

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Розробка системи рекомендацій товарів на платформі
для оренди одягу та аксесуарів»**

Виконав студент групи ІПЗм-24-1
КОЛІСНИК Ольга Володимирівна

Керівник роботи:
ГОДЛЕВСЬКИЙ Юрій Олександрович

Рецензент:
ВАКАЛЮК Тетяна Анатоліївна

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«18» вересня 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КОЛІСНИК Ольга Володимирівна**

Керівник роботи: **ГОДЛЕВСЬКИЙ Юрій Олександрович**

Тема роботи: «Розробка системи рекомендацій товарів на платформі для оренди одягу та аксесуарів»,

затверджена Наказом закладу вищої освіти від «18» вересня 2025 р. № 491/с

Вихідні дані для роботи: процеси персоналізації користувацького досвіду та рекомендації товарів платформи для оренди одягу та аксесуарів; методи та інформаційні технології, що використовуються для побудови систем рекомендацій товарів, їх алгоритмічні підходи, архітектурні рішення та вплив на ефективність роботи платформи.

Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Вакалюк Т. А.	22.09.2025р.	22.09.2025р.
2	Вакалюк Т. А.	22.09.2025р.	22.09.2025р.
3	Вакалюк Т. А.	22.09.2025р.	22.09.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота магістра складається системи рекомендацій товарів, а також з програмного комплексу веб - сайту платформи для оренди одягу та аксесуарів та пояснювальної записки. Пояснювальна записка до випускної роботи містить текстову частину, викладену на 92 сторінках друкованого тексту, має 8 сторінок додатків. Список використаних джерел містить 32 найменування і займає 7 сторінок. В роботі наведено 27 рисунків та 1 таблиця.

Метою роботи є розробка системи рекомендацій товарів для платформи оренди одягу та аксесуарів з метою підвищення персоналізації користувацького досвіду та оптимізації вибору товарів, що надаються для оренди.

КЛЮЧОВІ СЛОВА: СИСТЕМА РЕКОМЕНДАЦІЙ, ПЕРСОНАЛІЗОВАНІ РЕКОМЕНДАЦІЇ, ГІБРИДНІ АЛГОРИТМИ, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, КОНТЕНТНЕ МОДЕЛЮВАННЯ, ЕЛЕКТРОННА КОМЕРЦІЯ, ПЕРСОНАЛІЗАЦІЯ

ABSTRACT

The master's thesis consists of a software package for a web platform for renting clothing and accessories, as well as a product recommendation system and an explanatory note. The explanatory note of the thesis contains a textual part of 92 pages of printed text and includes 8 pages of appendices. The list of references contains 32 sources and takes 7 pages. The thesis includes 27 illustrations and 1 table.

The purpose of the work is to develop a product recommendation system for a platform for renting clothing and accessories in order to enhance the personalization of the user experience and optimize the selection of items available for rent.

KEY WORDS: RECOMMENDATION SYSTEM, PERSONALIZED RECOMMENDATIONS, HYBRID ALGORITHMS, COLLABORATIVE FILTERING, CONTENT MODELING, E-COMMERCE, PERSONALIZATION

ЗМІСТ

ВСТУП.....	9
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ.....	12
1.1. Постановка завдання.....	12
1.2. Огляд наявних аналогів.....	13
1.3. Вибір програмних засобів для реалізації.....	18
1.4. Огляд наявних методів, моделей, алгоритмів.....	22
Висновки до розділу 1.....	33
РОЗДІЛ 2. РОЗРОБКА ТА ПРОЕКТУВАННЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ.....	34
2.1. Аналіз задачі, засобів та методів її вирішення.....	34
2.2. Побудова моделі аналізу і практична реалізація.....	37
2.3. Результати проведеного аналізу в таблицях діаграмах і графіках та аналіз результатів.....	50
Висновки до розділу 2.....	56
РОЗДІЛ 3. ОПИС ПРОГРАМНОГО ПРОДУКТУ.....	58
3.1. Структура інтерфейсу рекомендаційної системи на платформі оренди одягу.....	58
3.2. Відображення результатів рекомендацій.....	72
Висновки до розділу 3.....	74
ВИСНОВКИ.....	75
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	77
ДОДАТКИ.....	84

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

MVT – схема розділення даних додатків і керуючої логіки на три окремі компоненти Model-View-Template.

CF – Collaborative Filtering (колаборативна фільтрація).

IBCF – Item Based Collaborative Filtering (колаборативна фільтрація на основі товарів).

UBCF – User Based Collaborative Filtering (колаборативна фільтрація на основі користувачів).

CBF – Content Based Filtering (контентна фільтрація).

SQL – Structured Query Language.

БД – база даних.

СУБД – система управління базами даних.

ПЗ – програмне забезпечення.

ВСТУП

Актуальність теми. У сучасних умовах стрімкого розвитку електронної комерції та сервісів спільного споживання особливого поширення набувають онлайн-платформи з оренди одягу та аксесуарів. Ці сервіси дають змогу користувачам економити кошти, зменшувати надмірне споживання та сприяти екологічній сталій моді. Водночас із розширенням асортименту та зростанням кількості користувачів з'являється проблема складності вибору серед великого каталогу товарів користувачеві непросто знайти речі, що максимально відповідають його стилю, фізичним параметрам, уподобанням і контексту використання [1].

Традиційні механізми пошуку (фільтри, категорії, ручне сортування) вже не здатні забезпечити достатньо персоналізовану взаємодію. У таких умовах ключовим інструментом підвищення зручності та ефективності взаємодії користувача з платформою стають рекомендаційні системи, які автоматично формують індивідуальні добірки товарів на основі аналізу поведінкових і контекстних даних.

Саме тому впровадження інтелектуальних рекомендаційних алгоритмів дозволяє суттєво покращити якість взаємодії з системою, підвищити конверсію оренд, зменшити час пошуку речей та забезпечити користувачу релевантні пропозиції.

Додатковою складністю є те, що для модних товарів традиційні методи рекомендацій (наприклад, класичний user-based чи item-based collaborative filtering) часто дають недостатню точність через високу суб'єктивність оцінок, наявність великої кількості унікальних товарів та користувачів з малою кількістю взаємодій. Це підсилює необхідність застосування гібридних моделей, що поєднують колаборативні, контентні та контекстні підходи.

Отже, актуальність теми кваліфікаційної роботи зумовлена потребою у створенні ефективної рекомендаційної системи для платформи оренди одягу та аксесуарів, яка здатна враховувати широкий спектр характеристик

товарів, особливості користувачів і контекст використання, забезпечуючи високий рівень персоналізації та покращення користувацького досвіду. Розробка такої системи сприятиме підвищенню якості сервісу, сталому розвитку індустрії моди та оптимізації процесів електронної комерції.

Мета і завдання дослідження. Метою дослідження є розробка системи рекомендацій товарів для платформи оренди одягу та аксесуарів з метою підвищення персоналізації користувацького досвіду та оптимізації вибору товарів, що надаються для оренди.

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

- проаналізувати сучасні підходи до розробки систем рекомендацій на платформах електронної комерції;
- вивчити особливості ринку оренди одягу та аксесуарів, а також вимоги до систем рекомендацій у даному сегменті;
- обрати оптимальні алгоритми для побудови системи рекомендацій (фільтрація на основі контенту, колаборативна фільтрація, гібридні моделі);
- реалізувати алгоритм персоналізації користувацького досвіду та рекомендацій товарів;
- розробити архітектуру та прототип системи рекомендацій для платформи;
- провести тестування та аналіз ефективності системи рекомендацій на основі реальних даних користувачів платформи.

Об'єктом дослідження є процеси персоналізації користувацького досвіду та рекомендації товарів платформи для оренди одягу та аксесуарів.

Предметом дослідження є методи та інформаційні технології, що використовуються для побудови систем рекомендацій товарів, їх алгоритмічні підходи, архітектурні рішення та вплив на ефективність роботи платформи.

Наукова новизна роботи полягає у поєднанні та адаптації сучасних алгоритмів рекомендацій до специфіки домену оренди одягу та аксесуарів, а

також у створенні комплексної моделі персоналізації, яка враховує унікальні поведінкові патерни користувачів, характерні саме для сервісів короткотривалої оренди. У розробленій системі враховуються оцінки з відгуків, перегляди, оренди (ренти як подія, що має найбільшу вагу в рекомендаціях), тоді як більшість класичних систем рекомендацій аналізують лише перегляди, або оцінки, або покупки.

Практичне значення полягає у створенні працездатної системи рекомендацій товарів, орієнтованої на потреби онлайн-платформи оренди одягу та аксесуарів. Розроблена система забезпечує формування персоналізованих рекомендацій на основі аналізу поведінкових даних користувачів, характеристик товарів та особливостей їх використання. Запропонований підхід дозволяє суттєво підвищити точність підбору товарів, скоротити час пошуку, а також покращити загальний користувацький досвід [2].

Апробація. За темою кваліфікаційної магістерської роботи було опубліковано 2 тези конференції [3, 4].

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ

1.1. Постановка завдання

Розвиток електронної комерції та економіки спільного використання зумовлює необхідність створення ефективних систем рекомендацій, які сприятимуть персоналізованому підбору товарів відповідно до потреб користувачів. У контексті платформи оренди одягу та аксесуарів система рекомендацій має враховувати специфічні аспекти цього ринку, зокрема змінюваність модних тенденцій, сезонність попиту та особливості споживчої поведінки [5].

Основне завдання розробки системи рекомендацій полягає у підвищенні рівня персоналізації взаємодії користувача з платформою та оптимізації вибору товарів. Для цього необхідно інтегрувати поведінкові дані користувачів, що дозволить формувати більш релевантні пропозиції та підвищувати рівень задоволеності клієнтів [6].

Виходячи з мети дослідження, можна сформулювати такі підзадачі:

- аналіз сучасних підходів до реалізації систем рекомендацій у сфері електронної комерції, включаючи методи фільтрації на основі контенту, колаборативну фільтрацію та гібридні моделі;
- вивчення специфіки ринку оренди одягу та аксесуарів, визначення основних факторів, що впливають на вибір товарів користувачами;
- визначення набору даних, необхідного для ефективної роботи системи рекомендацій, та методів його збору та обробки;
- обґрунтування вибору оптимального алгоритму рекомендацій, який найкраще відповідає вимогам платформи;
- реалізація прототипу системи та його тестування на основі реальних даних користувачів;
- аналіз ефективності впровадженої системи рекомендацій, її впливу на залученість користувачів, частоту оренди товарів.

Таким чином, розробка системи рекомендацій сприятиме вдосконаленню функціоналу платформи для оренди одягу та аксесуарів, забезпечуючи її конкурентоспроможність та відповідність сучасним вимогам цифрового ринку.

1.2. Огляд наявних аналогів

У сфері оренди одягу та аксесуарів важливу роль відіграють цифрові платформи, що забезпечують доступ до широкого асортименту товарів без необхідності їх купувати. Попит на такі сервіси зумовлює активне використання рекомендаційних систем, покликаних персоналізувати взаємодію з користувачем, скоротити час пошуку речей та підвищити конверсію оренд. У даному підрозділі розглянуто два найбільш відомі міжнародні сервіси так які Rent the Runway (США) та By Rotation (Велика Британія).

Rent the Runway (RTR) (рис. 1.1.) є однією з найбільш відомих платформ преміальної оренди одягу, що поєднує разові оренди та систему підписок. Масштабність сервісу забезпечує наявність великої кількості структурованих даних, що дає можливість застосовувати складні алгоритмічні моделі. Рекомендаційна система RTR базується на використанні колаборативної та контентної фільтрації, а також на інтеграції персональних параметрів користувачів. Колаборативна фільтрація дозволяє визначати подібні групи клієнтів за історією їхніх оренд та оцінок, формуючи рекомендації на основі статистично значущої схожості поведінкових патернів. Одночасно з цим контентний підхід передбачає аналіз характеристик самих товарів, включаючи фасон, колірну гаму, матеріали чи дизайнерські рішення, що дозволяє системі пропонувати речі, схожі на раніше обрані користувачем.

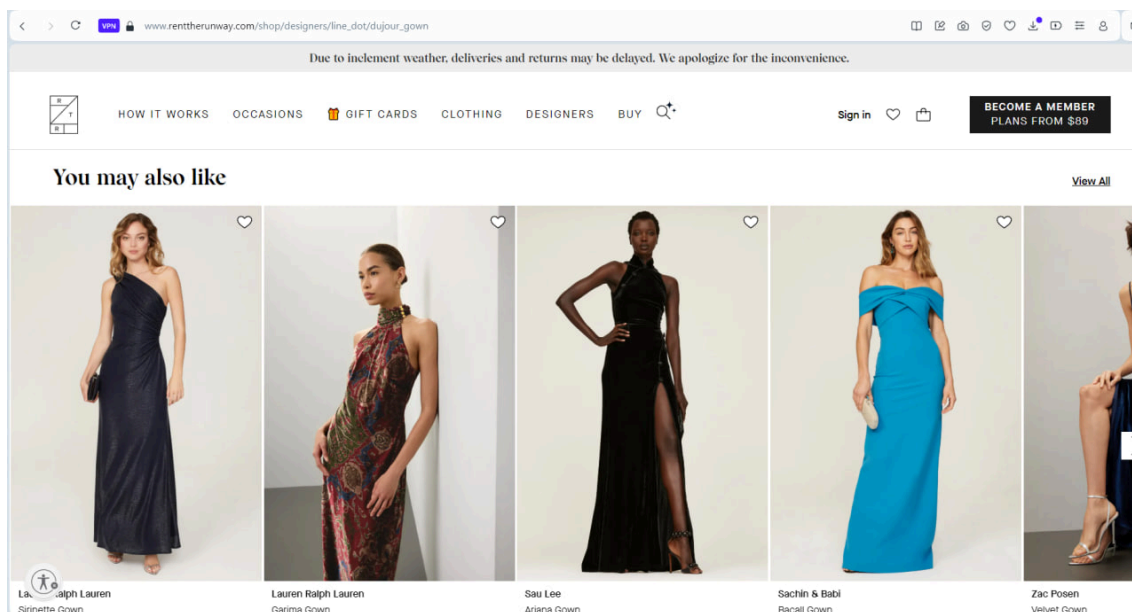


Рис. 1.1. Сторінка сайту на якій працює система рекомендацій «Rent the Runway»

Rent the Runway використовує алгоритмічну персоналізацію для підбору одягу відповідно до вподобань користувача. До основних принципів роботи рекомендаційної системи відноситься аналіз історії оренди. Система відстежує який одяг певних брендів чи стилів користувач орендує найчастіше, враховується частота оренди різних категорій (вечірній, офісний, буденний одяг), а тоді пропонує схожі речі. Рекомендації також формуються на основі рейтингу та відгуків клієнтів на товар. Користувачі залишають оцінки та коментарі, а відгуки включають параметри фігури та реальні фото, що допомагає в підборі ідеального розміру які враховуються для формування рекомендацій. Сезонність має свій вплив пропоновані товари. Рекомендації змінюються залежно від пори року, свят та популярних модних тенденцій, наприклад, у грудні система частіше пропонує сукні для святкових заходів, а влітку – легкий одяг. Останнім, але не менш важливим є дані про зріст, вагу, параметри користувача, оскільки Rent the Runway враховує введені користувачем фізичні параметри, щоб рекомендувати одяг, який найкраще підходить особливостям статури. Використовуються дані з попередніх оренд про те, який розмір пасував найкраще. На сайті працює алгоритми машинного навчання. Система навчається на основі поведінки користувачів,

оптимізуючи рекомендації для кожного клієнта і використовуються нейронні мережі та факторизаційні моделі для прогнозування уподобань. Алгоритм також застосовує контентну фільтрацію, суть якої полягає у підборі речей схожого стилю. Система аналізує матеріали, кольори, крій та інші особливості речей, які клієнт переглядав або орендував, і пропонує схожі варіанти. У свою чергу інший метод, колаборативна фільтрація, пропонує товар, який користується популярністю серед клієнтів зі схожими параметрами, наприклад з подібними параметрами тіла та стилем, тощо.

