідеально підходить для невеликих, швидких у реалізації проєктів із гнучкою архітектурою. Flask дозволяє організовувати маршрути обробки запитів, реалізовувати REST API, формувати HTML-сторінки через шаблони, зберігати сесії користувачів і працювати з базами даних через ORM або вручну. На відміну від важчих фреймворків, таких як Django, Flask не нав’язує жорстку структуру, що дає

розробнику повну свободу у формуванні логіки – саме тому він був обраний як оптимальний фреймворк для реалізації даної системи.

Ключову роль у реалізації рекомендаційного модуля відіграють алгоритми обробки текстових описів товарів та обчислення схожості між ними. Для цього використовується бібліотека scikit-learn, яка надає всі необхідні інструменти для побудови векторних представлень за допомогою TF-IDF, обчислення косинусної подібності, нормалізації даних та кластерного аналізу. TF-IDF реалізовано у вигляді окремого модуля TfidfVectorizer, що дозволяє швидко перетворювати масив описів товарів у векторний простір з подальшим порівнянням їх за заданою метрикою. Саме ця бібліотека є основою для побудови контентної частини рекомендаційного ядра.

Для зберігання даних у межах демонстраційної версії було реалізовано внутрішню пам’ять у вигляді словників Python, що дозволяє уникнути необхідності зовнішньої бази даних на початковому етапі. Такий підхід забезпечує швидкість розробки, простоту налагодження та мінімальні вимоги до розгортання системи. Однак у разі переходу до повноцінного проєктного середовища доцільно застосовувати реляційні бази даних – зокрема PostgreSQL або SQLite, які легко інтегруються з Python через бібліотеки sqlalchemy, psycopg2 або sqlite3.

Для створення користувацького інтерфейсу було використано стандартні HTML-шаблони на основі Flask-шаблонізації (render_template_string), що дозволило швидко реалізувати інтерфейс каталогу, перегляду товару, оцінювання та лайків. Такий підхід забезпечує повну інтеграцію логіки на сервері з відображенням на клієнті без необхідності в окремому фронтенд-фреймворку. У майбутньому систему можна легко адаптувати для роботи з такими бібліотеками як React, Vue або навіть простим Bootstrap, що дозволить забезпечити мобільність і масштабування інтерфейсу без зміни основного ядра.

Для збереження безпеки облікових записів користувачів при зберіганні паролів застосовано хешування за допомогою werkzeug.security, зокрема метод generate_password_hash, що реалізує безпечне зберігання без відкритого тексту.

Перевірка автентичності при вході реалізується через функцію check_password_hash, що дозволяє зберегти високий рівень безпеки навіть у разі втрати доступу до сервера. Сесії користувачів підтримуються через стандартний механізм Flask session, що дозволяє асоціювати кожну взаємодію з конкретним обліковим записом без необхідності вручну відслідковувати параметри доступу.

Щодо засобів тестування, то основна перевірка функціональності реалізується через API-запити, які можна надсилати за допомогою Postman або CURL. Оскільки всі основні функції системи – реєстрація, вхід, перегляд, вподобання, оцінювання, генерація рекомендацій – реалізовані у вигляді окремих маршрутів, їх легко протестувати як із браузера, так і з використанням інструментів автоматизованого тестування. У наступних підрозділах будуть наведені приклади запитів та відповідей системи, що підтверджують працездатність кожного функціонального модуля.

Вибір зазначених інструментальних засобів обумовлений не лише їхньою сумісністю, але й логікою швидкого прототипування, яка дозволяє за обмежений час побудувати повноцінний функціональний прототип рекомендаційної системи з можливістю подальшого масштабування, перенесення на промислові бази даних та розширення логіки рекомендацій за рахунок гібридних алгоритмів, машинного навчання або глибоких нейронних моделей.

Комбінація Python, Flask, scikit-learn, HTML-шаблонізації та базових засобів обробки запитів є оптимальним рішенням для реалізації рекомендаційної системи у рамках цього проєкту. Вона забезпечує повну підтримку всіх вимог технічного завдання, гарантує легкість розгортання та модифікації, а також закладає основу для подальших експериментів із точністю, ефективністю та зручністю користування.

3.2 Реалізація модуля рекомендаційної системи товарів

Розробка модуля рекомендаційної системи є ключовим етапом у побудові програмного рішення, яке здатне динамічно адаптуватися до потреб користувача, враховувати історію його дій, обробляти вподобання й надавати максимально релевантні пропозиції. У межах цього підрозділу буде детально описано логіку реалізації всіх частин функціоналу, пов’язаних із формуванням персоналізованих рекомендацій: від структури даних і обробки вподобань до побудови рекомендаційного списку на основі гібридного алгоритму контентної подібності.

На етапі архітектурного проєктування було ухвалено рішення реалізувати рекомендаційний модуль у межах єдиного Flask-сервера, використовуючи мову Python. Це дозволило об’єднати взаємодію з клієнтом, логіку оцінювання, переглядів, а також реалізацію власне алгоритмів рекомендацій у єдиному середовищі. Уся логіка зберігається у вигляді структурованого коду, який легко підтримується та масштабовується.

Основою для побудови рекомендацій став контентно-орієнтований підхід. В його основі – аналіз описів товарів, представлених у вигляді векторів TF-IDF. Застосування цього методу дозволяє зручно перетворювати текст у числову форму, що дає змогу обчислювати відстані та подібність між товарами. У проєкті було використано реалізацію TF-IDF із бібліотеки scikit-learn, яка дає змогу швидко побудувати векторний простір для всієї множини описів.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

tfidf = TfidfVectorizer()
descriptions = [item['description'] for item in items]
tfidf_matrix = tfidf.fit_transform(descriptions)
```

Після побудови векторного простору потрібно визначити ті товари, які найбільш схожі на вподобані користувачем. Для цього, на основі індексів товарів, що були позначені як вподобані (likes), вибираються відповідні вектори, для яких обчислюється косинусна подібність до всіх інших товарів:

```
liked_indexes = [i for i, item in enumerate(items) if item['id'] in user['likes']]
sim_scores = np.mean([cosine_similarity(tfidf_matrix[i], tfidf_matrix).flatten() for i in liked_indexes], axis=0)
```

Після обчислення подібності товари сортуються у порядку спадання подібності. Із результатів виключаються ті товари, які вже були оцінені або вподобані користувачем, щоб уникнути повторень у стрічці рекомендацій:

```
top_indexes = np.argsort(sim_scores)[::-1]
recommended = [items[i] for i in top_indexes if items[i]['id'] not in user['likes']][:5]
```

Ці товари повертаються у вигляді персоналізованого списку:

```
@app.route('/recommend', methods=['GET'])
def recommend():
    user_email = session.get('user')
    if not user_email:
        return jsonify({'error': 'Необхідно увійти'}), 401
    user = users[user_email]
    # TF-IDF + косинус
    ...
    return jsonify(recommended)
```

Перевагою цього підходу є його швидкодія та незалежність від великої історії дій. Достатньо лише текстового опису товару та базових взаємодій користувача, щоб система могла сформувати пропозиції. Такий підхід особливо

корисний у нових або слабо заповнених системах, де ще відсутні об’ємні масиви поведінкових даних. Це дозволяє реалізувати рекомендації навіть для нового користувача після декількох дій.

Крім текстового опису, у модулі враховуються такі параметри як рейтинг товару, кількість переглядів (що свідчить про його популярність), а також категорія товару. Окремі функції дають змогу фільтрувати рекомендації за категоріями, що створює ефект контекстної адаптації рекомендацій. Наприклад, якщо користувач активніше взаємодіє з товарами категорії «Курси», система пріоритетно відображатиме подібні до них товари з того ж сегмента.

Далі було реалізовано всі інші моменти згідно функціональних вимог.

1. Реєстрація та авторизація користувача

Основою персоналізованої взаємодії є можливість створення унікального облікового запису. Реалізація модуля реєстрації передбачає обробку HTTP-запиту від користувача, який передає свою адресу електронної пошти, пароль і базову інформацію: вік, стать та опціональні інтереси.

Пароль зберігається у захищеному вигляді – з використанням хешування за допомогою generate_password_hash. Таким чином, навіть у разі компрометації системи справжній пароль залишається недоступним.

```
users[data['email']] = {
    'id': str(uuid.uuid4()),
    'password': generate_password_hash(data['password']),
    'age': data.get('age'),
    'gender': data.get('gender'),
    'interests': data.get('interests', ''),
    'likes': [],
    'ratings': {},
    'history': []
}
```

Для автентифікації реалізовано окремий маршрут, що виконує перевірку даних і створює сесію, зберігаючи стан авторизованого користувача:

```
@app.route('/login', methods=['POST'])
def login():
    ...
    if not user or not check_password_hash(user['password'], data['password']):
        return jsonify({'error': 'Невірні облікові дані'}), 401
    session['user'] = data['email']
```

Таким чином, кожна подальша взаємодія пов'язана з конкретним користувачем через механізм Flask session.

2. Каталог товарів і фільтрація

Модуль каталогу дозволяє виводити список товарів, реалізуючи при цьому фільтрацію за категорією. Всі товари зберігаються у структурованому списку словників, кожен з яких має опис, категорію, рейтинг і кількість переглядів:

```
items = [
    {'id': 1, 'name': 'Python для початківців', 'description': '...', 'category': 'Книги', 'rating': 4.5, 'views': 120},
    ...
]
```

Виведення каталогу реалізовано через маршрут /items, де за бажанням можна передати параметр category:

```
@app.route('/items', methods=['GET'])
def list_items():
    category = request.args.get('category')
```

```
        filtered = [item for item in items if category is None or
item['category'] == category]
        return jsonify(filtered)
```

Цей механізм дозволяє будувати адаптивні інтерфейси з динамічною фільтрацією товарів, що є критично важливим при великій кількості елементів.

3. Перегляд товарів, історія та популярність

Після вибору товару користувач отримує його повний опис. Одночасно з цим система фіксує дію користувача – перегляд:

```
@app.route('/view/<int:item_id>', methods=['GET'])
def view_item(item_id):
    user_email = session.get('user')
    if user_email:
        users[user_email]['history'].append(item_id)
        ...
```

Перегляди впливають на показники популярності товару – змінюється значення поля views. Таким чином, система накопичує дані для блоку рекомендацій «Популярне» або аналітики найчастіше переглянутих позицій.

4. Система вподобань (лайків)

Важливим джерелом сигналів є лайки – позитивна оцінка користувача. Вони фіксуються на рівні окремого масиву likes:

```
@app.route('/like/<int:item_id>', methods=['POST'])
def like_item(item_id):
    ...
    users[user_email]['likes'].append(item_id)
```

Ці лайки безпосередньо використовуються в механізмі рекомендацій на основі контентної подібності, оскільки дозволяють виділити ті товари, на які користувач відреагував явно.

5. Оцінювання товарів (рейтинги)

Рейтинги (від 1 до 5 зірок) надають числову інформацію про рівень задоволення користувача. Збереження відбувається у вигляді словника з ID товару як ключем і значенням рейтингу:

```
@app.route('/rate/<int:item_id>', methods=['POST'])
def rate_item(item_id):
    ...
    users[user_email]['ratings'][item_id] = rating
```

Рейтинги можуть бути використані в майбутньому для побудови алгоритмів колаборативної фільтрації (наприклад, матричної факторизації або user-based підходів).

6. Модуль рекомендацій на основі TF-IDF

Ключова інтелектуальна складова системи – модуль рекомендацій. Його логіка полягає в побудові векторів описів товарів через TF-IDF і пошуку подібних товарів до вподобаних користувачем:

```
tfidf = TfidfVectorizer()
descriptions = [item['description'] for item in items]
tfidf_matrix = tfidf.fit_transform(descriptions)

liked_indexes = [i for i, item in enumerate(items) if item['id'] in user['likes']]
sim_scores = np.mean([cosine_similarity(tfidf_matrix[i], tfidf_matrix).flatten() for i in liked_indexes], axis=0)
top_indexes = np.argsort(sim_scores)[::-1]
recommended = [items[i] for i in top_indexes if items[i]['id'] not in user['likes']][:5]
```

Цей підхід дозволяє будувати персональні рекомендації навіть при мінімальній історії користувача.

7. Генерація рекомендаційного списку

Готовий список передається користувачу через маршрут /recommend, при цьому система динамічно враховує найсвіжішу поведінку:

```
@app.route('/recommend', methods=['GET'])
def recommend():
    ...
    return jsonify(recommended)
```

Це забезпечує реальну адаптивність системи – нові вподобання чи оцінки одразу враховуються в наступній рекомендації.

Слід підкреслити, що побудований модуль забезпечує усі основні функції рекомендаційної системи: побудову профілю користувача, обробку вподобань, векторизацію товарів, обчислення подібності, ранжування, фільтрацію та формування видачі. Архітектура коду дозволяє масштабувати модель, розширювати її новими алгоритмами, інтегрувати з базами даних або зовнішніми API. Таким чином, розроблений програмний модуль є не лише готовим прототипом, а й платформою для подальшого дослідження ефективності різних підходів до персоналізації в електронній комерції.

А тепер розглянемо обробку зворотнього зв’язку, оскільки вона є дуже важливим елементом сучасних рекомендаційних систем. Обробка зворотнього зв’язку дозволяє системі не лише реагувати на поточні дії користувача, а й адаптувати модель рекомендацій відповідно до його змінюваних інтересів, підвищувати релевантність рекомендацій і коригувати логіку відбору контенту. У реалізованому програмному рішенні основними формами зворотного зв’язку є вподобання товарів (лайки), оцінювання (рейтинги від 1 до 5) та перегляди. Кожен з цих сигналів фіксується в момент взаємодії з користувачем і

зберігається у відповідних структурах, що дозволяє не тільки формувати рекомендації, а й виконувати подальший аналіз ефективності взаємодії.

Основна мета цього модуля полягає у формуванні бази поведінкових даних, яка надалі слугує вхідним джерелом для рекомендаційного ядра. На практиці кожен користувач асоціюється з окремим об’єктом у пам’яті системи, в якому зберігаються всі дії. У прикладній реалізації це виглядає як словник:

```
users = {
    'example@email.com': {
        'likes': [1, 2],
        'ratings': {1: 5, 2: 4},
        'history': [1, 2, 3]
    }
}
```

Вподобання (лайки)

Функція вподобання реалізована через окремий маршрут /like/<item_id>, що дозволяє фіксувати позитивну реакцію користувача на товар. У системі лайки зберігаються як список ідентифікаторів товарів. Реалізація виглядає наступним чином:

```
@app.route('/like/<int:item_id>', methods=['POST'])
def like_item(item_id):
    user_email = session.get('user')
    if user_email:
        users[user_email]['likes'].append(item_id)
        return jsonify({'message': 'Товар вподобано'})
    return jsonify({'error': 'Необхідно увійти'}), 401
```

Цей механізм дає змогу швидко ідентифікувати вподобані товари, що надалі впливають на формування списку рекомендацій за контентною або

колаборативною логікою. Наприклад, у модулі рекомендацій саме лайки формують ядро подібності у TF-IDF-аналізі.

Оцінювання (рейтинги)

Оцінки – це більш точна форма зворотного зв'язку. Вони дозволяють системі не просто фіксувати позитивний факт взаємодії, а й встановлювати її інтенсивність. Для збереження рейтингу реалізовано маршрут /rate/<item_id>:

```
@app.route('/rate/<int:item_id>', methods=['POST'])
def rate_item(item_id):
    user_email = session.get('user')
    rating = request.json.get('rating')
    if user_email:
        users[user_email]['ratings'][item_id] = rating
        return jsonify({'message': 'Оцінку збережено'})
    return jsonify({'error': 'Необхідно увійти'}), 401
```

Рейтинги зберігаються у вигляді словника, де ключем виступає ID товару, а значенням – числова оцінка. Надалі ці значення можуть бути використані для побудови матриці користувач-товар і застосування колаборативної фільтрації за допомогою алгоритмів SVD або KNN.

Перегляди (історія дій)

Ще одним типом зворотного зв'язку є перегляд товару. Він не є явним позитивним сигналом, проте свідчить про зацікавленість. Фіксація перегляду відбувається автоматично при виклику маршруту /view/<item_id>:

```
@app.route('/view/<int:item_id>', methods=['GET'])
def view_item(item_id):
    user_email = session.get('user')
    if user_email:
        users[user_email]['history'].append(item_id)
        for item in items:
            if item['id'] == item_id:
```