Завдяки цій системі Rent the Runway допомагає клієнтам швидше знаходити підходящі речі, підвищуючи рівень персоналізації вибору.

By Rotation - це британська онлайн-платформа та мобільний додаток для оренди дизайнерського та преміального одягу за моделлю peer-to-peer (P2P), яка заохочує ділитися речами між учасниками програми (рис. 1.2). Відмінність від традиційних сервісів оренди в тому, що речі здають не компанії, а звичайні користувачі, які можуть заробляти на своєму гардеробі. Платформа об'єднує моду, технології та екологічний підхід, пропонуючи стильні рішення без необхідності купувати нові речі.

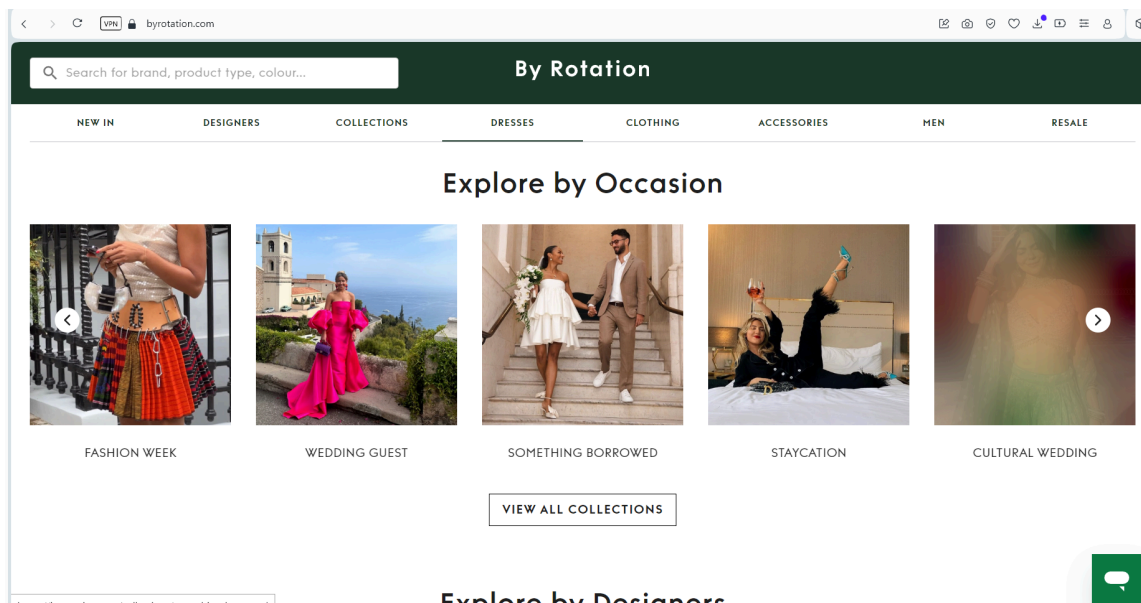


Рис. 1.2. Головна сторінка By Rotation

Основною перевагою By Rotation над конкурентами є саме модель P2P. Замість централізованого складу компанії, користувачі самі здають в оренду власний одяг. Це дає можливість мати ширший вибір речей, а також отримувати унікальні дизайнерські предмети, які не завжди доступні в традиційних магазинах або прокатах. Перевагою також є гнучкість у виборі та термінах оренди, тобто користувачі можуть орендувати одяг на різні періоди – від кількох днів до тижнів, що підходить як для короткострокових заходів, так і для довготривалого користування. Платформа функціонує не просто як магазин, а як соціальна мережа, користувачі можуть підписуватися одне на одного, переглядати гардероби інших учасників, обмінюватися порадами щодо стилю та залишати відгуки.

До недоліків By Rotation варто віднести те що оренда доступна лише в Великій Британії, що обмежує можливості міжнародного використання. Оскільки речі надають приватні особи, є ризик отримати одяг з дефектами або ж пошкодити його, що може спричинити додаткові витрати на компенсацію. Орендовані речі можуть відрізнятися за станом, оскільки власники самостійно доглядають за ними, тоді як відсутність єдиного стандарту може бути проблемою для деяких користувачів. Процес бронювання може займати більше часу, оскільки By Rotation це P2P-платформа, користувачам потрібно домовлятися про оренду напряму з власниками, що може ускладнювати процес порівняно з централізованими сервісами.

На відміну від RTR, платформа By Rotation функціонує за моделлю peer-to-peer, у якій користувачі напряму здають власний одяг в оренду. Така структура зумовлює інший акцент у побудові системи рекомендацій, який зміщується у бік соціальної взаємодії між користувачами. Рекомендаційний механізм By Rotation враховує стилістичні вподобання, сформовані на основі переглядів, збережених товарів та оцінених образів. Значну роль у формуванні рекомендацій відіграє соціальний профіль користувача: система аналізує підписки, вподобання, переглянуті гардероби та взаємодію з іншими

учасниками платформи, створюючи своєрідну мережу стилістичної близькості.

Контентний аналіз у By Rotation також залишається важливим компонентом, однак він тісно переплітається з соціальними фільтрами. Платформа аналізує характеристики товарів, а також рейтинги орендодавців і відгуки, що дозволяє формувати рекомендації, орієнтовані не лише на стиль, а й на якість та надійність орендодавця. Додатково алгоритми враховують географічні чинники, оскільки логістика між користувачами є важливою складовою моделі P2P [7]. Машинне навчання використовується для прогнозування популярності товарів, зміни трендів та адаптації рекомендацій до сезонних коливань.

By Rotation використовує інтелектуальну систему підбору одягу, яка враховує безліч параметрів, щоб зробити рекомендації максимально персоналізованими. Алгоритми машинного навчання аналізують вподобані та збережені речі користувача та використовуються для визначення стилістичних уподобань. Система рекомендує речі від найбільш активних та високооцінюваних орендодавців, що підвищує довіру до якості товару. Враховуються модні тренди та адаптує пропозиції відповідно до сезону (наприклад, улітку більше легких суконь, узимку — теплих пальт). На сайті використовує модель "соціальних рекомендацій", аналізуючи з ким взаємодіє користувач, чий стиль переглядає, кому ставить лайки, тощо. Враховується географічне положення користувача, підбираючи варіанти, які знаходяться найближче, щоб мінімізувати витрати на доставку. By Rotation бере до уваги відгуки та коментарі, одяг, який має найкращі оцінки та позитивні відгуки, з більшою ймовірністю буде рекомендований іншим користувачам. Також, якщо користувач орендував певні бренди або категорії одягу, система пропонує схожі речі, враховуючи його минулі вибори.

Отже, By Rotation — це унікальний сервіс оренди одягу, який поєднує комерцію та соціальну взаємодію. Його система рекомендацій базується не

лише на алгоритмічних методах, а й на взаємодії між користувачами, що створює більш персоналізований підхід.

У проекті, що розробляється в межах даної кваліфікаційної роботи, система рекомендацій повинна базуватися на гібридному підході, поєднуючи колаборативну фільтрацію, контентний аналіз та персоналізовані пропозиції на основі поведінкових даних користувачів. Важливо враховувати історію оренди та переглядів, щоб пропонувати товари, які можуть зацікавити клієнта, а також аналізувати параметри користувача, такі як зріст, вага та обхвати, для точного підбору розміру.

Рекомендаційна система повинна адаптуватися до сезону, святкових подій та змін у модних трендах, пропонуючи відповідний одяг для різних випадків. Важливим елементом є інтеграція механізму оцінок та відгуків, що допоможе уточнювати рекомендації на основі реального досвіду інших користувачів. Крім того, платформа має враховувати стильові уподобання клієнта, аналізуючи вибір брендів, кольорів і фасонів, та пропонувати готові образи, підібрані штучним інтелектом.

1.3. Вибір програмних засобів для реалізації

Веб-додаток буде побудований на клієнт-серверній архітектурі [8]. Серверна частина буде представлена у вигляді API та буде написана з використанням сучасного бекенд фреймворку Django [9]. Для клієнтської частини буде використано HTML, CSS та JavaScript, а як сховище буде використовуватись база даних SQL3.

Django — це високорівневий веб-фреймворк на мові програмування Python, який дозволяє швидко створювати надійні і масштабовані веб-застосунки. Django розроблявся з акцентом на зручність використання, безпеку та масштабованість, що робить його популярним вибором серед розробників веб-застосунків.

Django належить до веб-фреймворків, що реалізують підхід, подібний до архітектурного патерну MVC, однак у власній специфічній інтерпретації, відомій як MTV (Model-Template-View). Такий підхід чітко розділяє логіку

роботи застосунку, структуру даних і спосіб їх відображення. У Django модель відповідає за опис структури даних, створення таблиць у базі та взаємодію з ними через ORM-рівень. Шаблони формують зовнішній вигляд веб-сторінок і керують тим, як інформація буде відображатися у браузері користувача. Логічний компонент – view – містить бізнес-логіку, обробляє HTTP-запити, отримує необхідні дані з моделі та передає їх у шаблони для подальшого рендерингу.

Однією з ключових переваг Django є висока швидкість розробки. Фреймворк надає великий набір вбудованих інструментів, включно з ORM, системою маршрутизації, адміністраторською панеллю та механізмами аутентифікації, що дає змогу значно скоротити час розробки складних веб-систем. Важливою перевагою є також підвищена безпека: Django автоматично захищає застосунок від більшості поширених вразливостей, таких як SQL-ін'єкції, міжсайтове виконання скриптів (XSS), підміна запитів (CSRF) та інші. Завдяки цьому він широко використовується для створення надійних платформ, де важливими є коректність обробки даних та захист користувацької інформації.

Ще однією суттєвою перевагою Django є його масштабованість. Завдяки ефективному управлінню ресурсами і підтримці різних механізмів кешування, балансування навантаження та інтеграції з високопродуктивними серверами (наприклад, Gunicorn, Nginx), фреймворк підходить не лише для невеликих застосунків, а й для великих високонавантажених систем. Важливим фактором є також велика та активна спільнота розробників, а також детальна офіційна документація, що значно спрощує навчання, пошук рішень та впровадження нових можливостей у проект.

Попри численні переваги, Django має і певні недоліки. Для початківців він може здатися складним через велику кількість концепцій та вбудованих механізмів, які потрібно освоїти. Крім того, для невеликих чи дуже простих проєктів використання Django може бути надмірним, адже його структура та вимоги до конфігурації інколи перевищують потреби таких систем. Окремо

варто згадати й шаблонізатор Django, що має певні обмеження порівняно з сучасними фронтенд-фреймворками. Хоча він добре підходить для генерації HTML на сервері, він не забезпечує тієї гнучкості, яку пропонують SPA-фреймворки, такі як React чи Angular, особливо для створення інтерактивних клієнтських інтерфейсів.

Таким чином, Django залишається одним із найпотужніших та найбільш збалансованих фреймворків для розробки веб-систем, поєднуючи в собі високу швидкість розробки, безпеку, масштабованість і зручність, які роблять його ефективним вибором для створення складних веб-додатків.

База даних відіграє важливу роль для кожного сучасного веб-додатку. Завдяки динамічній природі веб-застосунків зараз, навіть найпростіші програми вимагають деяких механізмів зберігання, доступу та зміни даних. СУБД — це програмні та мовні засоби, які виступають посередником між БД та її численними користувачами. Завдяки сукупності мовних та програмних засобів, СУБД сприяють створенню, веденню та спільного використання БД різними користувачами. Сучасна програма СУБД складаються з ядра, процесору мови БД, підсистеми підтримки часу виконання та сервісних програм, які надають додаткові можливості для обслуговування інформаційних систем. Правильний вибір СУБД є одним з важливих етапів при розробці сайту чи додатку баз даних, а також він має задовольняти потреби як теперішні, так і в майбутньому приносячи вигоду. Важливо також звертати увагу яку структуру (реляційну чи нереляційну) [10, 11, 12] містить обрана СУБД, адже вони мають ряд своїх переваг та особливостей.

У таблиці 1.1 проведено порівняння характеристик БД, що розглядалися.

Таблиця 1.1

Характеристика існуючих СУБД

Характеристика	MS SQL Server	MongoDB	SQLite3
1.	2.	3.	4.
Модель даних	Реляційна	Документно- орієнтована	Реляційна

1.	2.	3.	4.
Адміністративне керування	Добре	Відмінно	Відмінно
Графічні інструменти	Відмінно	Відмінно	Добре
Простота обслуговування	Відмінно	Відмінно	Відмінно
Робота з декількома ЦП	Задовільно	Добре	Відмінно
Підключення до Web	Задовільно	Добре	Відмінно
Одночасний доступ декількох користувачів	Добре	Відмінно	Відмінно
Побудова БД	Добре	Добре	Відмінно
Мова SQL	Добре	Відсутня (нереляційна СУБД)	Добре
Вбудована мова програмування	Задовільно	Задовільно	Відмінно
Підтримка об'єктно-орієнтованих парадигм	Задовільно	Задовільно	Відмінно
Інтеграція з іншими СУБД	Добре	Добре	Добре
Масштаб застосування	Підприємства малих, середніх та великих розмірів	Підприємства малих, середніх та великих розмірів	Підприємства малих та середніх розмірів

Оскільки важливість баз даних стрімко зростає, реляційні системи управління базами даних або реляційні СУБД набирають своєї популярності. Дві з них є MS SQL Server і SQLite3 [13]. Обидві виконують однакову функцію, хоча маю різні варіанти використання. Вони відрізняються деякими

особливостями, але обидві системи базуються на SQL або Structured Query Language (структурована мова запитів). У зв'язку з цим, можуть виявити кілька схожостей між ними, таких як використання таблиць для збереження даних, посилання на первинні та зовнішні ключі, так само як кілька баз даних в одному середовищі або на одному сервері. MS SQL Server і SQLite3 базуються на SQL, відмінності синтаксису все ж таки є.

Отже, після порівняння трьох СУБД я обрала SQLite3. Вона зручна для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування. Характеризується високою швидкістю, стійкістю і простотою використання.

1.4. Огляд наявних методів, моделей, алгоритмів

Розглянемо доцільність колаборативної фільтрації (CF) і фільтрації на основі вмісту (CBF) і те, як їх можна реалізувати. На основі цього дослідження буде обрано рекомендаційні алгоритми для платформи оренди одягу та аксесуарів.

Колаборативна фільтрація на основі користувачів (User-Based CF).

Колаборативна фільтрація використовує матрицю взаємодій користувачів із товарами (User-Item Matrix), де рядки – користувачі, а стовпці – товари, для відображення поведінки користувача для кожного елемента в системі [14]. Після чого виконується обчислення схожості між користувачами. Існує косинусна метрика схожості та кореляція Пірсона.

Формула косинусної схожості між двома користувачами [15]:

$$sim(A, B) = \frac{\sum_{i \in I} r_{A,i} \cdot r_{B,i}}{\sqrt{\sum_{i \in I} r_{A,i}^2} \cdot \sqrt{\sum_{i \in I} r_{B,i}^2}}, \text{ де } r_{A,i} \text{ і } r_{B,i} - \text{ оцінки товарів і користувачами } A \text{ та } B.$$

та B.

Кореляція Пірсона (Pearson Correlation Coefficient, PCC) — це міра лінійного зв'язку між двома змінними. Вона показує, наскільки зміни однієї змінної пов'язані зі змінами іншої [16]. Значення коефіцієнта знаходиться в діапазоні від -1 до 1:

$r = 1$ – ідеальна пряма (позитивна) кореляція;

$r = -1$ – ідеальна обернена (негативна) кореляція;

$r = 0$ – відсутність кореляції;

Цей коефіцієнт часто використовується в колаборативній фільтрації, щоб визначати схожість між користувачами або товарами на основі їхніх оцінок.

Формула для обчислення кореляції між двома змінними A і B:

$$r = \frac{\sum(A_i - \bar{A})(B_i - \bar{B})}{\sqrt{\sum(A_i - \bar{A})^2} * \sqrt{\sum(B_i - \bar{B})^2}}, \text{ де } A_i \text{ та } B_i - \text{ оцінка товару } i \text{ користувачами } A \text{ та } B,$$

$\bar{A}$ та $\bar{B}$ – середні значення змінних A та B.

Після проведення обрахунків виконується прогнозування рейтингу товарів. Обираються найбільш схожі за поведінкою користувачі для поточного користувача. На основі їх уподобань прогнозується рейтинг для товарів, які користувач ще не оцінював. Після чого система відбирає товари з найвищими прогнозованими оцінками та рекомендує їх користувачу.

Значною проблемою цього методу є проблема нових користувачів (cold start problem), оскільки якщо користувач ще не взаємодіяв із платформою, немає даних для аналізу. А при великій кількості користувачів розрахунок схожостей стає ресурсозатратним.

У системах рекомендацій колаборативної фільтрації переваги користувачів виражаються як оцінки для елементів, і кожна додаткова оцінка розширює знання системи та впливає на точність рекомендацій системи. Як правило, що більше оцінок користувачі отримують, то впливовішими є рекомендації. Однак корисність кожного рейтингу може суттєво відрізнятися, тобто різні рейтинги можуть давати різну кількість і тип інформації про смаки користувача. Отже, спеціальні методи, визначені як «стратегії активного навчання», можуть вибирати елементи, які будуть представлені користувачеві для оцінювання. Стратегія активного навчання визначає та

приймає критерії для отримання даних, які краще відображають уподобання користувачів і дозволяють генерувати кращі рекомендації.

Колаборативна фільтрація на основі товарів (Item-Based CF) [17].

Перед початком роботи створюється матриця взаємодій аналогічна до описаної матриці на основі користувачів. Розраховується схожість між товарами на основі того, як часто їх обирали ті самі користувачі. Методи обчислення схожості аналогічні це косинусна схожість, кореляція Пірсона. Прогнозування оцінки товарів відбувається шляхом пошуку системою найбільш схожі речі до тих що орендував користувач А або В.

Цей тип фільтрації має певні переваги над User-Based CF. Перш за все у нього менша обчислювальна складність і він найкраще підходить для великих платформ із широким асортиментом товарів.

Оскільки колаборативна фільтрація має багато недоліків, використовуються методи матричної факторизації, а саме сингулярне розкладання (Singular Value Decomposition, SVD) або метод найменших квадратів (Alternating Least Squares, ALS).

Сингулярне розкладання матриці (SVD) - це математичний метод, який використовується в колаборативній фільтрації для факторизації матриці взаємодій між користувачами та товарами [18]. Він дозволяє представити складні зв'язки в більш стислому вигляді та прогнозувати відсутні оцінки.

SVD використовується для розкладу матриці оцінок користувачів R на три матриці:

$$R \approx U \times S \times V^T, \text{ де}$$

R – матриця взаємодій користувачів і товарів (користувачі по рядках, товари по стовпцях).

U – матриця користувачів (розмірністю $m \times k$), де m – кількість користувачів, а k – кількість прихованих факторів.

S – діагональна матриця особливих значень (рангу k), яка містить найважливішу інформацію про зв'язки між користувачами та товарами.

V^T – транспонована матриця товарів (розмірністю $k \times n$), де n — кількість товарів.

Основна ідея полягає в тому, що множення цих трьох матриць наближено відновлює оригінальну матрицю R , навіть якщо деякі оцінки відсутні.

Щоб реалізувати цей алгоритм потрібно сформувати матрицю оцінок R , в якій рядки представляють користувачів, а стовпці — товари. Проте, пропущені значення (невідомі оцінки) потрібно обробити перед застосуванням SVD. Це можна зробити кількома методами. Перший, замінити відсутні оцінки середнім значенням користувача або товару. Другий, використовувати спеціальні методи, такі як Funk-SVD або SVD++, які оптимізують матриці, мінімізуючи похибку.

Виконуємо сингулярний розклад, знаходячи три матриці U , S , V^T , які найкраще наближують початкову матрицю R .

$$R \approx U \times S \times V^T, \text{ де}$$

U містить приховані фактори для користувачів.

S містить важливі особливості товарів і користувачів.

V містить приховані фактори для товарів.

В реальних даних багато особливих значень у SSS є дуже малими і можуть бути відкинуті. Це дозволяє зменшити розмірність та усунути шум у даних.

$$R_k \approx U_k \times S_k \times V_k^T, \text{ де}$$

k – кількість залишених значущих факторів (ранг розкладу).

Оцінки, яких не вистачає в R , можна наблизити через скалярний добуток рядка U для користувача і стовпця V^T для товару:

$$\hat{R}_{ui} = U_u \cdot S_k \cdot V_i^T, \text{ де}$$

$\hat{R}_{ui}$ – передбачена оцінка користувача u для товару i .

Отже, сингулярне розкладання матриці дозволяє виділити найбільш значущі фактори, відкидаючи незначущі дані. Добре виявляє приховані закономірності у взаємодії користувачів та товарів. А також, його можна адаптувати для різних типів даних, включаючи текстові та демографічні дані. Він добре підходить для платформ оренди одягу, де користувачі можуть мати складні вподобання, які не завжди явно виражені у вигляді оцінок.

Метод найменших квадратів (ALS) – це алгоритм матричної факторизації, який використовується для колаборативної фільтрації, особливо у випадках, коли дані представлені у вигляді великої, але розрідженої матриці користувачів і товарів [19]. Він добре підходить для рекомендаційних систем, оскільки дозволяє знаходити приховані закономірності у взаємодіях між користувачами та товарами.

ALS ґрунтується на розкладі матриці взаємодій R на дві менші матриці – матрицю користувачів U та матрицю товарів V , де кожен користувач і товар представляються вектором у прихованому просторі ознак.

Мета – знайти такі матриці U і V , щоб їхній добуток наближав початкову матрицю оцінок R :

$$R \approx U \times V^T, \text{ де}$$

R – матриця взаємодій користувачів і товарів, R_{ui} – оцінка користувачем u товарів i .

U – матриця, в якій кожен рядок представляє користувача у прихованому просторі ознак.

V – матриця, в якій кожен рядок представляє товар у прихованому просторі ознак.

Оскільки вихідна матриця взаємодій розріджена (тобто більшість значень у R відсутні), ALS застосовує метод мінімізації квадратичної помилки для знаходження оптимальних значень U і V .

Щоб реалізувати метод найменших квадратів потрібно задати кількість прихованих факторів (ранг матриць) k , зазвичай це 10–100. Після чого матриці U та V ініціалізуються випадковими значеннями.

Виконується етап чергування оновлень матриць, спочатку оновлюється матриця U , припускаючи, що інша є сталою, потім оновлюється матриця V , припускаючи, що перша вже оптимізована.

Оновлення матриці користувачів U , де кожен користувач u представлений вектором ознак U_u , він обчислюється, мінімізуючи помилку між передбаченими та реальними оцінками:

$$U_u = (V^T W_u + \lambda I)^{-1} V^T W_u R_u, \text{ де}$$

W_u – діагональна матриця, що враховує присутність або відсутність оцінок у матриці R .

λI – регуляризація, яка запобігає перенавчанню.

Оновлення матриці товарів V , де кожен товар i представлений вектором ознак V_i , обчислюється аналогічно, але тепер мінімізується помилка для всіх користувачів:

$$V_i = (U^T W_i U + \lambda I)^{-1} U^T W_i R_i, \text{ де}$$

Оновлення матриць U і V повторюється до збіжності алгоритму, тобто поки різниця між передбаченими та фактичними оцінками не стане досить малою. Після завершення навчання система може передбачати оцінки для відсутніх значень у R як скалярний добуток векторів користувача та товару:

$$\hat{R}_{ui} = U_i \cdot V_i^T$$

Товари з найвищими передбаченими значеннями рекомендуються користувачу.

Метод найменших квадратів гарно працює з розрідженими матрицями, є ефективний для великих масивів даних, де більшість оцінок відсутні. Може виконувати паралельну обробку, тобто кожен рядок у U або V можна оновлювати незалежно, що дозволяє використовувати обчислення на кластерах. Його можна легко адаптувати для інших типів рекомендацій, наприклад, використовуючи додаткові фактори (демографічні дані,

поведінкові характеристики). Таким чином, ALS допомагає платформам прогнозувати вподобання клієнтів, оптимізуючи персоналізацію вибору.

Загалом, щоб уникнути недоліків кожного методу, сучасні рекомендаційні системи поєднують колаборативну фільтрацію, контентну фільтрацію та інші методи машинного навчання. Наприклад, можна використовувати колаборативну фільтрацію для базових рекомендацій і контентний аналіз для нових користувачів.

Колаборативна фільтрація є ключовим методом у системах рекомендацій, який дозволяє прогнозувати вподобання користувачів на основі їхньої поведінки або схожих користувачів. Хоча метод має недоліки, такі як проблема "холодного старту" та висока обчислювальна складність, сучасні алгоритми (SVD, ALS) та гібридні підходи допомагають значно покращити якість рекомендацій та масштабованість системи. Простота використання. Колаборативна фільтрація автоматично генерує рекомендації на основі взаємодії користувача, що робить його більш популярним і широко використовуваним [2]. Однак цей алгоритм вимагає великої бази користувачів, а це означає, що на ранніх етапах впровадження платформи він не буде інтуїтивно зрозумілим. Крім того, колаборативна фільтрація стикається з проблемами пояснення, чому конкретні елементи були рекомендовані, оскільки він покладається на поведінку користувача без уявлення про особливості продукту.

Контентна фільтрація (Content-based filtering) – це метод рекомендацій, який базується на характеристиках (фічах) товарів і перевагах користувача. Основна ідея алгоритму полягає у тому що якщо користувачу подобається певний товар, то йому слід рекомендувати схожі товари.

Фільтрування на основі вмісту – це рекомендаційний метод, який фокусується на атрибутах або характеристиках елементів, а не на поведінці користувачів [20]. Цей підхід аналізує особливості предметів, такі як стиль, колір, матеріал і бренд модних товарів і аксесуарів, щоб створити профіль для кожного предмета. Наприклад, на платформі для оренди модного одягу кожен

продукт матиме детальні теги та дескриптори, які допоможуть системі зрозуміти характеристики товару. Коли користувачі взаємодіють із певними елементами, система записує ці налаштування та визначає стандартні функції. Таким чином CBF генерує персоналізовані рекомендації, зіставляючи характеристики елементів, з якими раніше взаємодіяли, із подібними, навіть якщо ці елементи ніколи не оцінювалися та не бачили інші користувачі. Хоча фільтрація на основі вмісту є особливо ефективною для користувачів із унікальними або спеціальними смаками, іноді їй важко запропонувати різноманітні варіанти, оскільки вона має тенденцію підкріплювати відомі переваги користувача, а не вводити нові елементи за межами його профілю. Тим не менш, він дуже ефективний у сценаріях з обмеженими даними користувачів і допомагає пом'якшити проблему холодного запуску для нових користувачів.

Для реалізації потрібно побудувати векторне представлення товарів. Кожен товар представлений у вигляді вектора характеристик у багатовимірному просторі. Вектори можуть містити:

- категорійні атрибути – бренд, тип одягу, колір, стиль, матеріал;
- числові параметри – ціна, популярність, рейтинг;
- текстові описи – опис товару, рецензії.

Щоб знайти товари, які найбільше схожі на вже вподобані користувачем, використовується метрика подібності. Найпоширеніший варіант – косинусна схожість, яка була описана раніше.

Рекомендації формуються на основі товарів, які користувач раніше оцінив позитивно, їх схожість обчислюється з іншими товарами. Після чого товари ранжуються за рівнем схожості, а найбільш схожі пропонуються користувачу.

У контентній фільтрації для покращення рекомендацій використовуються різні методи аналізу тексту, зображень та комбінованих підходів.