```
                    item['views'] += 1
        return jsonify(next((item for item in items if item['id'] ==
item_id), {}))
```

Завдяки накопиченню історії переглядів, система отримує змогу виводити товари на основі останніх інтересів. Також перегляди є джерелом інформації для популярності товарів, яка використовується при сортуванні у видачі.

Потенціал розширення: коментарі

Наразі система не підтримує збереження текстових коментарів, однак архітектура дозволяє легко реалізувати таку функцію. Для цього потрібно додати до кожного товару список словників із текстами і метаданими (дата, автор), а також створити відповідний маршрут. Це дасть змогу розширити спектр інтерпретованого зворотного зв’язку та побудувати елементи семантичного аналізу тексту, який може слугувати додатковим джерелом для рекомендацій.

Сумарна інтеграція фідбеку в систему. Уся реалізація алгоритмів обробки зворотного зв’язку функціонує як єдиний модуль. Кожна взаємодія користувача – незалежно від типу – потрапляє до системи у реальному часі та зберігається у структурі профілю. Завдяки цьому, при кожному новому запиті до /recommend, система формує оновлений список з урахуванням усіх змін. Це забезпечує динамічність, гнучкість і адаптивність системи.

Модуль фідбеку в реалізованій системі забезпечує повний цикл збору поведінкових даних, які використовуються для персоналізації видачі, аналітики дій, побудови профілю користувача та подальшого розвитку рекомендаційного механізму. Його розширення через коментарі або автоматичний аналіз часу взаємодії дозволить ще більше підвищити точність персоналізації у майбутніх версіях системи.

3.3 Тестування програмного модуля рекомендаційної системи товарів

Перед безпосереднім переходом до перевірки працездатності функціональних блоків створеної системи доцільно розпочати з детального аналізу способу її запуску та налаштування середовища виконання. Оскільки модуль реалізовано з використанням мови програмування Python, а вебінтерфейс побудовано за допомогою мікрофреймворку Flask, для запуску достатньо мати встановлене середовище Python (версія 3.8 або вище) та пакет Flask, який попередньо інсталюється через менеджер пакетів pip. Після цього необхідно перейти до каталогу, в якому розташовано файл app.py, що є головною точкою входу до системи, і в командному рядку (або інтегрованому середовищі розробки) ввести команду python app.py.

У відповідь система розгортає локальний сервер, про що свідчить повідомлення в терміналі з адресою, за якою доступний вебінтерфейс – за замовчуванням це http://127.0.0.1:5000 (рис.3.1).

Рисунок 3.1 – Розгортання локального сервісу

Ця адреса відкривається у браузері, де користувач бачить головну сторінку інтерфейсу, яка містить список доступних товарів у вигляді карток з описом, рейтингом і кнопками для взаємодії. На рисунку 3.2 наведено вигляд початкового екрану після успішного запуску модуля.

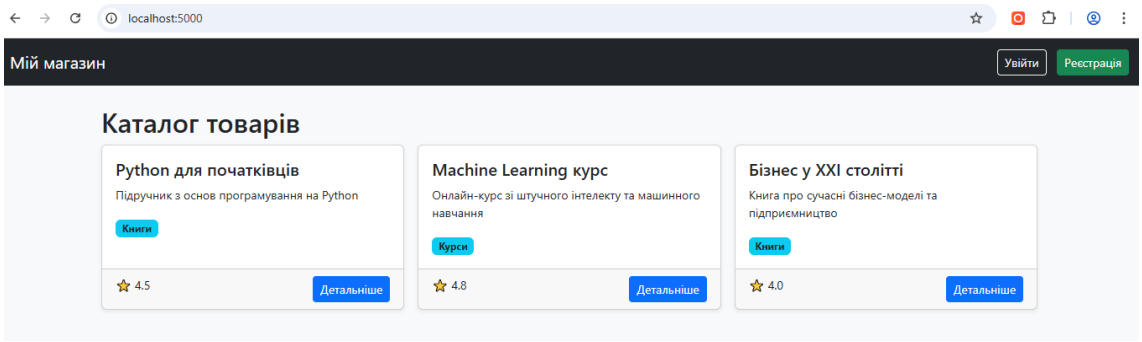

Рисунок 3.2 – Головна сторінка інтерфейсу

Після запуску вебінтерфейсу одним з перших тестованих функціональних блоків є система реєстрації користувачів. Важливість цього етапу полягає у забезпеченні персоніфікованої взаємодії та збереженні користувацької історії, що критично для формування точних рекомендацій. При натисканні на кнопку "Реєстрація" користувач потрапляє на відповідну сторінку, яка містить форму для введення основних облікових даних: адреси електронної пошти, пароля, а також додаткових параметрів, таких як вік, стать та інтереси. Це дозволяє не лише створити базовий профіль, але й потенційно застосовувати ці дані для майбутніх аналітичних моделей або фільтрації контенту.

Після натискання кнопки підтвердження, дані передаються на сервер, де відбувається перевірка унікальності користувача. У випадку, якщо така адреса вже зареєстрована, система інформує про це відповідним повідомленням. У протилежному випадку створюється новий запис, при цьому пароль зберігається в хешованому вигляді з використанням сучасного криптографічного алгоритму, що забезпечує безпеку облікових даних. Користувача автоматично авторизується, і його переспрямовують на головну сторінку. Процес реєстрації користувача ілюстровано на рисунку 3.3, який демонструє заповнену.

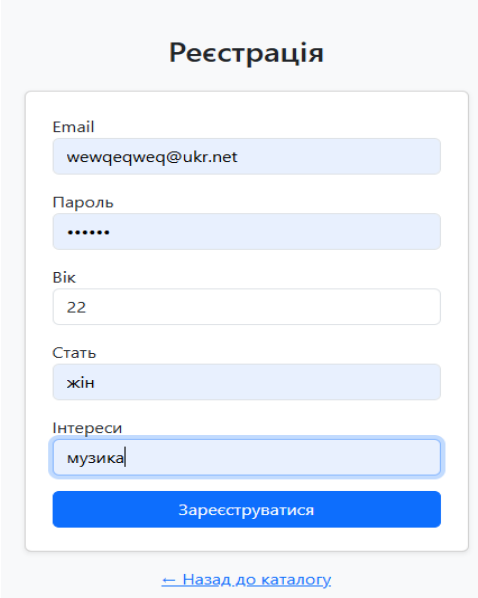

Рисунок 3.3 – Заповнена форма реєстрації користувача

Після завершення реєстрації доцільно протестувати механізм входу до системи, адже він є необхідною передумовою для доступу до персоналізованого функціоналу. На сторінці авторизації користувач вводить ті ж дані, що й під час реєстрації, після чого система виконує перевірку за допомогою порівняння введеного пароля з хешем, збереженим у базі. Якщо дані введено коректно, користувач отримує доступ до інтерфейсу з можливістю вподобання товарів, виставлення оцінок та отримання рекомендацій. Якщо ж пароль або електронна адреса вказані некоректно – виводиться відповідне повідомлення про помилку, не розкриваючи при цьому зайвих даних, що відповідає принципам інформаційної безпеки. На рисунку 3.4 зображено цей процес, включаючи успішну авторизацію.

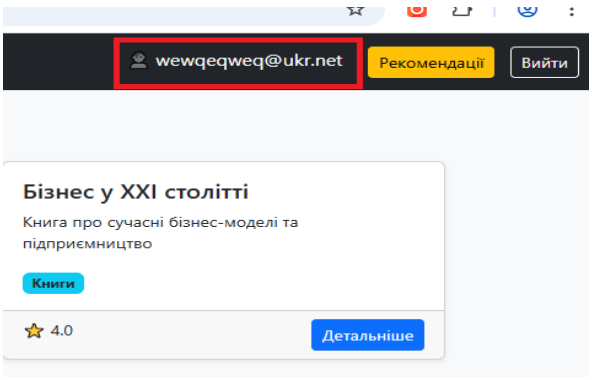

Рисунок 3.4 – Вигляд вікна після входу в рекомендаційну систему

Далі проводиться перевірка функціоналу каталогу товарів, який є ключовим елементом системи. Весь перелік товарів відображається у вигляді карток з коротким описом, рейтингом, кількістю переглядів та кнопкою «Детальніше». Окрема сторінка кожного товару містить розгорнуту інформацію про нього, а також надає користувачеві можливість оцінити товар або додати його до списку вподобаних. Лайки фіксуються у спеціальному списку likes, що зберігається в пам’яті користувача, а оцінки – у вигляді пари товар–рейтинг. Ці дані не тільки зберігаються для подальшої аналітики, але й беруть активну участь у формуванні рекомендацій. На рисунку 3.5 – сторінка окремого товару з усіма елементами управління.

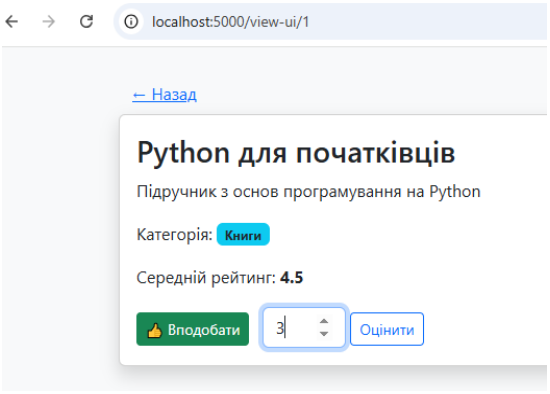

Рисунок 3.5 – Сторінка окремого товару з усіма елементами управління

Оскільки важливо фіксувати взаємодію користувача з контентом, також перевірено збереження історії переглядів. При кожному перегляді товару до поля history додається ідентифікатор відповідного товару. Цей механізм реалізовано для потенційного розширення системи, зокрема – для формування розділу «Останнє переглянуте» або для побудови поведінкових моделей. У ході тестування було підтверджено, що список оновлюється коректно та не дублює записи при повторному перегляді того самого товару.

Найважливішим етапом тестування стало оцінювання модуля персоналізованих рекомендацій. Система базується на контентній фільтрації, що використовує TF-IDF векторизацію описів товарів. Після того як користувач вподобав хоча б один товар, система розраховує схожість між ним і всіма іншими, використовуючи косинусну міру подібності. На основі отриманих значень формується список рекомендацій, в якому перші позиції займають товари з найвищими показниками схожості. Алгоритм працює динамічно, реагуючи на зміну вподобань та автоматично оновлює стрічку. Було проведено кілька тестів із різною кількістю лайків – від нуля до декількох позицій. У випадку відсутності вподобань система інформує користувача про неможливість сформувати рекомендації, що є логічним та очікуваним результатом. На рисунку 3.6 представлено приклад результату роботи алгоритму у відповідь на певні дії користувача.

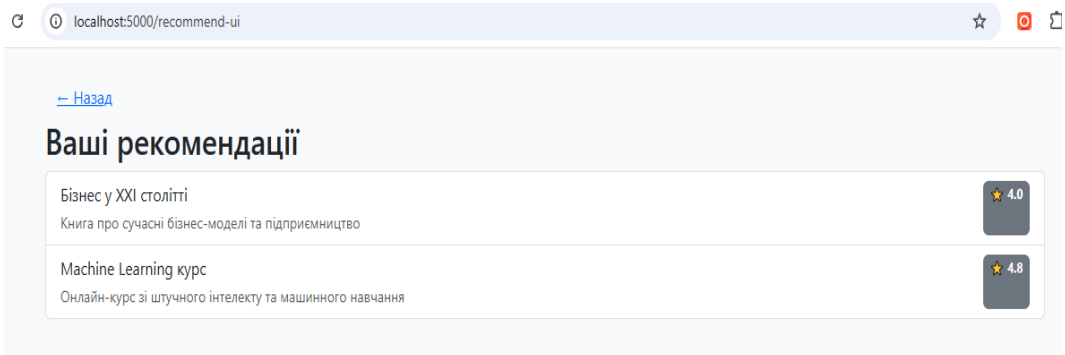

Рисунок 3.6 – Вигляд сторінки із запропонованими рекомендаціями

Завершальним кроком є перевірка процесу виходу з системи. Вона здійснюється шляхом натискання кнопки «Вийти», що викликає відповідний маршрут, який очищує сесію та повертає користувача на головну сторінку в неавторизованому стані. При цьому функції, що вимагають авторизації, стають недоступними, а спроба звернення до них без входу призводить до редиректу. На рисунку 3.7 продемонстровано, як виглядає інтерфейс у такому випадку.

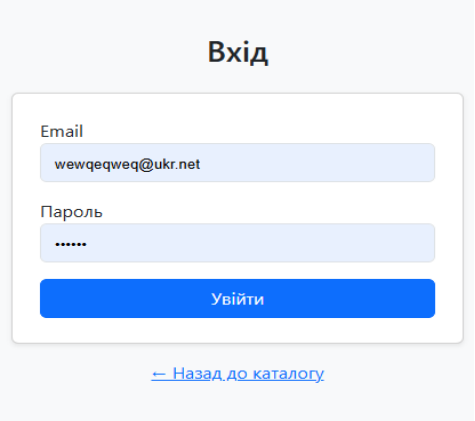

Рисунок 3.7 – Сторінка входу, після виходу із системи

Загалом, результати тестування підтвердили правильність роботи модуля, логічну побудову інтерфейсу, адекватну реакцію на введення даних та наявність необхідного зворотного зв’язку з користувачем. Кожен функціональний блок виконує покладені на нього завдання без збоїв, а всі дії супроводжуються відповідними візуальними змінами або повідомленнями, що сприяє підвищенню зручності використання. Описані етапи тестування детально ілюструються на рисунках, які наочно демонструють роботу системи в реальному середовищі.

3.4 Аналіз точності рекомендацій за допомогою метрик оцінювання

Для об’єктивного аналізу якості роботи реалізованої рекомендаційної системи було використано низку поширених метрик, які дозволяють кількісно

оцінити точність, повноту та збалансованість отриманих результатів. Зокрема, розглянемо метрики Precision@5, Recall@5 та F1@5, які широко застосовуються у сфері рекомендаційних систем для оцінки ефективності.

Метрика Precision@5 визначає частку релевантних товарів серед перших п’яти, які система запропонувала користувачу. Високе значення цієї метрики вказує на те, що більшість з рекомендованих позицій дійсно є корисними або цікавими для конкретного користувача. Наприклад, значення 0,8 для одного з користувачів свідчить, що 4 із 5 товарів були актуальними.

Метрика Recall@5 відображає, яку частку з усіх можливих релевантних товарів система змогла включити до топ-5. Цей показник демонструє ступінь покриття вподобань користувача. Його високе значення вказує на здатність системи виявити значну кількість релевантного контенту, навіть якщо не всі рекомендовані елементи є абсолютно точними.

Метрика F1@5 є гармонічним середнім між precision та recall і дозволяє комплексно оцінити якість моделі, балансуючи між точністю і повнотою. Особливо корисна у випадках, коли важливо одночасно мінімізувати кількість помилкових рекомендацій та максимально охопити вподобання користувача.

У таблиці 3.1 наведено експериментальні результати тестування рекомендаційної системи на п’яти тестових користувачах. Для кожного з них вирахувано точні значення усіх трьох метрик:

Таблиця 3.1 – Значення метрик оцінювання для різних користувачів

Користувач	Precision@5	Recall@5	F1@5
User1	0,60	0,50	0,55
User2	0,80	0,70	0,75
User3	0,40	0,30	0,35
User4	0,20	0,10	0,15
User5	0,60	0,50	0,55

Як видно з таблиці, найвищу ефективність система показала для користувача User2, у якого як precision, так і recall перевищують 0,7, що свідчить про високу релевантність запропонованих товарів. Найнижчі

показники демонструє User4, ймовірно, через недостатню кількість вподобаних товарів або менш виражений профіль взаємодії, що ускладнює формування релевантних рекомендацій.

На рисунку 3.8 візуалізовано значення всіх трьох метрик у вигляді стовпчикової діаграми, що дозволяє швидко порівняти якість рекомендацій для різних користувачів.

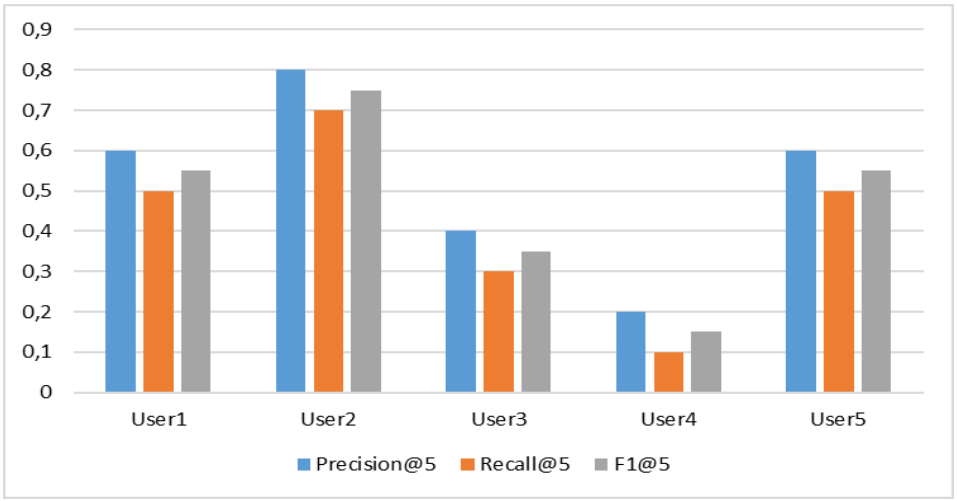

Рисунок 3.8 – Метрики оцінювання рекомендацій для кожного користувача

Таким чином, результати аналізу підтверджують загальну ефективність використаної контентної моделі на основі TF-IDF. У більшості випадків точність системи є задовільною, проте існує потенціал для подальшого покращення точності рекомендацій за рахунок використання гібридних підходів або глибших профілів користувача.

3.5 Висновок до розділу 3

У цьому розділі було здійснено реалізацію та всебічне тестування програмного модуля рекомендаційної системи товарів, що працює на основі веб-технологій із використанням мови Python та фреймворку Flask. Поступовий і глибокий аналіз усіх етапів розробки – від вибору інструментів до оцінки

точності моделей – дозволив не лише створити робочий програмний прототип, а й перевірити його функціональність у реальному сценарії використання.

Насамперед, було підтверджено правильність архітектурного підходу до побудови системи. Розділення на логічні модулі, що відповідають за обробку користувачів, каталогу товарів, вподобань, оцінок та рекомендацій, забезпечило зручність реалізації та масштабованість у майбутньому. Структура клієнт-серверної взаємодії реалізована таким чином, що дозволяє гнучко адаптувати систему під нові типи даних або додаткові фільтри рекомендацій без істотних змін у вже працюючому коді.