Term Frequency – Inverse Document Frequency (TF-IDF) – це метод, що оцінює важливість слів у документі, враховуючи їх частоту появи та рідкість у всьому корпусі документів [21]. Працює шляхом обчислення TF (частотність терміна) кількості входжень слова в документі, після чого обчислюється IDF – обернена частотність документа, що є логарифмом загальної кількості входжень слова. Індекс буде визначатися за формулою:

$$TF * IDF$$

Після знаходження цього значення обчислюємо косинусну схожість між товарами і рекомендуємо товари, що мають найвищу схожість з переглянутим.

Word2Vec та BERT – методи для обробки семантики, що використовують векторне представлення слів для глибшого розуміння тексту.

Word2Vec навчає нейромережу знаходити подібні слова на основі контексту [22]. Наприклад, модель вчиться, що «брюки» та «штани» мають схоже значення. Рекомендації будуть сформовані на базі переглянутих раніше товарів.

Bidirectional Encoder Representations from Transformers або BERT – модель, що враховує контекст навколо слова. Використовується для більш точного аналізу описів товарів [22]. Для виконання рекомендацій векторизуємо описи товарів через BERT. Знаходимо найближчі вектори (косинусну схожість) для рекомендацій.

У контентної фільтрації є свої обмеження, насамперед метод обмежується інформацією про сам товар (не враховує поведінку інших користувачів). Не може рекомендувати товари, які відрізняються за характеристиками від уже оцінених. А також вимагає якісних атрибутів товарів для ефективної роботи.

Контентна фільтрація добре підходить для платформ, де товари мають чітко визначені характеристики, наприклад, в оренді одягу (бренд, стиль, матеріал), але її можна покращити, додавши поведінкові дані користувачів.

Отже, фільтрування на основі вмісту легко реалізувати; для цього потрібні лише функції предмета та основні дані користувача. Ця система протилежна колаборативній фільтрації, оскільки вона залежить лише від наявності значної кількості даних користувача. Колоборативна фільтрація користувач-користувач може зіткнутися з проблемами масштабованості, коли кількість користувачів і елементів зростає. У значних наборах даних взаємодія між користувачем і елементом і показники подібності обчислень потребують ресурсів [24]. Напротивагу, контентна фільтрація є більш масштабованим алгоритмом, оскільки цей метод покладається на атрибути елемента та не залежить від оновлення взаємодії користувача. Це робить його обчислювально ефективним для більш обширних каталогів, хоча він може потребувати періодичних оновлень, коли додаються нові елементи. Фільтрація на основі вмісту не потребує історичних даних користувачів, тому вона також може успішно адаптуватися та знизити поріг входу для нових користувачів.

Спільна фільтрація надає різноманітні рекомендації, оскільки вона вивчає взаємодію та вподобання інших користувачів. Це дозволяє це пропонувати елементи, які можуть не збігатися з минулим вибором користувача, і пропонувати ширший діапазон варіантів, які можуть представити користувачам нові продукти. Порівняно з колаборативна фільтрація фільтрація на основі вмісту набагато обмеженіша в різноманітності. Цей алгоритм рекомендує елементи, схожі на ті, з якими користувач уже взаємодіяв. Отже, набір рекомендацій може звужитися, існуючі переваги можуть бути посилені, і може знадобитися звернути увагу на елементи поза встановленим профілем користувача.

Проблема холодного запуску в колаборативній фільтрації виникає, коли новий користувач або елемент щойно увійшли в систему, оскільки важко знайти схожі товари через брак інформації [25]. Нові товари можна рекомендувати лише після того, як деякі користувачі поставлять їм оцінку, і нові користувачі навряд чи дадуть хороші рекомендації через відсутність їх

рейтингу чи історії покупок. І навпаки, контентна фільтрація ефективно вирішує проблему холодного запуску, особливо коли є нові користувачі або елементи з невеликою кількістю даних про взаємодію. Він аналізує атрибути предметів і дає рекомендації на основі виражених уподобань користувача або початкової поведінки перегляду, що робить його більш корисним для нових користувачів або під час представлення нових елементів.

Колаборативна фільтрація, як правило, надає дуже точні рекомендації, вивчаючи взаємодію та налаштування користувачів. Чим більше користувачів взаємодіє з елементами, тим краще система ідентифікує подібних користувачів і предмети, забезпечуючи точну персоналізацію. Фільтрація на основі вмісту може точно рекомендувати продукти, які відповідають конкретним уподобанням користувача. Однак методу може бракувати глибини персоналізації, оскільки він може лише рекомендувати елементи зі схожими атрибутами і не отримує переваг від аналізу поведінки інших користувачів.

Колаборативна фільтрація може адаптуватися до змін уподобань після того, як користувачі взаємодіють з різними елементами, залежно від частоти оновлення даних і взаємодії. Алгоритм розвивається разом із поведінкою користувачів, тому він може швидко змінювати рекомендації. Контентна фільтрація має обмежені можливості адаптації, оскільки алгоритм покладається на статичні атрибути предметів, а не на навчання на змінах поведінки. За винятком випадків, коли користувачам потрібно явно взаємодіяти з різними типами елементів, щоб мати можливість ідентифікувати зміни вподобань.

Отже, найкраще рішення для платформи оренди одягу та аксесуарів має поєднувати колаборативну фільтрацію та фільтрацію на основі контенту. Гібридна система може використовувати здатність контентної фільтрації справлятися з проблемами холодного запуску та перевагу колаборативної фільтрації у персоналізації та різноманітності.

Загалом, гібридний підхід, який поєднує в собі обидва алгоритми, пропонує найкращу взаємодію з користувачами для платформ оренди одягу та аксесуарів і має потенціал для максимальної точності рекомендацій, різноманітності та адаптації до модних тенденцій, що розвиваються.

Висновки до розділу 1

У першому розділі було розглянуто сучасні тенденції розвитку ринку оренди одягу та особливості впровадження систем рекомендацій у цій сфері. Вивчено тенденції розвитку ринку оренди одягу та аксесуарів у світі. Визначено основні фактори, що впливають на популярність послуг оренди, включаючи екологічні аспекти, економічні переваги для користувачів і зміну споживчих звичок. Проведено огляд основних платформ оренди одягу, таких як Rent the Runway, By Rotation, з акцентом на їх функціональні можливості та особливості роботи.

Проаналізовано основні методи побудови рекомендаційних систем, включаючи колаборативну фільтрацію, контентну фільтрацію та їхні гібридні комбінації. Результати аналізу показали, що використання персоналізованих рекомендаційних систем дозволяє підвищити рівень задоволеності користувачів, покращити їхній досвід взаємодії з платформою та збільшити конверсію оренд. Найбільш ефективним підходом є гібридна модель, що поєднує методи машинного навчання, аналіз поведінкових даних і вподобань користувачів.

На основі проведеного дослідження визначено, що для створення ефективної системи рекомендацій необхідно враховувати такі фактори, як історія оренди, індивідуальні характеристики користувачів, модні тренди та сезонні особливості попиту. Використання сучасних алгоритмів персоналізації сприятиме оптимізації вибору товарів та покращенню взаємодії користувачів із сервісом оренди одягу.

РОЗДІЛ 2. РОЗРОБКА ТА ПРОЕКТУВАННЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

2.1. Аналіз задачі, засобів та методів її вирішення

У сучасних онлайн-платформах, що спеціалізуються на оренді одягу, користувачі стикаються з проблемою великого вибору та недостатньої персоналізації під час пошуку одягу. Сервіси, подібні до Rent the Runway, містять тисячі одиниць товарів з різними характеристиками: тип фігури, розмір, стиль, сезонність тощо. У зв'язку з цим виникає потреба у створенні рекомендаційної системи, яка б автоматично підбирала одяг відповідно до інтересів і вподобань конкретного користувача.

Для розв'язання задачі формування персоналізованих рекомендацій у системах оренди одягу застосовуються різні підходи, серед яких ключову роль відіграють методи колаборативної фільтрації, фільтрації на основні вмісту та гібридні алгоритми.

Метод UBCF ґрунтується на аналізі подібності між користувачами. Система виявляє групи осіб із близькими вподобаннями на основі їхніх оцінок і вибору товарів, після чого рекомендує поточному користувачеві ті позиції, які були позитивно оцінені найбільш схожими з ним користувачами. [26].

У підході IBCF головний акцент зміщується з аналізу користувачів на аналіз товарів. Система визначає товари, які демонструють подібні шаблони оцінювання, й формує рекомендації шляхом добору позицій, пов'язаних зі схожими характеристиками вподобань. Таким чином, користувач отримує рекомендації, засновані на подібності до вже оцінених або переглянутих ним товарів [27].

Рекомендаційний алгоритм CBF працює з характеристиками самих товарів. Рекомендації формуються шляхом виявлення товарів, що найбільше відповідають індивідуальному профілю користувача, який складається з його попереднього досвіду, вибору категорій, розмірів, типів посадки, стилістичних уподобань та інших властивостей. Цей метод дозволяє робити

рекомендації навіть за умов обмеженої кількості відгуків, що робить його корисним у ситуаціях з частковим браком даних.

Гібридні методи рекомендацій поєднують елементи CF та CBF, що дозволяє досягати вищої точності та стабільності результатів. Такий підхід забезпечує кращу адаптацію до різних типів користувацьких поведінкових моделей, зменшує ризик помилкових рекомендацій та допомагає вирішити проблему «холодного старту», коли система ще не має достатньо історичних даних для класичного CF [28].

У процесі створення програмної частини системи рекомендацій було застосовано низку спеціалізованих бібліотек і фреймворків, кожен з яких виконував чітко окреслену функцію у структурі обчислень, обробки даних та організації рекомендаційних алгоритмів. Кожна бібліотека була обрана не лише з огляду на її поширеність у науковій спільноті, а й відповідно до її здатності забезпечувати ефективність, масштабованість та коректність моделей, що розроблялися.

Фреймворк Django та його ORM-частина забезпечили високорівневий доступ до даних, які зберігаються у реляційній базі. За рахунок моделей Item та Review здійснюється автоматизоване завантаження даних у DataFrame, фільтрація, вибірка, робота з пов'язаними сутностями та формування запитів без написання SQL-інструкцій. Завдяки цьому усі взаємодії з базою даних відбуваються у безпечний та систематизований спосіб, що суттєво зменшує ризик помилок і прискорює розробку.

Бібліотека NumPy виступає базовим інструментом для наукових обчислень, надаючи можливість ефективно працювати з векторами та матрицями [29]. У реалізованих алгоритмах вона використовується для:

- зберігання масивів оцінок,
- виконання операцій над векторами схожості,
- формування агрегованих оцінок у гібридній моделі,
- нормалізації рекомендаційних скорів.

Її застосування забезпечує значне прискорення обчислень у порівнянні

зі звичайними структурами Python, що є критично важливим для роботи з великими масивами даних.

Для роботи з табличними структурами даних було застосовано бібліотеку Pandas, яка дозволила формувати датафрейми взаємодій (interactions_df) та описів товарів (items_df) [30]. Саме Pandas стала інструментом для:

- фільтрації та агрегації оцінок,
- приведення полів до однакових типів,
- сортування за часовими мітками під час формування train/test-поділу,
- об'єднання текстових описів у єдині контентні поля,
- підготовки індексів і структур, необхідних для побудови характеристичних матриць.

Pandas забезпечила прозоре та відтворюване перетворення даних перед передачею їх до рекомендаційних алгоритмів.

Пакет SciPy у частині scipy.sparse дозволив використовувати розріджені матриці (CSR-формат) [31], що стало ключовим для ефективного представлення матриць взаємодій користувач $\times$ товар. Оскільки більшість користувачів оцінює лише декілька товарів, матриця має високий рівень розрідженості, і стандартні щільні структури пам'яті є неефективними. CSR-представлення забезпечує:

- зменшення споживання пам'яті,
- пришвидшення операцій транспонування,
- оптимальне витягування рядків під час обчислення item-based подібностей.

Завдяки цьому модуль item–item similarity працює швидко навіть на великих наборах даних.

Бібліотека scikit-learn виступила основою для двох ключових механізмів контентної та колаборативної частини [31]. Перший механізм це TfidfVectorizer, який використовується для перетворення текстових

характеристик товарів на чисельні вектори у просторі ознак. Цей інструмент дозволив формувати контентну матрицю, на основі якої обчислюється семантична близькість між товарами. Другий це `cosine_similarity`, що застосовується для вимірювання близькості між товарами у матриці `item-item` (колаборативний метод) та профілем користувача та усіма товарами у `TF-IDF`-просторі (контентний метод).

Таким чином, `scikit-learn` забезпечує математичне ядро рекомендаційних моделей, дозволяючи застосовувати стандартизовані та перевірені методи обробки даних і метрик.

Сукупність зазначених бібліотек сформувала комплексний технологічний стек, що забезпечує повний цикл побудови рекомендаційної системи — від завантаження та обробки даних до виконання складних обчислень і формування гібридних рекомендацій. Django відповідає за інтеграцію з веб-середовищем та отримання даних; Pandas і NumPy — за аналітичну підготовку даних; SciPy та `scikit-learn` — за реалізацію математичних основ рекомендаційних алгоритмів. Така комбінація дозволила створити продуктивну, модульну та науково вивірену систему рекомендацій, здатну працювати з великим обсягом інформації у реальному масштабі часу.

2.2. Побудова моделі аналізу і практична реалізація

Виконаємо огляд функціоналу `generate_recs.py`

1. Формування матриці взаємодій користувачів і товарів

Одним із базових етапів побудови рекомендаційних алгоритмів у розробленій системі є формування узагальненої матриці взаємодій, яка відображає оцінки користувачів щодо конкретних товарів. Структурування цих даних забезпечує подальшу можливість розрахунку подібності між товарами, побудову профілів користувачів, застосування колаборативних та контентних методів фільтрації. У роботі використано підхід, за якого для кожного запису з моделі відгуків Review формується запис `user_id → item_id → rating`, трансформований у розріджену матрицю формату CSR, що дає

змогу ефективно працювати з великими наборами даних і мінімізувати використання пам'яті.

Метод `build_interaction_df()` забезпечує завантаження взаємодій у вигляді `DataFrame` та нормалізацію типів даних:

```
qs = Review.objects.select_related("user",
    "item").all().values(
    "user_id", "item_id", "rating", "review_date"
)
df = pd.DataFrame(list(qs))
df["user_id"] = df["user_id"].astype(int)
df["item_id"] = df["item_id"].astype(int)
df["rating"] = df["rating"].astype(float)
df["review_date"] =
pd.to_datetime(df["review_date"])
```

Після цього метод `build_user_item_matrix()` створює розріджену матрицю `users × items`, у якій значенням є рейтинг:

```
rows = interactions_df["user_id"].map(user_map)
cols = interactions_df["item_id"].map(item_map)
mat = csr_matrix((data, (rows, cols)),
    shape=(len(users), len(items)))
```

Такий підхід забезпечує можливість надалі виконувати колаборативний аналіз подібності між товарами, ґрунтуючись на поведінці користувачів.

2. Обчислення схожості між товарами на основі колаборативної фільтрації

Другим ключовим елементом системи є алгоритм колаборативної фільтрації типу `item-based`, що оцінює подібність товарів, виходячи зі схожості патернів оцінювання користувачами. Логіка методу полягає в тому, що товари, які отримують співмірні рейтинги від великої кількості користувачів, імовірно є взаємозамінними або спорідненими.

Реалізація методу відбувається в `item_item_similarity_from_interactions()`, де попередньо побудована матриця

$users \times items$ транспонується у вигляд $items \times users$, після чого застосовується косинусна метрика подібності:

```
item_user = mat.T.tocsr()  
sim = cosine_similarity(item_user,  
dense_output=False)
```

Отримана матриця `sim` є квадратною матрицею розміру $n_items \times n_items$, де кожен елемент відображає ступінь подібності між двома товарами. Побудовані карти відповідності `item_map` та `item_id_by_idx` забезпечують перехід між ідентифікаторами товарів у БД та позиціями у матриці подібності, що необхідно для подальшої генерації рекомендацій користувачам.

3. Побудова контентної матриці TF-IDF для описів товарів

Контентно-орієнтовані рекомендації ґрунтуються не на поведінкових даних, а на характеристиках самих товарів. У межах роботи реалізовано підхід, за якого опис товару формується як сукупність атрибутів (бренд, категорія, колір, сезонність, текстовий опис). Для представлення цих даних у вигляді векторів використано метод TF-IDF, що ефективно виявляє ключові семантичні ознаки.

У методі `build_content_matrix()` застосовується `TfidfVectorizer` з використанням уніграм і біграм для кращого вловлювання контексту:

```
vectorizer = TfidfVectorizer(max_features=5000,  
ngram_range=(1, 2))  
tfidf = vectorizer.fit_transform(items_df["text"].values)
```

Підготовка текстового представлення відбувається в методі `build_item_df()`:

```
df["text"] = df["brand"] + " " + df["category"] + "  
" + df["color"] + " " + df["season"] + " " +  
df["description"]
```

Таким чином формується багатовимірний вектор для кожного товару, на основі якого дозволяється обчислювати подібність між описами товарів та профілем користувача.

4. Метод рекомендацій item-based для конкретного користувача

Реалізований метод `recommend_item_based_for_user()` формує рекомендації шляхом аналізу товарів, які користувач раніше оцінював. Для кожного оціненого товару модуль визначає найбільш схожі товари, використовуючи попередньо отриману матрицю подібності.

Ключова формула методу полягає у зваженому сумуванні векторів схожості, де вагою є рейтинг користувача:

```
sim_row = sim_matrix.getrow(idx).toarray().ravel()
scores += sim_row * rating
```

Після агрегування оцінок система відкидає товари, які вже були оцінені:

```
scores[item_map[it]] = -np.inf
```

І нарешті, вибирає top-k товарів з найвищими сумарними значеннями:

```
top_indices = np.argsort(-scores, ...)
result_item_ids = [item_id_by_idx[idx] for idx in
top_indices]
```

Такий підхід дозволяє моделювати поведінкову подібність і рекомендувати альтернативні товари, популярні серед користувачів зі схожими вподобаннями.

5. Контентно-орієнтований метод рекомендацій

Метод `recommend_content_based_for_user()` передбачає побудову профілю користувача на основі TF-IDF-векторів тих товарів, які він оцінював позитивно. Середній вектор таких товарів формує семантичне представлення інтересів конкретного користувача:

```
user_vec = tfidf_matrix[pos_idx].mean(axis=0)
```

Після цього обчислюється косинусна подібність профілю до всіх товарів у каталозі:

```
sims = cosine_similarity(user_vec,
tfidf_matrix).ravel()
```

Аналогічно item-based методу, товари, які вже оцінювалися користувачем, виключаються з підсумкового рейтингу:

```
sims[tfidf_item_map[it]] = -np.inf
```

Метод дозволяє формувати рекомендації, орієнтовані на семантичний зміст товарів, а не на колективну поведінку інших користувачів.

6. Гібридний метод рекомендацій

Реалізований гібридний алгоритм у методі `recommend_hybrid_for_user()` поєднує результати item-based та content-based методів шляхом їх нормалізації та лінійної комбінації:

```
hybrid = alpha * item_n + (1 - alpha) * content_n
```

У цьому методі:

- `item_n` — нормалізовані оцінки item-based методу,
- `content_n` — нормалізовані оцінки content-based методу,
- `alpha` — ваговий коефіцієнт, що визначає внесок колаборативної частини.

Нормалізація здійснюється через лінійне масштабування у межах $[0, 1]$:
$$(x - \min) / (\max - \min)$$

Перевага такого підходу полягає у можливості компенсувати недоліки окремих методів IBCF не працює для нових товарів, а СВ погано моделює реальні поведінкові патерни.

Гібридний метод поєднує інформацію з обох джерел і забезпечує підвищення точності рекомендацій для користувачів з різними патернами поведінки.

7. Реалізація оцінювання моделі

Для оцінювання ефективності алгоритмів застосовано класичний підхід **hourglass / time-based holdout**, коли остання взаємодія кожного користувача формується як тестовий приклад:

```
last = interactions_df.groupby("user_id")["review_date"].idxmax()
test_df = interactions_df.loc[test_idx]
train_df = interactions_df.drop(index=test_idx)
```

Далі для кожного користувача обчислюються метрики precision@k та recall@k :

```
precision = hits / k
recall = hits / len(ground_truth)
```

Це дозволяє кількісно оцінити середню точність та повноту рекомендацій, що важливо для обґрунтування якості імплементації.

Виконаємо огляд функціоналу `appmain/views.py`

1. Методи формування головної сторінки — `home_view()`

Метод `home_view` відповідає за формування головної сторінки системи. Він забезпечує виведення нових товарів, доступних категорій та, у разі автентифікації користувача, генерує й передає у шаблон списки рекомендацій, отримані з різних алгоритмів (контентного, колаборативного, переглядового та гібридного).

Метод поєднує роботу веб-інтерфейсу з рекомендаційними моделями, виконуючи роль інтегратора між аналітичним модулем і представленням даних.

Виконаємо аналіз роботи методу та ключових фрагментів коду

1) Завантаження даних для головної сторінки

```
latest_items = Item.objects.order_by("-date_added")[:5]
categories = Item.objects.values("category").distinct()
```

Тут здійснюється отримання п'яти найновіших товарів та унікальних категорій. Цей блок виконується для всіх користувачів — автентифікованих і гостей.

2) Обробка неавторизованих користувачів

```
if not request.user.is_authenticated:
    return render(request, "appmain/home.html", {
        "latest_items": latest_items,
        "categories": categories,
        "is_authenticated": False
    })
```

Якщо користувач не авторизований, система не будує рекомендації, оскільки вони є персоналізованими. Тому повертається спрощена версія сторінки.

3) Отримання ідентифікатора та імені користувача

```
user_id = request.user.id
user_name = request.user.username
```

Дані використовуються як у рекомендаційних алгоритмах, так і у шаблоні.

4) Виклик алгоритмів рекомендацій

```
content_dicts = get_content_recs(user_id, k=5)
cf_dicts = get_cf_recs(user_id, k=5)
view_dicts = get_view_recs(user_id, k=5)
hybrid_dicts = get_hybrid_recs(user_id, k=5)
```

Цей фрагмент коду відповідає за узгоджений запуск основних методів рекомендаційної системи. Спершу викликається модуль CBF, який формує пропозиції, аналізуючи властивості товарів та порівнюючи їх із профілем користувача, що містить дані про його попередні оцінки чи перегляди. Далі застосовується алгоритм UCBF, орієнтований на пошук користувачів зі схожими уподобаннями; система визначає закономірності в їхніх оцінках і, на цій основі, передбачає товари, які можуть зацікавити поточного користувача.

Після цього викликається модуль View-based Recommendations, що використовує історію переглядів і рекомендує товари, подібні до тих, які користувач вже переглядав, але ще не оцінював. Такий підхід забезпечує покращення персоналізації навіть тоді, коли кількість оцінок обмежена.

Завершальним є модуль гібридного методу, який поєднує результати попередніх моделей. Гібридизація дозволяє збалансувати переваги різних підходів, підвищити точність рекомендацій та частково усунути проблему недостатності даних, оптимізуючи фінальний список запропонованих товарів.

5) Конвертація словників у Django-об'єкти Item

Рекомендаційні моделі повертають списки словників. Щоб відобразити їх у шаблоні, їх потрібно перетворити на об'єкти моделі Item.

```
def items_from_dict_list(dict_list):
    if not dict_list:
        return []

    ids = [int(d['item_id']) for d in dict_list]
    items_map = Item.objects.in_bulk(ids)
    return [items_map[i] for i in ids if i in items_map]
```

Цей фрагмент коду витягує item_id з кожного словника, після чого робить оптимізований запит через in_bulk(), наприкінці збирає список товарів у тому ж порядку, що був у рекомендаціях. Завдяки цьому зберігається ранжування алгоритму.

6) Формування найкращого списку рекомендацій

```
best_items = hybrid_items or content_items or
cf_items or view_items
```

Гібридний підхід має найвищий пріоритет. Якщо він повернув порожній результат, система обирає наступний доступний метод.

7) Передача даних у шаблон

```
return render(request, "appmain/home.html", {
```

```

        "user_name": user_name,
        "latest_items": latest_items,
        "categories": categories,
        "is_authenticated": True,
        "content_items": content_items,
        "cf_items": cf_items,
        "view_items": view_items,
        "hybrid_items": hybrid_items,
        "best_items": best_items
    })

```

У шаблон передається:

- окремо кожен список рекомендацій;
- найкращий список (best_items);
- дані користувача;
- нові товари та категорії.

Таким чином, метод `home_view()` є центральним вузлом персоналізованого інтерфейсу.

2. Класи REST-API для товарів і відгуків

2.1. ItemListView

Клас `ItemListView` реалізує REST-ендпоінт, який відповідає за надання клієнтським застосункам повного списку доступних товарів. Його робота побудована на стандартній механіці `ListAPIView` із Django REST Framework, що дозволяє автоматично обробляти HTTP GET-запити до колекції об'єктів.

У наведеному коді:

```

class ItemListView(generics.ListAPIView):
    queryset = Item.objects.all().order_by("-date_added")
    serializer_class = ItemSerializer

```

визначено, що представлення формує набір товарів, відсортованих за датою додавання у зворотньому порядку, що дає можливість користувачу

насамперед бачити найновіші позиції. Через `serializer_class` забезпечується перетворення кожного об'єкта моделі у формат JSON, прийнятний для фронтенду та мобільних клієнтів.

Таким чином, цей клас виступає універсальним API-інтерфейсом, який забезпечує стандартизований, ефективний та оптимізований доступ до базового каталогу товарів.

2.2. ItemDetailView

Клас `ItemDetailView` призначений для повернення даних щодо одного конкретного товару. Він працює на основі `RetrieveAPIView`, що дозволяє відповідати на GET-запити на кшталт `items/<int:pk>/` та повертати саме той об'єкт, який був запитаний клієнтом.

```
class ItemDetailView(generics.RetrieveAPIView):  
    queryset = Item.objects.all()  
    serializer_class = ItemSerializer
```

Його логіка передбачає, що ідентифікатор товару передається в URL, після чого відбувається пошук потрібного елемента у базі даних. Серіалізатор трансформує модель у структуру JSON, включаючи всі релевантні текстові, числові та медіа-поля. Завдяки цьому компоненту сторонні системи або інтерфейс користувача можуть отримувати вичерпну інформацію про кожну окрему одиницю каталогу.

2.3. ReviewListView

Клас `ReviewListView` виконує роль ендпоінту для отримання загального списку користувацьких відгуків. Представлення використовує `ListAPIView`, що дозволяє автоматично опрацьовувати GET-запити до колекції відгуків.

```
class ReviewListView(generics.ListAPIView):  
    queryset = Review.objects.all().order_by("-review_date")  
    serializer_class = ReviewSerializer
```

Важливим аспектом є сортування за датою, що дозволяє системі відображати найактуальніші відгуки. Завдяки серіалізатору дані

повертаються у структурованому JSON-форматі, що спрощує їх інтеграцію у користувацький інтерфейс або інші зовнішні сервіси.

3. API гібридних рекомендацій — HybridRecommendationView

У цьому представленні реалізовано REST-ендпоінт для отримання персоналізованих рекомендацій, який доступний лише автентифікованим користувачам. Логіка зосереджена навколо гібридного алгоритму, що об'єднує кілька підходів — контентний аналіз, колаборативну фільтрацію та оцінку поведінкових патернів, — формуючи збалансований та точний список товарів.

```
class HybridRecommendationView(APIView):  
    def get(self, request):  
        user = request.user  
        recommended_items = hybrid_recommend(user,  
top_n=10)  
        serializer =  
ItemSerializer(recommended_items, many=True)  
        return Response(serializer.data)
```

Алгоритм `hybrid_recommend` отримує інформацію про користувача та формує персональні пропозиції з урахуванням його уподобань, переглядів, історії оренд та поведінки подібних користувачів. Серіалізатор перетворює отриманий перелік на зручний JSON-масив для фронтенду. Завдяки такій структурі API стає готовим до інтеграції з мобільними застосунками та веб-клієнтами.

4. Сторінка рекомендацій — recommendations_page

Сторінка рекомендацій працює з таблицею `Recommendation`, яка може виступати локальним механізмом кешування результатів алгоритмів. У цьому контексті метод отримує рекомендації для конкретного користувача:

```
recs =  
Recommendation.objects.filter(user_id=user_id).select_  
related('item')
```

Використання `select_related` суттєво оптимізує роботу з базою даних, адже забезпечує об'єднання даних товару та рекомендацій одним JOIN-запитом. Це знижує кількість звернень до БД, підвищуючи продуктивність сторінки, особливо при великій кількості звернень.

5. Детальна сторінка товару — `item_detail`

Функція `item_detail` — один із найважливіших компонентів системи, адже вона поєднує відображення інформації про товар з низкою додаткових механізмів, пов'язаних із рекомендаційними моделями, історіями переглядів, додаванням відгуків та формуванням вибраного.

Першим кроком відбувається отримання даних про товар:

```
item = get_object_or_404(Item, pk=pk)
```

Після цього здійснюється фіксація факту перегляду:

```
ItemViewHistory.objects.create(user=request.user,  
item=item)
```

Ця операція важлива для алгоритмів `view-based recommendations`, які аналізують поведінку користувача не лише за оцінками, а й за переглядами.