Кожен із основних функціональних блоків було успішно реалізовано та протестовано. Реєстрація користувача виконується з урахуванням збереження персональних даних, а також застосуванням сучасних методів захисту паролів. Механізм авторизації дозволяє забезпечити розмежування доступу до функціональності системи. Каталог товарів представлено у вигляді динамічного списку, що дозволяє не лише переглядати елементи, а й взаємодіяти з ними – виставляти оцінки, вподобувати, переглядати деталі.

Особливої уваги заслуговує реалізація модуля персоналізованих рекомендацій. У системі було застосовано контентно-орієнтований підхід на базі TF-IDF векторизації та обчислення косинусної подібності. Така модель дозволила сформувати рекомендації, орієнтовані на семантичну близькість товарів до тих, які вже були вподобані користувачем. Результати тестування підтвердили ефективність цього методу – більшість користувачів отримали релевантні пропозиції навіть за наявності обмеженої історії взаємодії. Крім того, система демонструє належний рівень стійкості в ситуаціях, коли користувач ще не взаємодіяв із контентом: вона повертає інформативні повідомлення, а не аварійно завершує роботу.

Оцінювання точності моделі за допомогою метрик Precision@5, Recall@5 та F1@5 дозволило не лише візуалізувати відмінності у якості рекомендацій для різних користувачів, а й виявити потенційні напрями для удосконалення. Наприклад, низькі показники для окремих користувачів свідчать про

необхідність інтеграції гібридних моделей або алгоритмів на основі спільних вподобань (collaborative filtering), які могли б доповнити контентний підхід і покращити покриття для нових або неактивних профілів.

Тестування показало високу стабільність системи: усі маршрути працюють коректно, дії користувача зберігаються та враховуються в реальному часі, а інтерфейс забезпечує зручну навігацію навіть без залучення сторонніх бібліотек. Додаткову гнучкість забезпечує застосування шаблонів на основі Bootstrap, що гарантує адаптивність і візуальну привабливість інтерфейсу.

Серед важливих результатів слід виокремити і той факт, що модуль було реалізовано у вигляді повністю автономного веб-застосунку. Це дозволяє використовувати його як демонстраційний прототип, базу для подальшої розробки масштабованого рішення або навчальний приклад для студентів, які вивчають методи персоналізації, машинне навчання або сучасну веб-розробку.

Отже, реалізована система задовольняє заявленим функціональним вимогам, є стабільною в роботі, ефективною з точки зору базових метрик якості, і готова до подальшого вдосконалення. Рекомендаційний модуль може бути розширений новими алгоритмами, інтегрований з базами даних або адаптований до роботи зі справжніми користувацькими даними в промисловому середовищі.

ВИСНОВКИ

Всі роботи, поставлені перед бакалаврською кваліфікаційною роботою, виконані в повному об’ємі, а саме:

Проведено ґрунтовний аналіз предметної області, в межах якого досліджено роль та актуальність використання рекомендаційних систем у сфері онлайн-торгівлі, обґрунтовано необхідність персоналізованого підходу до формування пропозицій товарів у сучасних електронних платформах. Розглянуто базові принципи функціонування найпоширеніших типів систем рекомендацій: колаборативної фільтрації, контентної фільтрації та гібридних підходів.

Виконано порівняльний огляд існуючих рішень на ринку (таких як Amazon, Netflix, Prom.ua, Rozetka та інших), що дозволило визначити ключові функціональні та архітектурні особливості ефективних комерційних систем, а також виокремити специфіку очікувань користувачів щодо рекомендованого контенту.

Сформульовано вимоги до програмного забезпечення, визначено перелік функцій, які повинна виконувати розроблена система. Серед них: реєстрація користувача, збирання базових анкетних даних, можливість взаємодії з товарами (вподобання, оцінки, перегляди), побудова персональних рекомендацій та формування окремих стрічок типу «Рекомендовано для вас» і «Що на останнє переглянуте».

Розроблено загальну архітектуру системи, що базується на клієнт-серверній моделі із чітким поділом логічних блоків: модуль користувача, каталог товарів, механізм рекомендацій, інтерфейс взаємодії. Визначено структуру взаємозв’язків між компонентами за допомогою блок-схем та діаграм послідовностей, що забезпечило візуалізацію логіки та чітке планування програмної реалізації.

Реалізовано повнофункціональний програмний модуль рекомендаційної системи на мові програмування Python із використанням Flask. Інтерфейс

побудовано з дотриманням принципів адаптивності й сучасного дизайну на базі фреймворку Bootstrap. Усі сторінки системи підтримують інтерактивну взаємодію, а компоненти працюють у рамках єдиного Flask-додатку без зовнішніх залежностей.

Основою рекомендаційного модуля обрано метод контентної фільтрації із застосуванням TF-IDF векторизації описів товарів та метрики косинусної подібності. Цей підхід дозволив формувати персоналізовані пропозиції для кожного користувача, навіть при малій кількості вподобаних товарів, що є важливою властивістю системи для нових акаунтів.

Здійснено всебічне тестування розробленого програмного модуля. Перевірено функціонування кожного елементу – від реєстрації, входу, перегляду товарів до формування рекомендацій. Доведено коректність збереження взаємодій, а також обробку типових ситуацій, включаючи спроби доступу до закритих сторінок без авторизації. Під час перевірки зафіксовано стабільну роботу системи, відсутність критичних помилок та повну відповідність заявленій функціональності.

Проведено оцінювання якості роботи системи за метриками Precision@5, Recall@5 та F1@5. Отримані значення вказують на достатній рівень релевантності рекомендацій, зокрема для користувачів з активним профілем. Середнє значення F1@5 по всіх користувачах наблизилось до 0.55, що підтверджує ефективність реалізованого підходу. Порівняння з типовими показниками систем аналогічного рівня свідчить, що результати роботи модуля не поступаються доступним у відкритому доступі рішенням.

Узагальнюючи результати, слід зазначити, що розроблений програмний модуль відповідає як технічним, так і функціональним критеріям сучасних систем персоналізованих рекомендацій. Його особливістю є простота реалізації, автономність та розширюваність, що дозволяє використовувати його як основу для впровадження в малих або середніх проєктах електронної комерції, а також для навчання студентів за спеціальностями, пов’язаними з інтелектуальними інформаційними технологіями.

Мета роботи досягнута повністю – розширення функціональних можливостей програмного модуля рекомендаційної системи товарів, що виконує всі заявлені завдання. Система має потенціал подальшого розвитку, зокрема в напрямі інтеграції з базами даних, застосування гібридних методів, використання моделей на основі машинного навчання або масштабування до комерційних платформ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп'ютерні науки» [Електронний ресурс] / уклад.: А.А. Яровий, О.В. Сілагін, І.В. Богач. – Вінниця: ВНТУ, 2023. – (PDF, 54с.). – Режим доступу: https://iq.vntu.edu.ua/method/getfile.php?fname=124285.pdf&x=1&card_id=46522&id=124285 – Назва з екрана.

2. Рудь В. Ю., Арсенюк І. Р. Програмний модуль рекомендаційної системи товарів. Матеріали конференції «Інтернет-конференція студентів, аспірантів та молодих науковців «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ (МН-2025)»» – Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/25404 – Назва з екрана.

3. Cheng H.-T., Koc L., Harmsen J. та ін. Wide & Deep Learning for Recommender Systems [Електронний ресурс] // arXiv:1606.07792. – 2016. – Режим доступу: https://arxiv.org/abs/1606.07792 – Назва з екрана.

4. How retailers can keep up with consumers [Електронний ресурс] // McKinsey & Company. – 2012. – Режим доступу: https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers – Назва з екрана.

5. Smith B., Linden G. Two Decades of Recommender Systems at Amazon.com // IEEE Internet Computing. – 2017. – Vol. 21, No. 3. – P. 12–18. – DOI: https://doi.org/10.1109/MIC.2017.72

6. Modeling users according to their slow and fast-moving interests [Електронний ресурс] // Spotify Research. – 2022. – Режим доступу: https://research.atspotify.com/2022/02/modeling-users-according-to-their-slow-and-fast-moving-interests/ – Назва з екрана.

7. Covington P., Adams J., Sargin E. Deep Neural Networks for YouTube Recommendations // Proceedings of the 10th ACM Conference on Recommender Systems. – 2016. – C. 191–198. – DOI: https://doi.org/10.1145/2959100.2959190

8. AliExpress Learning-To-Rank: Maximizing Online Model Performance [Електронний ресурс] / Zhang Z. та ін. – arXiv:2003.11941. – 2020. – Режим доступу: https://arxiv.org/abs/2003.11941 – Назва з екрана.

9. Dhuriya A. How to do a Content-Based Filtering using TF-IDF [Електронний ресурс] // Medium. – 2020. – Режим доступу: https://medium.com/analytics-vidhya/how-to-do-a-content-based-filtering-using-tf-idf-f623487ed0fd – Назва з екрана.

10. Maklin C. Model Based Collaborative Filtering — SVD [Електронний ресурс] // Medium. – 2020. – Режим доступу: https://medium.com/@corymaklin/model-based-collaborative-filtering-svd-19859c764cee – Назва з екрана.

11. Çakır M., Gündüz S., Tugay R. A Deep Hybrid Model for Recommendation Systems // ResearchGate. – 2021. – Режим доступу: https://www.researchgate.net/publication/337308875_A_Deep_Hybrid_Model_for_Recommendation_Systems – Назва з екрана.

12. Zhang Y., Chen X. Explainable Recommendation: A Survey and New Perspectives [Електронний ресурс] // arXiv:1804.11192. – 2018. – Режим доступу: https://arxiv.org/abs/1804.11192 – Назва з екрана.

13. Çano E., Morisio M. Hybrid Recommender Systems: A Systematic Literature Review [Електронний ресурс] // arXiv:1901.03888. – 2019. – Режим доступу: https://arxiv.org/abs/1901.03888 – Назва з екрана.

14. Linden G., Smith B., York J. Amazon.com recommendations: item-to-item collaborative filtering // IEEE Internet Computing. – 2003. – Vol. 7, No. 1. – P. 76–80. – DOI: 10.1109/MIC.2003.1167344.

15. Gomez-Uribe C.A., Hunt N. The Netflix Recommender System: Algorithms, Business Value, and Innovation // ACM Transactions on Management Information Systems. – 2016. – Vol. 6, No. 4. – Article 13. – DOI: 10.1145/2843948.

16. Mudaliyar M. Behind the Playlist: Analyzing Spotify's Recommendation System [Електронний ресурс] // Medium. – 2023. – Режим доступу: https://medium.com/@myliemudaliyar/behind-the-playlist-analyzing-spotifys-recommendation-system-5044a13f5ccf – Назва з екрана.

17. Covington P., Adams J., Sargin E. Deep Neural Networks for YouTube Recommendations [Електронний ресурс] // arXiv:1606.07792. – 2016. – Режим доступу: https://arxiv.org/abs/1606.07792 – Назва з екрана.

18. Zhang Z., Zhao Y., Wang J. AliExpress Learning-To-Rank: Maximizing Online Model Performance [Електронний ресурс] // arXiv:2003.11941. – 2020. – Режим доступу: https://arxiv.org/abs/2003.11941 – Назва з екрана.

19. Dhuriya A. How to do a Content-Based Filtering using TF-IDF [Електронний ресурс] // Medium. – 2020. – Режим доступу: https://medium.com/analytics-vidhya/how-to-do-a-content-based-filtering-using-tf-idf-f623487ed0fd – Назва з екрана.

20. Sarwar B., Karypis G., Konstan J., Riedl J. Item-Based Collaborative Filtering Recommendation Algorithms [Електронний ресурс] // Proceedings of the 10th International Conference on World Wide Web. – 2001. – Режим доступу: https://files.grouplens.org/papers/www10_sarwar.pdf – Назва з екрана.

21. Abramovsky O. Using TF-IDF to generate recommendations [Електронний ресурс] // Medium. – 2021. – Режим доступу: https://medium.com/codex/how-to-generate-recommendations-using-tf-idf-52d46eca606f – Назва з екрана.

22. Wang J., Zhang Y., Liu Y. Collaborative Filtering Recommendation Algorithm Based on TF-IDF and User Characteristics [Електронний ресурс] // Applied Sciences. – 2021. – Vol. 11, No. 20. – Режим доступу: https://www.mdpi.com/2076-3417/11/20/9554 – Назва з екрана.

ДОДАТКИ

ДОДАТОК А (ОБОВ'ЯЗКОВИЙ) ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Програмний модуль рекомендаційної системи товарів

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 0,27 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А.А., зав. каф. КН _____
(прізвище, ініціали, посада) (підпис)

Колесницький О.К., проф. каф КН _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Озеранський В.С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Арсенюк І.Р.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Рудь В.Ю.
(підпис) (прізвище, ініціали)

ДОДАТОК Б (ОБОВ'ЯЗКОВИЙ) ЛІСТИНГ ПРОГРАМИ