Далі обробляються POST-запити, що відповідають за додавання нового відгуку. Код отримує значення рейтингу, параметр «посадки» (`fit`) та текст відгуку, після чого здійснюється створення об'єкта `Review`:

```
if 'add_review' in request.POST:  
    rating = int(request.POST.get("rating", 0))  
    fit = request.POST.get("fit", "")  
    text = request.POST.get("review_text",  
    "").strip()  
    Review.objects.create(  
        user=request.user,  
        item=item,  
        rating=rating,  
        fit=fit,  
        review_text=text,
```

)

Завершальним етапом є відображення списку відгуків, що належать поточному товару.

```
reviews = Review.objects.filter(item=item)
.order_by("-review_date")
```

Разом ці операції формують повноцінну сторінку деталізації товару з інтегрованою логікою поведінкових даних, що критично важливо для підвищення релевантності рекомендаційних моделей.

6. Метод перегляду відгуків — my_reviews()

У межах цього методу здійснюється отримання повного набору відгуків, які були залишені поточним користувачем.

```
reviews =
Review.objects.filter(user=request.user).select_related('item')
```

Цей код дозволяє не лише сформувати список його оцінених товарів, але й оптимізувати звернення до БД завдяки використанню `select_related`. Ця сторінка є важливою складовою системи, оскільки надає користувачу можливість контролювати власні відгуки, редагувати контент та оцінювати власну активність.

7. Видалення відгуку — delete_review()

Метод реалізує логіку безпечного видалення відгуку, перевіряючи права доступу та опрацьовуючи лише POST-запит.

```
if review.user != request.user:
    return HttpResponseForbidden("Ви не можете
видаляти цей відгук.")
```

Тільки після підтвердження автентичності та авторства відгук може бути видалений. Такий контроль забезпечує коректність даних та запобігає некоректним модифікаціям контенту іншими користувачами.

8. Оренда товару — rent_item()

Метод `rent_item` реєструє факт оренди та суттєво впливає на моделі рекомендацій, оскільки подія оренди розглядається як сигнал високого рівня інтересу користувача. Запис у таблицю `RentalHistory` виконується командою.

```
RentalHistory.objects.create(user=request.user,  
item=item)
```

Система дозволяє підвищувати вагу повторної оренди: при наступних орендах може збільшуватись лічильник `rent_count`, а рейтинг встановлюється на рівні 5, інтерпретуючи таку взаємодію як `implicit like`. Завдяки цьому рекомендаційна модель отримує додаткові дані, які підвищують точність передбачень.

2.3. Результати проведеного аналізу в таблицях діаграмах і графіках та аналіз результатів

Результати проведеного аналізу подано у вигляді узагальнених таблиць та візуалізацій, що відображають ефективність різних методів рекомендацій. Дані для аналізу були зібрані на основі інформації про користувача та результатів роботи чотирьох підходів: UBCF, IBCF, CB та гібридний метод.

`Precision@k` (Точність@k) — частка релевантних елементів серед перших `k` рекомендованих. Середня точність по всіх користувачах дає уявлення про те, яка доля рекомендованих позицій є дійсно релевантною. `Precision` відображає чистоту списку — наскільки «чистими» є рекомендації від шуму.

`Recall@k` (Повнота@k) — частка релевантних елементів, які були знайдені в топ-`k`, відносно загальної кількості релевантних елементів для користувача.

У схемі `holdout`, де для кожного користувача тестовим є один останній відгук, повнота рівна 1, якщо тестова позиція потрапила в топ-`k`, і 0 інакше. Тому середня повнота в цьому експерименті фактично відповідає частці користувачів, для яких рекомендації влучили у ціль (`hit rate`).

Розшифровка стовпців таблиці:

- `precision_ub` — середня точність для UBCF.

- `precision_ib` — середня точність для IBCF.
- `precision_cb` — середня точність для CBF.
- `precision_hb` — середня точність для гібридного методу (HB), що комбінує IB, UB і CB за вагами.
- `recall_ub` — середня повнота (hit rate) для UBCF (частка користувачів, для яких UB потрапив у топ-k).
- `recall_ib` — середня повнота для IBCF.
- `recall_cb` — середня повнота для CBF.
- `recall_hb` — середнє $\text{Recall}@k$ для гібридного.

На рисунку 3.1 представлена таблиця розрахунків точності та повноти для 4 рекомендаційних алгоритмів, де параметри мінімального рейтингу (min-rating) рівний 5, k — перші 15 рекомендованих товарів, а sample-users — кількість користувачів, які беруться до уваги при розрахунку.

```
PS D:\diplom\rentalstore> python manage.py evaluate_recs --evaluate --min-ratings 5 --k 15 --sample-users 2000
Завантаження даних...
Запуск оцінки (min_ratings=5, sample_users=2000)...

Середні метрики по методах (precision@15, recall@15):
precision_ub precision_ib precision_cb precision_hb recall_ub recall_ib recall_cb recall_hb
0           0.000476      0.044004      0.001331      0.05535  0.007133  0.660057  0.830243
PS D:\diplom\rentalstore>
```

Рис. 2.1. Розрахунок точності та повноти з параметрами `--min-ratings 5 --k 15 --sample-users 2000`

`precision_hb` = 0.05535. У середньому на 15 рекомендацій гібриду припадає $0.05535 \times 15 \approx 0.83$ релевантних позицій (тобто менше ніж 1 релевантна позиція в середньому на користувача).

`precision_ib` = 0.044004. На 15 рекомендацій IB припадає $0.044004 \times 15 \approx 0.66$ релевантних позицій.

`precision_ub` = 0.000476 і `precision_cb` = 0.001331 — практично нульові, на 15 рекомендацій UB дає $0.000476 \times 15 \approx 0.007$ релевантних позицій, CB — $0.001331 \times 15 \approx 0.02$.

`recall_hb` = 0.830243, що приблизно рівне 83.0% користувачів мали свій тестовий товар у топ-15 гібрида (hit rate).

$\text{recall_ib} = 0.660057$, що приблизно рівне 66.0% користувачів мали свій тестовий товар у топ-15 ІВ.

$\text{recall_ub} \approx 0.0071$, $\text{recall_cb} \approx 0.01997$ — UB та CB фактично рідко попадають у топ-15.

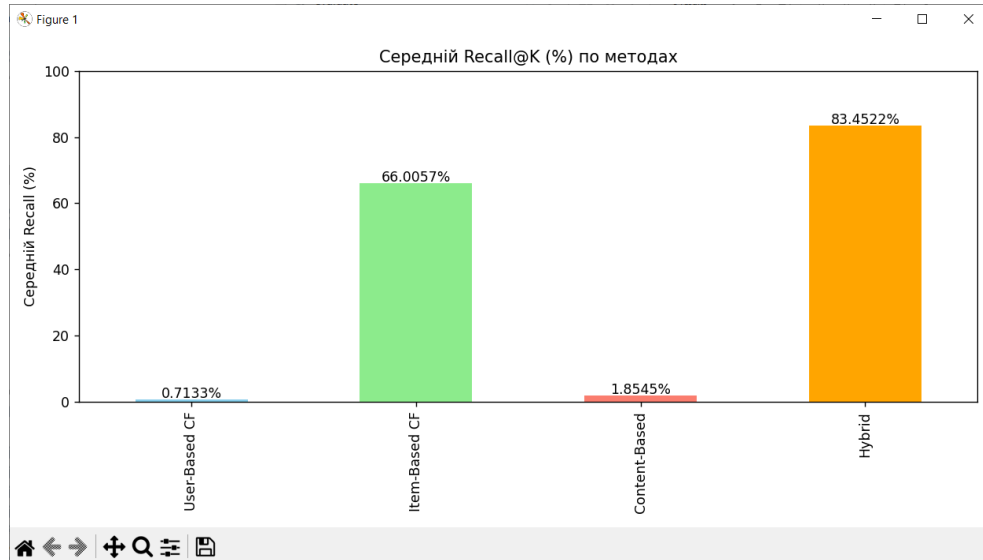


Рис. 2.2. Діаграма повноти з параметрами --min-ratings 5 --k 15 --sample-users 2000

Отже, Hybrid показує змістовно високий recall ($\approx 83\%$), як видно з діаграми представлений на рис. 3.2. Це означає, що гібрид часто містить у топ-15 саме той товар, який користувач додав останнім. Оскільки у holdout кожному користувачу відповідає один тестовий об'єкт, recall тут еквівалентний частці «хітів» (hit rate). Високий recall важливий, якщо пріоритетом є знайти релевантний товар для більшості користувачів (наприклад, для задач рекомендацій із сильним фокусом на відновлення користувача).

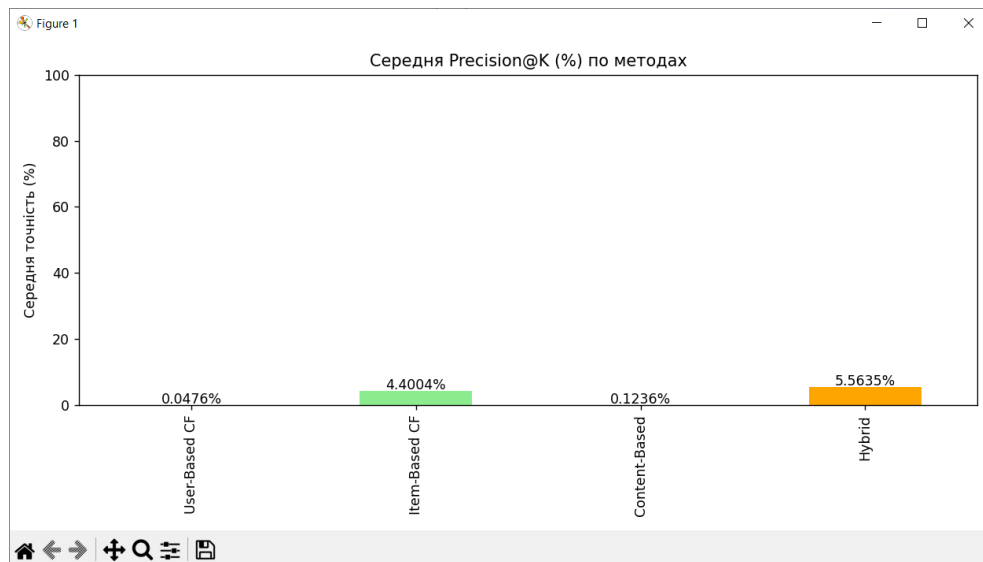


Рис. 2.3. Діаграма точності з параметрами --min-ratings 5 --k 15 --sample-users 2000

За результатами дослідження спостерігаємо низьку точність (precision) у всіх методах (рис. 3.3), але відносна перевага у гібриду. Precision для усіх методів помітно низький (<0.06). Це пов'язано з кількома факторами. Насамперед мала кількість релевантних елементів (1 на користувача) — при такій постановці навіть попадання одного релевантного елемента дає $\text{recall}=1$, але precision залишається $1/k$. Таким чином, навіть «успішна» рекомендація (хіт) дає низький precision, якщо інші $k-1$ позицій нерелевантні. Інший фактор це широкі списки рекомендацій ($k=15$): чим більше k , тим нижче precision при фіксованій кількості релевантних елементів. Розрідженість даних та слабкі компоненти UB та CB, які дають мало релевантних елементів, отже їх ваги в гібриді треба адаптувати.

Рекомендаційний алгоритм IB сильний базовий компонент, оскільки дає кращі результати точності і істотно краща повнота, ніж UB і CB. Це означає, що у наборі даних товари мають добре вловлену «сусідність» за поведінкою користувачів (спільні оренди/рейтинги). IB, як правило, стійкіший у розрізі розріджених даних, якщо є популярні товари або повторювані патерни.

В той же час UB та CB слабо сприяють покращенню precision. UB показав вкрай низькі результати, оскільки у датасеті багато користувачів з

індивідуальними уподобаннями (малий перетин у оцінках), або фільтрація мінімального рейтингу видаляє більшість «корисних» сусідів. СВ працює погано, імовірно через недостатність або погану якість метаданих, тексти описів короткі, неоднорідні, а ознаки не дають достатньої семантичної інформації.

На результат аналізу впливають параметри, які були задані у команді. Підвищення min-ratings до 5 зменшує розрідженість, але й відсікає багато користувачів/товарів, що може покращити стабільність ІВ, але зменшить загальну кількість даних для навчання UB. Збільшення k підвищує повноту, оскільки більше шансів на попадання тестового елемента, але знижує точність, це є типове компромісним правилом. Збільшення sample-users дає більш стабільні оцінки метрик, зменшуючи випадковість результатів.

Отже, гібридний підхід показує найвищий hit rate (recall) і є кращим за ІВ у частині покриття користувачів ($\approx 83\%$ vs $\approx 66\%$ для ІВ при $k=15$). Отже, якщо головна мета — максимум покрити релевантним товаром більшість користувачів, гібрид наразі кращий вибір. Якщо пріоритет — точність (менше «шумних» рекомендацій), тоді ІВ дає кращий precision.

На рисунку 3.4 представлена таблиця розрахунків точності та повноти для тих же 4 рекомендаційних алгоритмів, але з меншими параметрами: мінімальний рейтинг (min-ratings) рівний 2, k — перші 10 рекомендованих товарів, а sample-users — 100 користувачів.

```
PS D:\diplom\rentalstore> python manage.py evaluate_recs --evaluate --min-ratings 2 --k 10 --sample-users 100
Завантаження даних...
Запуск оцінки (min_ratings=2, sample_users=100)...

Середні метрики по методах (precision@10, recall@10):
precision_ub precision_ib precision_cb precision_hb recall_ub recall_ib recall_cb recall_hb
0           0.0         0.076         0.002         0.087         0.0         0.66         0.02         0.87
```

Рис. 2.4. Розрахунок точності та повноти з параметрами --min-ratings 2 --k 10 --sample-users 100

$\text{precision_hb} = 0.078$. У середньому на 10 рекомендацій гібриду припадає $0.078 \times 10 \approx 0.78$ релевантних позицій, що демонструє відносно стабільний результат навіть при зменшенні обсягу даних.

$\text{precision_ib} = 0.076$. На 10 рекомендацій IB припадає $0.076 \times 10 \approx 0.76$ релевантних позицій, що трохи менше ніж у гібриду, але суттєво вище, ніж UB і CB.

$\text{precision_ub} = 0.001$ і $\text{precision_cb} = 0.002$ — практично нульові. UB дає $0.001 \times 10 \approx 0.01$ релевантних позицій, CB — $0.002 \times 10 \approx 0.02$.