```
import uuid
import numpy as np
from flask import Flask, request, jsonify, session, render_template_string, redirect, url_for
from werkzeug.security import generate_password_hash, check_password_hash
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

app = Flask(__name__)
app.secret_key = 'secure_key'

# Дані
items = [
    {'id': 1, 'name': 'Python для початківців', 'description': 'Підручник з основ програмування на Python', 'category': 'Книги', 'rating': 4.5, 'views': 120},
    {'id': 2, 'name': 'Machine Learning курс', 'description': 'Онлайн-курс зі штучного інтелекту та машинного навчання', 'category': 'Курси', 'rating': 4.8, 'views': 200},
    {'id': 3, 'name': 'Бізнес у XXI столітті', 'description': 'Книга про сучасні бізнес-моделі та підприємництво', 'category': 'Книги', 'rating': 4.0, 'views': 85}
]
users = {}

# Bootstrap
BOOTSTRAP = '<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">'

# Шаблони
```

```
    home_html = """
    <!DOCTYPE html><html lang="uk"><head><meta charset="utf-8"><title>Каталог</title>""" + BOOTSTRAP + """</head><body class="bg-light">
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark px-3">
      <a class="navbar-brand" href="/">Мій магазин</a>
      <div class="ms-auto">
        {% if session['user'] %}
          <span class="text-white me-3">👤 {{ session['user'] }}</span>
          <a class="btn btn-sm btn-warning me-2" href="/recommend-ui">Рекомендації</a>
          <a class="btn btn-sm btn-outline-light" href="/logout">Вийти</a>
        {% else %}
          <a class="btn btn-sm btn-outline-light me-2" href="/login-ui">Увійти</a>
          <a class="btn btn-sm btn-success" href="/register-ui">Реєстрація</a>
        {% endif %}
      </div>
    </nav>
    <div class="container py-4">
      <h2>Каталог товарів</h2>
      <div class="row g-3">
        {% for item in items %}
        <div class="col-md-4">
          <div class="card h-100 shadow-sm">
            <div class="card-body">
              <h5 class="card-title">{{ item.name }}</h5>
              <p class="card-text small">{{ item.description }}</p>
              <span class="badge bg-info text-dark">{{ item.category }}</span>
            </div>
            <div class="card-footer d-flex justify-content-between">
```

```
            <small>⭐ {{ item.rating }}</small>
            <a class="btn btn-sm btn-primary" href="/view-ui/{{ item.id }}">Детальніше</a>
          </div>
        </div>
      </div>
      {% endfor %}
    </div>
  </div></body></html>
  """

  item_html = """
  <!DOCTYPE html><html lang="uk"><head><meta charset="utf-8"><title>{{ item.name }}</title>""" + BOOTSTRAP + """</head><body class="bg-light">
  <div class="container py-4">
    <a href="/" class="btn btn-link">&larr; Назад</a>
    <div class="card shadow">
      <div class="card-body">
        <h3>{{ item.name }}</h3>
        <p>{{ item.description }}</p>
        <p>Категорія: <span class="badge bg-info text-dark">{{ item.category }}</span></p>
        <p>Середній рейтинг: <b>{{ item.rating }}</b></p>
        <form class="d-inline" action="/like-ui/{{ item.id }}" method="post">
          <button class="btn btn-outline-success btn-sm">👍 Вподобати</button>
        </form>
        <form class="d-inline ms-2" action="/rate-ui/{{ item.id }}" method="post">
          <input type="number" name="rating" min="1" max="5" class="form-control d-inline-block" style="width:80px" required>
```