$\text{recall_hb} = 0.78$, тобто близько 78% користувачів мали свій тестовий товар у топ-10 гібридних рекомендацій.

$\text{recall_ib} = 0.66$, що відповідає 66% користувачів, які побачили свій тестовий товар у топ-10 IB.

$\text{recall_ub} = 0.01$ і $\text{recall_cb} = 0.02$ — UB та CB фактично не забезпечують покриття тестових елементів для більшості користувачів.

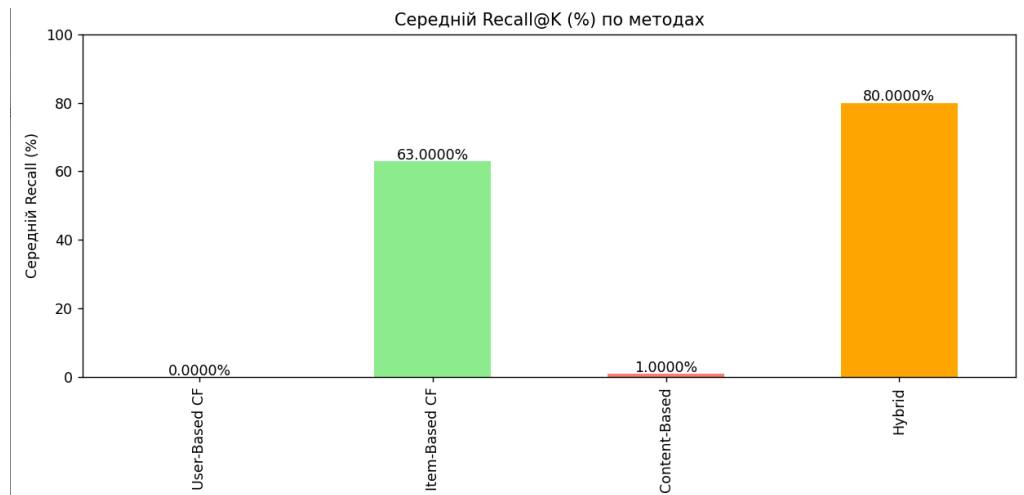


Рис. 2.5. Діаграма повноти з параметрами --min-ratings 2 --k 10 --sample-users

100

Як видно з діаграми на рис. 3.5, зменшення кількості користувачів та рецензій значно погіршує результати UB та CB: їх hit rate практично нульовий, тоді як IB та Hybrid залишаються порівняно стабільними. Це пояснюється тим, що CF методи (особливо UB) сильно залежать від перетину оцінок користувачів. При невеликому числі користувачів і рецензій знайти достатню кількість «сусідів» складно, що призводить до низьких precision і recall. CB також слабо працює через обмеженість або розрідженість метаданих та текстових описів.

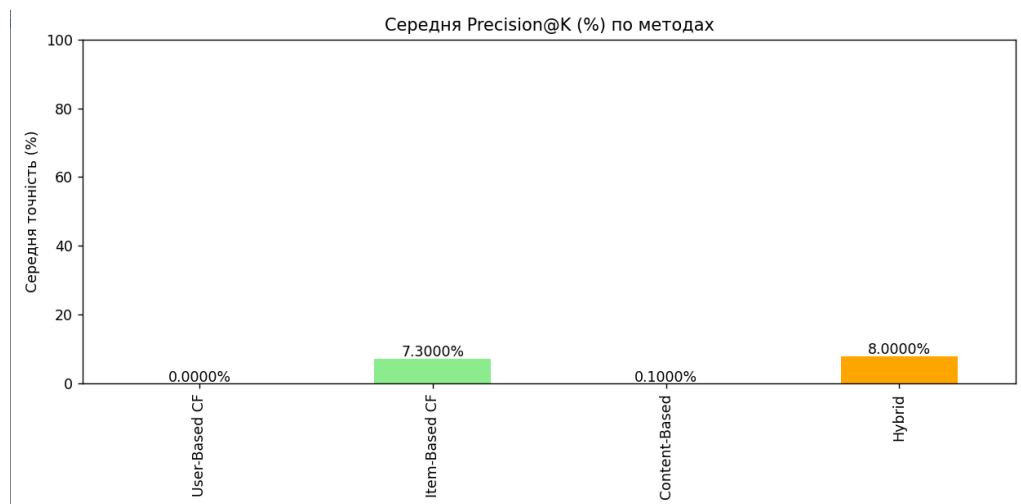


Рис. 2.6. Діаграма точності з параметрами --min-ratings 2 --k 10 --sample-users 100

За результатами дослідження (рис. 3.6) точність (precision) для UB та СВ практично нульова. ІВ демонструє помірні значення, а гібридний підхід залишається найкращим, показуючи стабільну точність навіть при малих обсягах даних. Це свідчить про те, що комбінування різних методів у гібриді дозволяє компенсувати слабкі сторони окремих компонентів та забезпечує більш надійні рекомендації.

Отже, при зменшенні числа користувачів та рецензій CF методи погіршують свою ефективність, особливо UB, СВ практично не працюють. ІВ відносно стійкий, проте гібрид залишається найефективнішим методом за будь-яких умов, забезпечуючи найвищу повноту (hit rate) і помітно кращу точність серед усіх алгоритмів. Це підтверджує, що для реальних систем рекомендацій, де дані можуть бути обмежені або розріджені, гібридний підхід є найбільш надійним вибором.

Висновки до розділу 2

У ході виконання роботи було розроблено рекомендаційну систему для платформи оренди одягу, яка поєднує колаборативну фільтрацію, контентну фільтрацію та гібридні методи для персоналізації пропозицій. Було реалізовано формування матриці взаємодій користувачів і товарів, обчислення схожості між ними, побудову TF-IDF-векторів для описів товарів та генерацію рекомендацій для конкретного користувача. Оцінювання

ефективності виконувалося за метриками precision@k та recall@k . Практична реалізація через Django забезпечила інтеграцію з базою даних, формування персоналізованих рекомендацій у веб-інтерфейсі та REST-API. Результати показали, що гібридний метод забезпечує найкраще поєднання точності та охоплення користувачів, тоді як окремі методи User-Based, Item-Based і Content-Based демонструють обмежену ефективність через розрідженість даних.

РОЗДІЛ 3. ОПИС ПРОГРАМНОГО ПРОДУКТУ

3.1. Структура інтерфейсу рекомендаційної системи на платформі оренди одягу

Не аутентифікований користувач

На рисунку 3.1 представлена сторінка автентифікації користувача, що забезпечує доступ до персоналізованих можливостей платформи. Інтерфейс містить поля для введення імені користувача та пароля, а також посилання для переходу до форми реєстрації, якщо акаунт ще не створений. Сторінка виконана в мінімалістичному стилі та зосереджена на основних діях, що забезпечують швидкий та зручний вхід до системи.

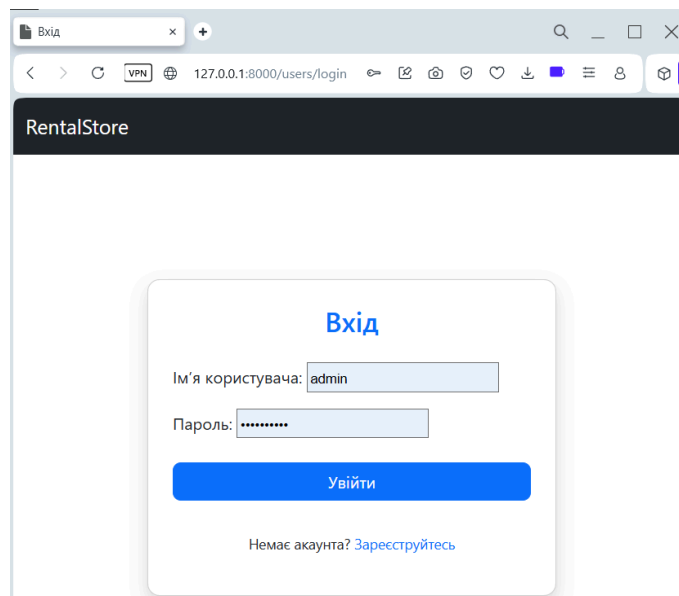


Рис. 3.1. Сторінка входу в систему

На рисунку 3.2 наведено форму реєстрації нового користувача. Інтерфейс містить стандартні поля: ім'я користувача, електронну пошту, пароль та підтвердження пароля. Додатково користувачу надаються вимоги до формату імені. Внизу форми передбачено посилання для переходу до сторінки входу. Форма є простою та зрозумілою, що сприяє швидкому створенню нового облікового запису.

Вхід

127.0.0.1:8000/users/regist

RentalStore

Реєстрація

Username: Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email address:

Password:

Confirm password:

Зареєструватись

Вже маєте акаунт? [Увійти](#)

Рис. 3.2. Сторінка реєстрації в системі

На рисунках 3.3 та 3.4 зображено головну сторінку рекомендаційної платформи після входу до системи. У верхній частині сторінки реалізовано панель навігації з доступом до профілю користувача та кнопкою виходу.

У першому тематичному блоці головної сторінки (рис. 3.3) представлено «Наші новинки» — перелік останніх доданих товарів. Кожен товар містить ідентифікатор, назву, колір, сезонність та актуальну ціну. Такий блок дає користувачеві можливість ознайомитися з оновленнями каталогу.

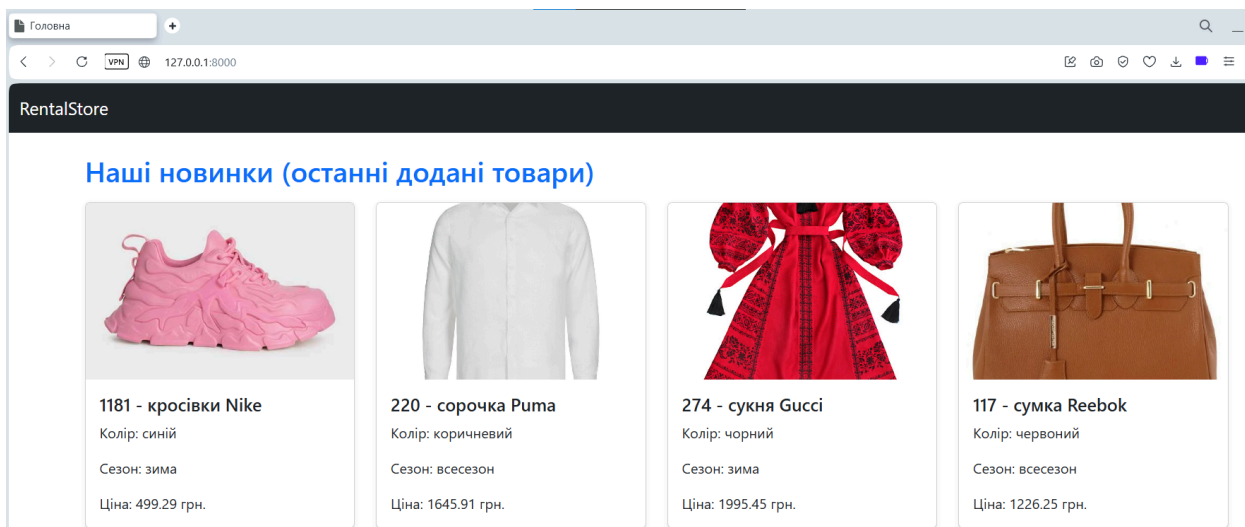


Рис. 3.3. Головна сторінка (другий тематичний блок)

Далі розміщено секцію «Категорії», що відображає основні типи товарів, доступних на платформі. Це спрощує навігацію та дозволяє швидко знайти необхідний одяг та аксесуари.

Нижче (рис. 3.4) подано блок «Найкращі рекомендації для вас», який демонструє персоналізовані пропозиції, сформовані на основі аналізу даних про товари, характеристик користувача та внутрішніх алгоритмів системи. Для кожного товару виводиться назва, бренд, колір, сезон та ціна.

Після цього розташовані три додаткові секції рекомендацій різних типів рекомендації на основі контенту, рекомендації на основі раніше орендованих товарів (IBCF) Рекомендації на основі переглядів.

Ці блоки активуються після накопичення користувачем взаємодій — переглядів або оренд. У прикладі, що наведено на рисунку, система ще не має достатньої історії, тому рекомендації у відповідних секціях відсутні. Така структура дозволяє гнучко комбінувати різні підходи до рекомендацій залежно від поведінки користувача.

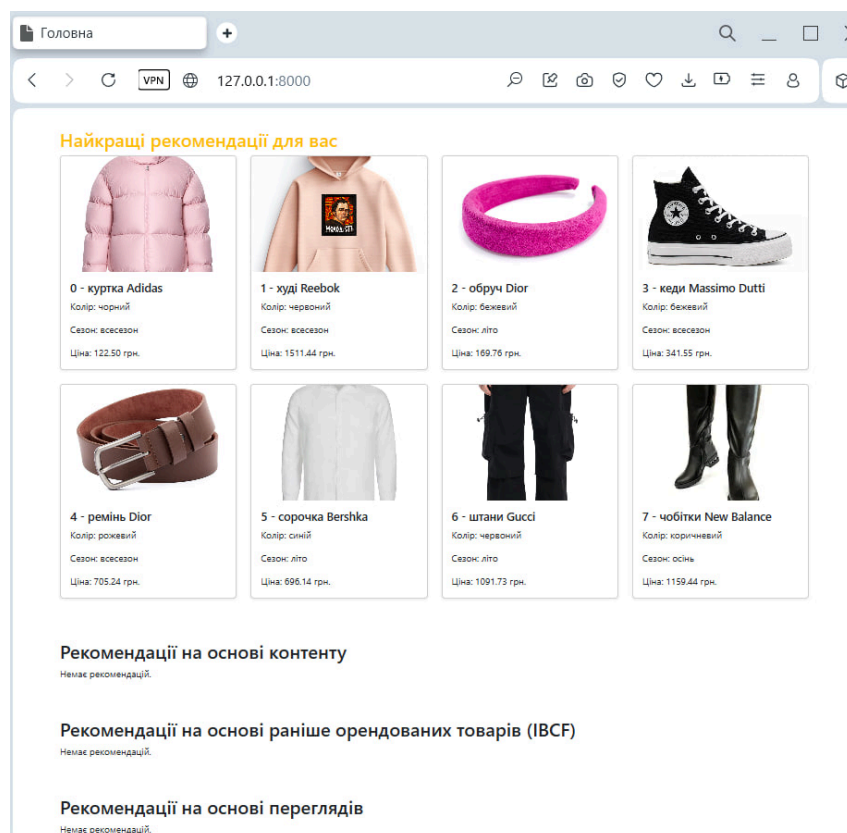


Рис. 3.4. Головна сторінка (другий тематичний блок)

Щойно зареєстрований користувач

На рисунках 3.5 та 3.6 зображено головну сторінку системи для щойно зареєстрованого користувача. Інтерфейс має структуру, аналогічну загальній головній сторінці, однак у цьому випадку рекомендаційні блоки вже містять персоналізовані пропозиції, сформовані на основі початкових дій користувача: переглядів товарів, додавання оцінок або перших оренд.