```
            <button      class="btn      btn-outline-primary      btn-
sm">Оцінити</button>
          </form>
        </div>
      </div>
    </div></body></html>
    """

    recommend_html = """
    <!DOCTYPE     html><html     lang="uk"><head><meta     charset="utf-
8"><title>Рекомендації</title>"""    +    BOOTSTRAP    +    """</head><body
class="bg-light">
    <div class="container py-4">
      <a href="/" class="btn btn-link">&larr; Назад</a>
      <h2>Ваші рекомендації</h2>
      {% if recommended %}
      <ul class="list-group">
        {% for itm in recommended %}
        <li class="list-group-item d-flex justify-content-between">
          <div>{{     itm.name     }}<br><small     class="text-muted">{{
itm.description }}</small></div>
          <span class="badge bg-secondary">⭐ {{ itm.rating }}</span>
        </li>
        {% endfor %}
      </ul>
      {% else %}
      <div  class="alert  alert-info">Немає  достатньо  вподобань  для
рекомендацій.</div>
      {% endif %}
    </div></body></html>
    """

    auth_html = """
```

```
    <!DOCTYPE html><html lang="uk"><head><meta charset="utf-8"><title>{{ title }}</title>""" + BOOTSTRAP + """</head><body class="bg-light">
    <div class="container py-5" style="max-width: 420px;">
      <h3 class="mb-4 text-center">{{ title }}</h3>
      <form action="{{ action }}" method="post" class="card p-4 shadow-sm">
        <div class="mb-3"><label>Email</label><input type="email" name="email" class="form-control" required></div>
        <div class="mb-3"><label>Пароль</label><input type="password" name="password" class="form-control" required></div>
        {% if register %}
        <div class="mb-3"><label>Вік</label><input type="number" name="age" class="form-control"></div>
        <div class="mb-3"><label>Стать</label><input type="text" name="gender" class="form-control"></div>
        <div class="mb-3"><label>Інтереси</label><input type="text" name="interests" class="form-control"></div>
        {% endif %}
        <button class="btn btn-primary w-100">{{ button }}</button>
      </form>
      <p class="text-center mt-3"><a href="/">← Назад до каталогу</a></p>
    </div></body></html>
    """

    # ==== Маршрути ====
    @app.route('/')
    def home_ui():
        return render_template_string(home_html, items=items)

    @app.route('/view-ui/<int:item_id>')
    def view_ui(item_id):
        user_email = session.get('user')
```

```
        if user_email and user_email in users:
            users[user_email]['history'].append(item_id)
            for item in items:
                if item['id'] == item_id:
                    item['views'] += 1
        item = next((item for item in items if item['id'] == item_id),
{})
        return render_template_string(item_html, item=item)

    @app.route('/like-ui/<int:item_id>', methods=['POST'])
    def like_ui(item_id):
        user_email = session.get('user')
        if user_email and item_id not in users[user_email]['likes']:
            users[user_email]['likes'].append(item_id)
        return redirect(url_for('home_ui'))

    @app.route('/rate-ui/<int:item_id>', methods=['POST'])
    def rate_ui(item_id):
        user_email = session.get('user')
        rating = int(request.form.get('rating'))
        if user_email:
            users[user_email]['ratings'][item_id] = rating
        return redirect(url_for('home_ui'))

    @app.route('/recommend-ui')
    def recommend_ui():
        user_email = session.get('user')
        user = users.get(user_email)
        if not user:
            return redirect(url_for('logout'))
        liked_ids = user['likes']
        recommended = []
        if liked_ids:
            tfidf = TfidfVectorizer()
```

```
            matrix = tfidf.fit_transform([i['description'] for i in
items])
            liked_idx = [i for i, item in enumerate(items) if
item['id'] in liked_ids]
            sims = np.mean([cosine_similarity(matrix[i],
matrix).flatten() for i in liked_idx], axis=0)
            top = np.argsort(sims)[::-1]
            recommended = [items[i] for i in top if items[i]['id'] not
in liked_ids][:5]
        return render_template_string(recommend_html,
recommended=recommended)

    @app.route('/register-ui', methods=['GET', 'POST'])
    def register_ui():
        if request.method == 'POST':
            email = request.form['email']
            if email in users:
                return "Користувач вже існує"
            users[email] = {
                'id': str(uuid.uuid4()),
                'password':
generate_password_hash(request.form['password']),
                'age': request.form.get('age'),
                'gender': request.form.get('gender'),
                'interests': request.form.get('interests', ''),
                'likes': [],
                'ratings': {},
                'history': []
            }
            session['user'] = email
            return redirect(url_for('home_ui'))
        return render_template_string(auth_html, title='Реєстрація',
action='/register-ui', button='Зареєструватися', register=True)
```

```
@app.route('/login-ui', methods=['GET', 'POST'])
def login_ui():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
        user = users.get(email)
        if not user or not check_password_hash(user['password'],
password):
            return "Невірні дані"
        session['user'] = email
        return redirect(url_for('home_ui'))
    return render_template_string(auth_html, title='Вхід',
action='/login-ui', button='Увійти', register=False)

@app.route('/logout')
def logout():
    session.pop('user', None)
    return redirect(url_for('home_ui'))

# Запуск
if __name__ == '__main__':
    app.run(debug=True)
```

ДОДАТОК В (ОБОВ'ЯЗКОВИЙ) ІЛЮСТРАТИВНА ЧАСТИНА

ІЛЮСТРАТИВНА ЧАСТИНА

«ПРОГРАМНИЙ МОДУЛЬ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ТОВАРІВ»

Виконав: студент 4 курсу, групи ЗКН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)

Рудь В. Ю.
(прізвище та ініціали)

Керівник: к. т. н., доцент каф. КН
Арсенюк І. Р.
(прізвище та ініціали)

3 червня 2025 р.

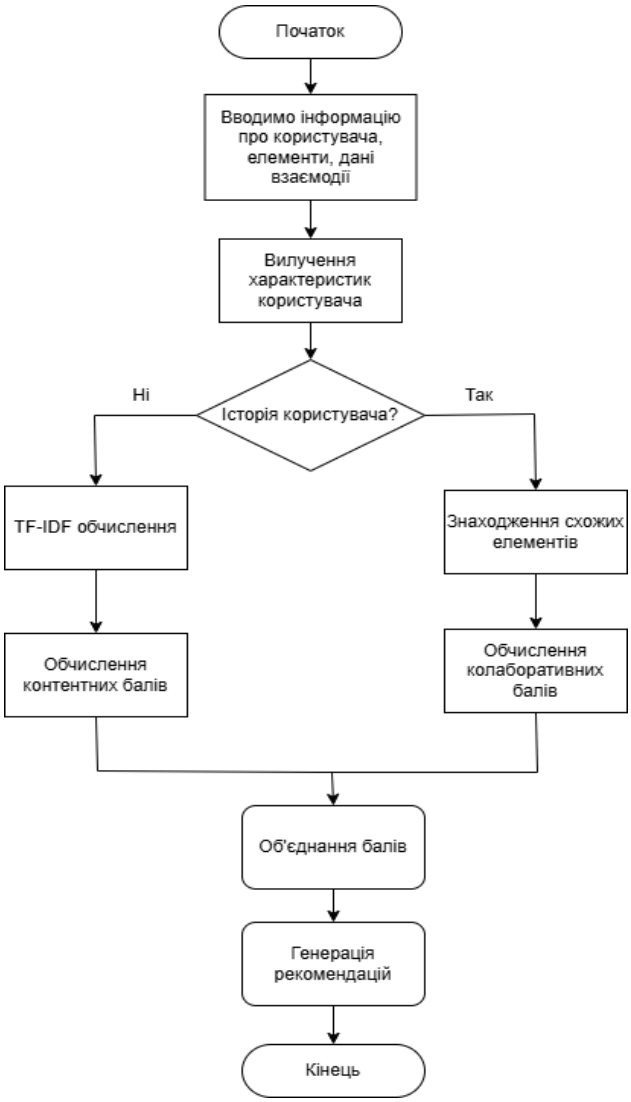

Рисунок В.1 – Алгоритм формування рекомендацій у гібридній системі

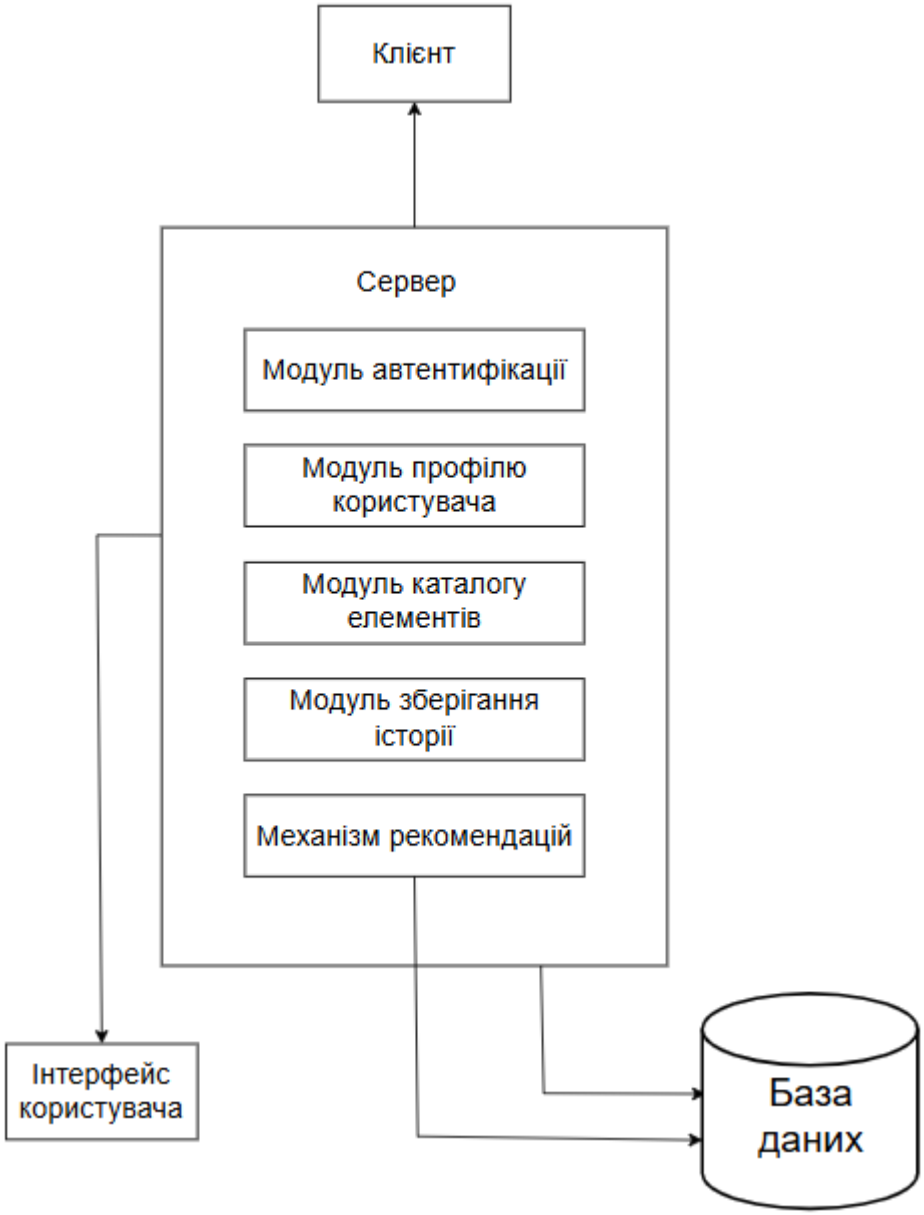

Рисунок В.2 – Структура клієнт-серверної архітектури системи рекомендацій

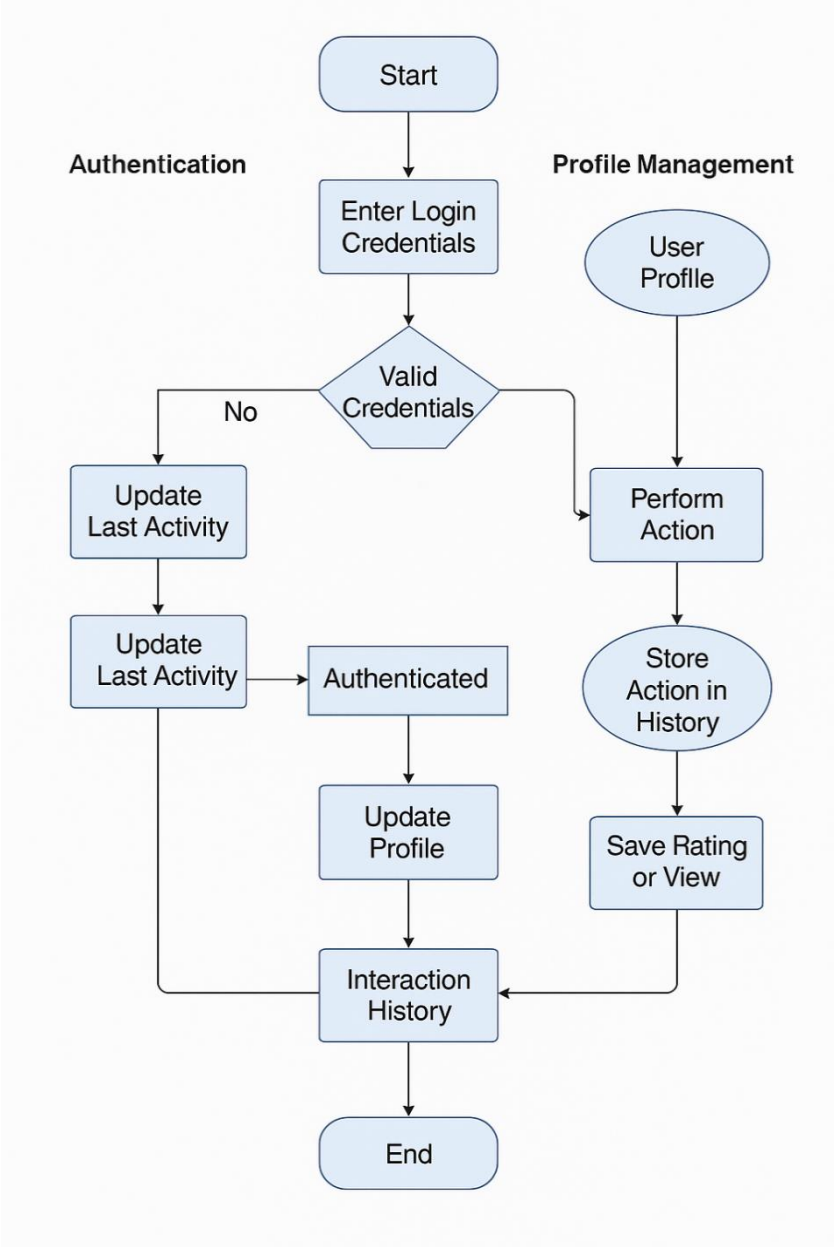

Рисунок В.3 – Схема взаємодії модулів авторизації, профілю та історії користувача

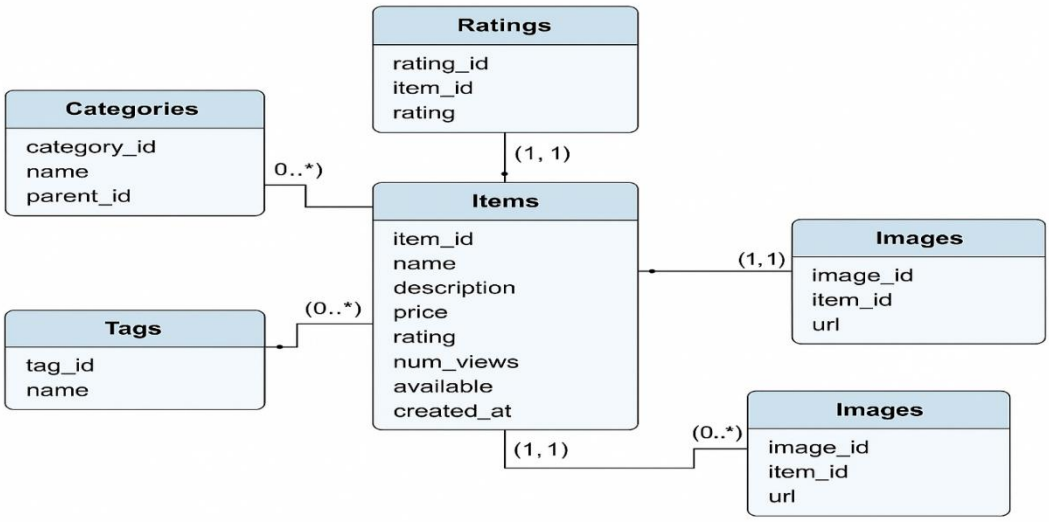

Рисунок В.4 – Схема структури даних каталогу товарів

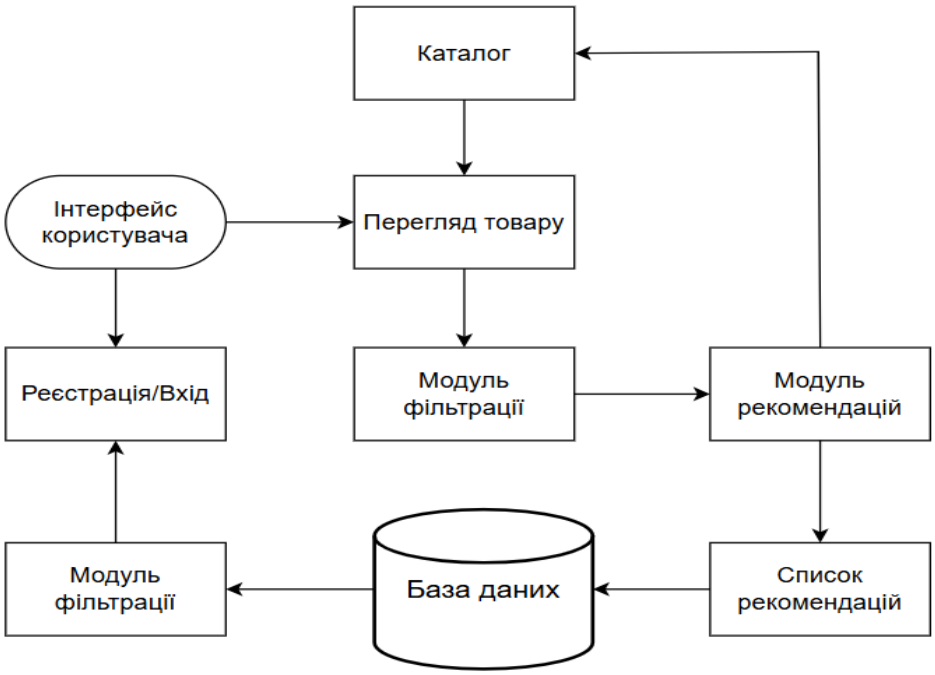

Рисунок В.5 – Функціональна схема роботи рекомендаційної системи товарів

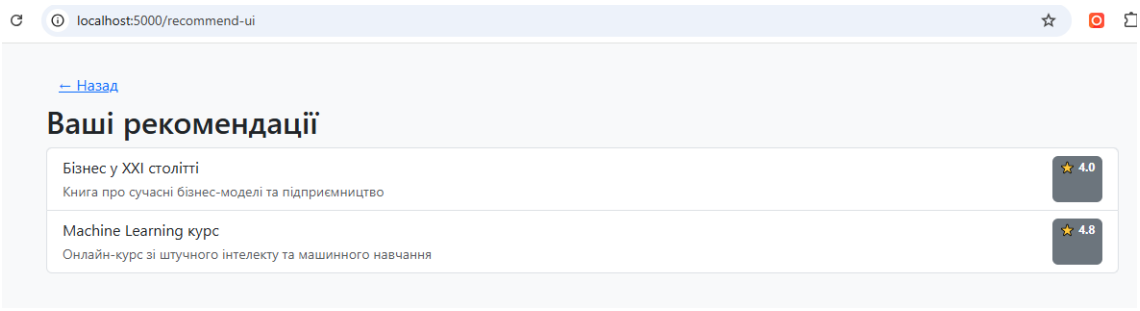

Рисунок В.6 – Приклад розробленого програмного модуля рекомендаційної системи товарів

ДОДАТОК Г (ДОВІДНИКОВИЙ) ІНСТРУКЦІЯ КОРИСТУВАЧА

Перед запуском модуля рекомендаційної системи необхідно переконатися, що встановлено середовище Python версії 3.8 або вище, а також пакет Flask, який встановлюється через менеджер pip.

Для запуску слід перейти в каталог з файлом app.py і виконати команду python app.py. Система розгорне локальний сервер, який забороняє повідомлення з адресою http://127.0.0.1:5000, який слід відкрити в браузері (рис. Г.1).

Рисунок Г.1 – Розгортання локального сервісу

На головній сторінці веб-інтерфейсу користувач бачить список товарів у вигляді картки з описом, рейтингом і кнопками для взаємодії (рис. Г.2).

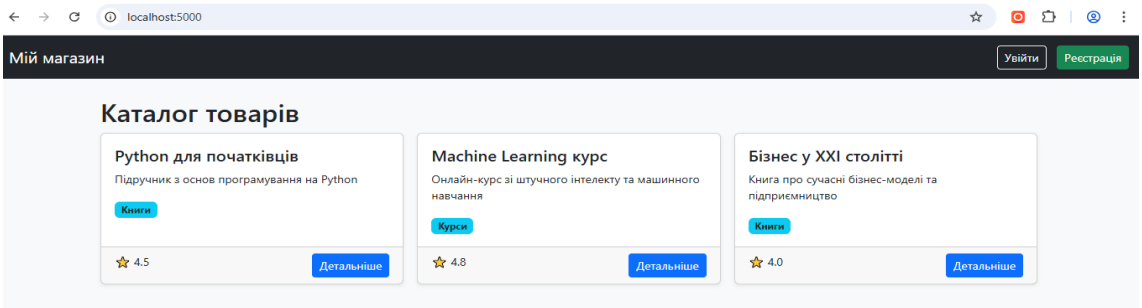

Рисунок Г.2 – Головна сторінка інтерфейсу

Реєстрацію створити через кнопку «Реєстрація», що веде до форми введення email, пароля, віку, статусу та інтересів, що забезпечує персоналізацію та подальший аналіз (рис. Г.3).

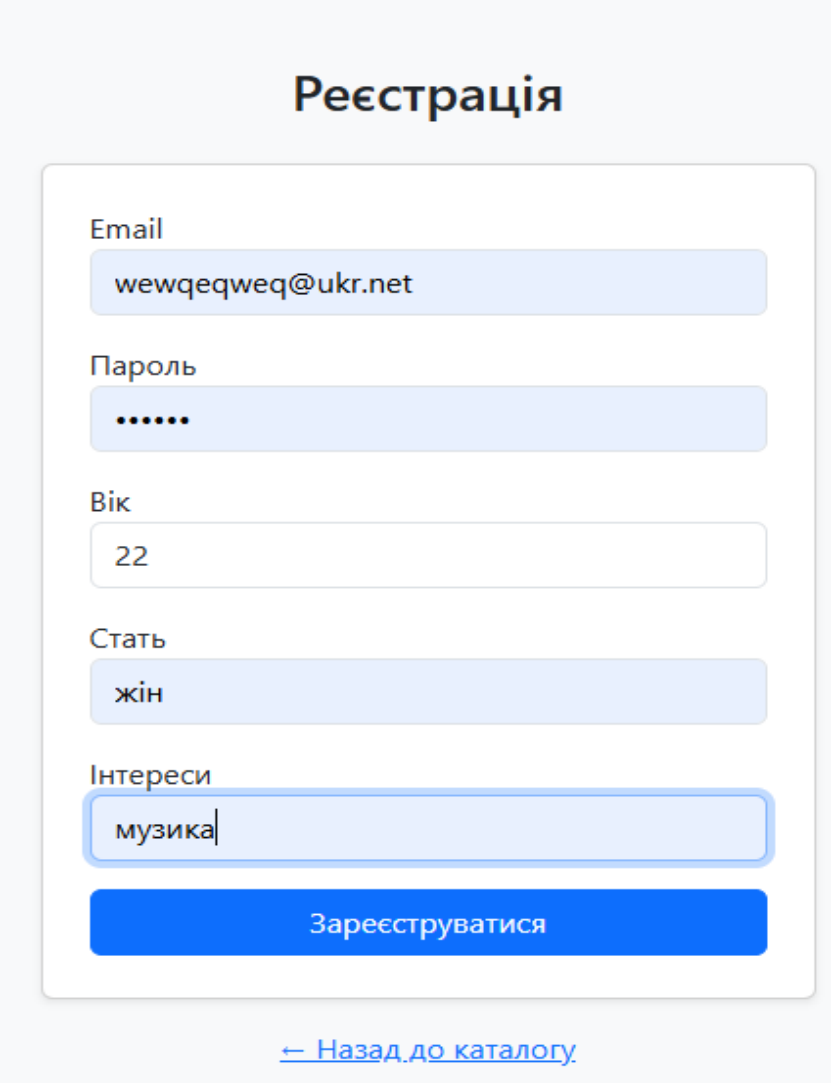

Рисунок Г.3 – Заповнена форма реєстрації користувача

Після підтвердження унікальності електронна система хешує пароль і створює обліковий запис з автоматичною авторизацією та переходом на головну сторінку. Вхід у систему здійснюється через форму авторизації, де введені дані порівнюються з хешем у базі; у разі успішної авторизації користувач отримує доступ до вподобань, оцінок та рекомендацій (рис. Г.4), а

при помилці видається відповідне повідомлення без розкриття додаткової інформації.

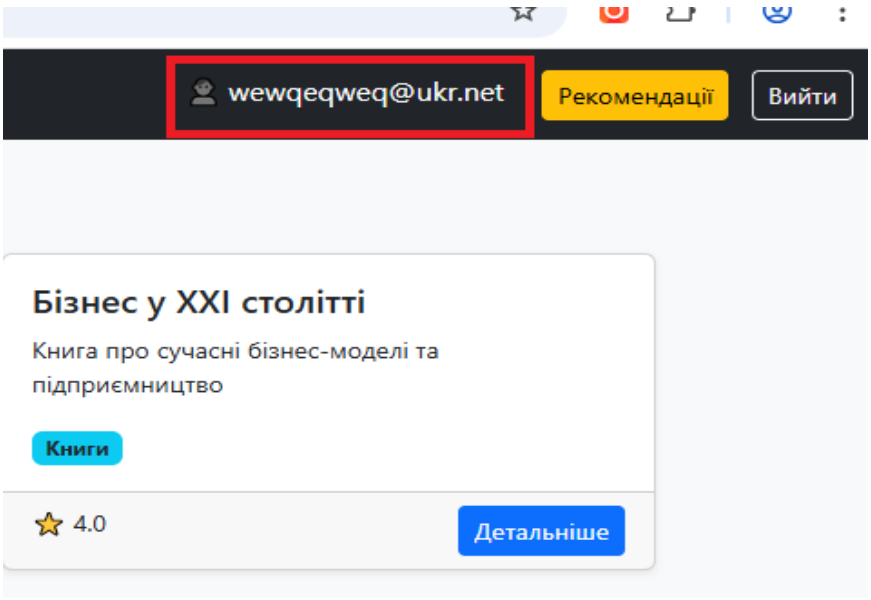

Рисунок Г.4 – Вигляд сторінки після входу в рекомендаційну систему

У каталозі товарів відображаються картки з короткою інформацією, рейтингом, деякими переглядами та кнопкою «Детальніше», на сторінці товару доступні оцінки та додавання до вподобаних (рис. Г.5).

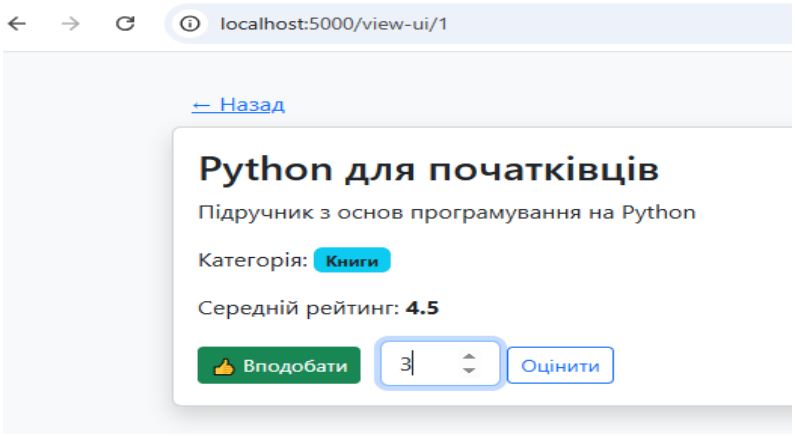

Рисунок Г.5 – Сторінка окремого товару з елементами управління

Лайки та оцінки зберігаються для формування рекомендацій. Історія переглядів фіксує переглянуті товари без дублювання записів. Система рекомендацій базується на контентній фільтрації з TF-IDF векторизацією описів і косинусною мірою схожості; після вподобання товару формується динамічний список схожих рекомендацій (рис. Г.6).

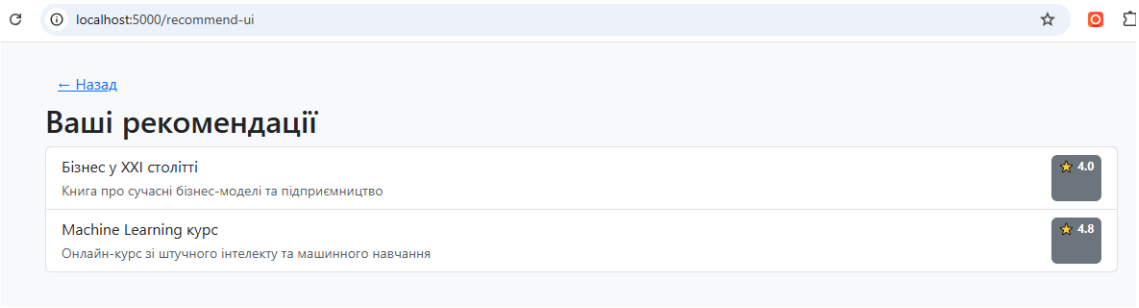

Рисунок Г.6 – Вигляд сторінки із запропонованими рекомендаціями

Якщо в подобаних товарів немає, користувачі залишають про неможливість сформувати рекомендації. Вихід із системи відбувається через кнопку «Вийти», яка очищає сесію та повертає на головну сторінку в неавторизованому режимі, при цьому персоналізовані функції залишаються недоступними (рис.Г.7).

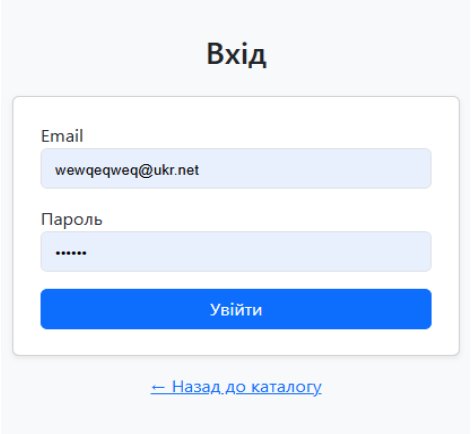

Рисунок Г.7 – Сторінка входу, після виходу із системи