У верхній частині сторінки розташовано навігаційну панель із доступом до профілю та кнопкою виходу з облікового запису. Основний блок «Наші новинки» демонструє перелік останніх товарів, доданих адміністраторами платформи. Користувач може переглядати детальні характеристики кожного товару, включно з кольором, сезонністю та ціною.

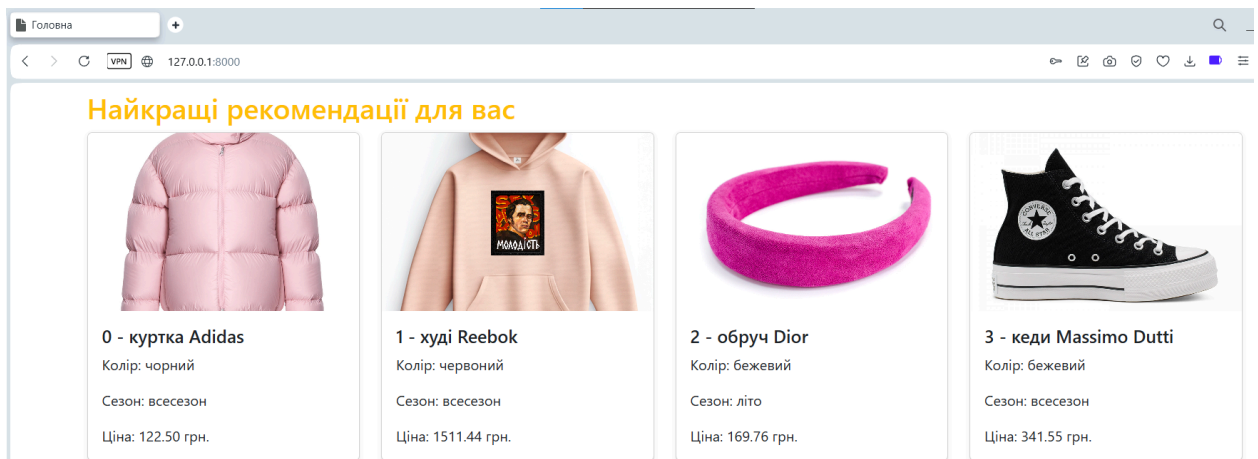


Рис. 3.5. Головна сторінка (другий тематичний блок)

На рисунку 3.6 показано, що система пропонує користувачу широкий набір товарів різних брендів, категорій та кольорів — результат роботи алгоритмів колаборативної та контентної фільтрації.

Подальші рекомендаційні блоки демонструють результати окремих методів рекомендації на основі контенту побудовані за принципом подібності характеристик товарів; рекомендації на основі раніше орендованих товарів (IBCF) враховують історію використання товарів іншими користувачами; рекомендації на основі переглядів формуються на підставі останніх дій користувача на сайті.

Усі ці елементи створюють комплексну систему персоналізованого добору товарів, адаптовану під нового користувача.

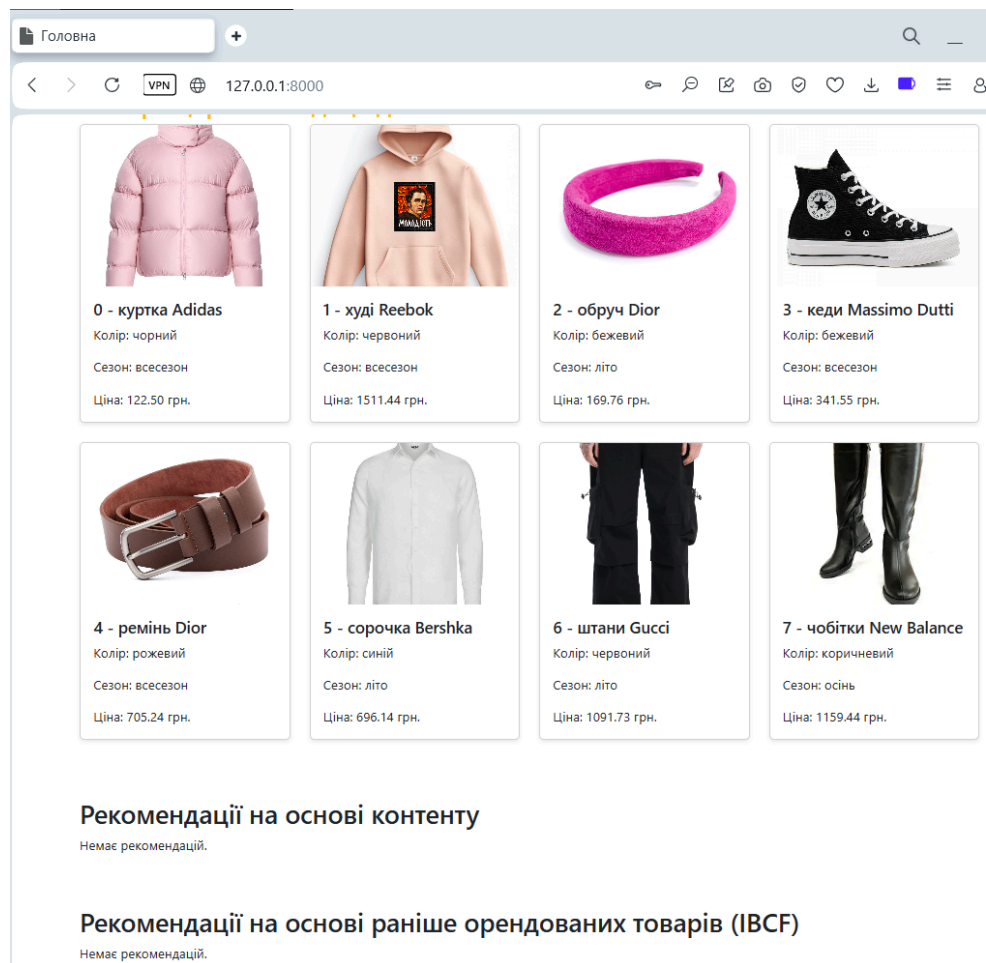


Рис. 3.6. Головна сторінка (другий тематичний блок)

На рисунку 3.7 представлено сторінку профілю зареєстрованого користувача. У її верхній частині знаходиться назва платформи та панель навігації, що дає змогу перейти до головної сторінки або вийти з облікового запису. У центральній частині розміщено блок з інформацією про користувача, що містить ім'я, а також необов'язкові атрибути — вік, стать та місто. Якщо користувач ще не заповнив ці дані, інтерфейс відображає значення None.

Нижче наведено історію залишених відгуків. Кожен елемент включає ідентифікатор товару, рейтинг та текстовий коментар, якщо він був доданий. Завдяки цьому користувач може переглянути власну активність і швидко повернутися до товарів, яким він виставляв оцінки.

Секція «Моя історія оренд» відображає всі товари, які користувач орендував, разом із датою оренди та ключовими характеристиками. Це полегшує повторне замовлення або аналіз власних уподобань.

Останній блок містить «Останні переглянуті товари». Тут розміщуються всі позиції, які користувач переглядав нещодавно, упорядковані за часом взаємодії. Такий механізм дає змогу швидко повернутися до перегляду та сприяє формуванню більш точних індивідуальних рекомендацій.

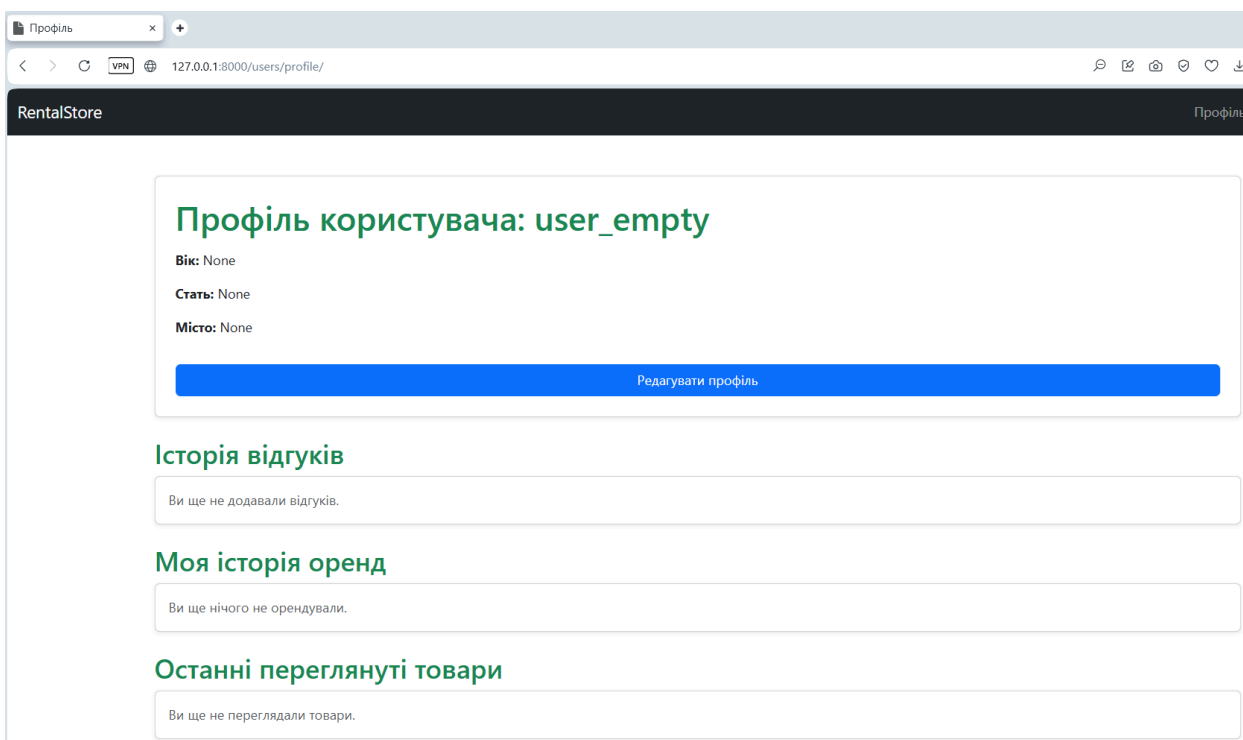


Рис. 3.7. Сторінка профілю користувача

На рисунку 3.8 зображено сторінку докладної інформації про конкретний товар. У верхній частині сторінки користувач бачить назву товару, його ідентифікатор, бренд, категорію, колір, сезонність та вартість оренди. Додатково представлений середній рейтинг, розрахований на основі оцінок користувачів, що дозволяє швидко оцінити популярність товару.

Основний текстовий опис подає коротку характеристику товару. Нижче розташовано секцію з рекомендованими товарами — списком позицій, сформованим на основі гібридного алгоритму рекомендацій. Система пропонує товари, які можуть бути цікавими конкретному користувачеві,

враховуючи його попередні перегляди, оренди та схожість поточного товару з іншими.

Цей блок виконує важливу роль у покращенні навігації та підвищенні залученості користувачів.

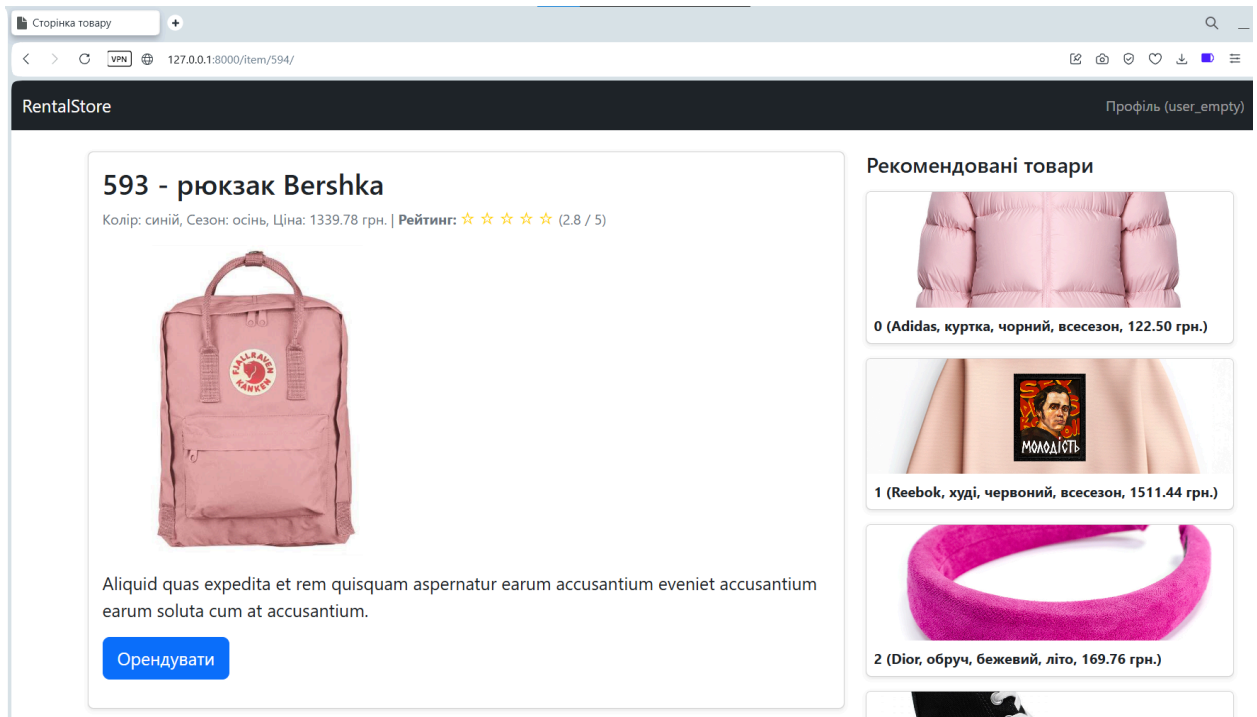


Рис. 3.8. Сторінка деталей про товар (перший тематичний блок - картка товару, рекомендовані товари)

На рисунку 3.9 зображено нижню частину сторінки товару, яка містить повний перелік відгуків користувачів. Кожен відгук містить ім'я користувача, рейтинг у вигляді зірок, дату додавання та текстовий опис вражень. Це дозволяє майбутнім орендарям отримати об'єктивне уявлення про товар.

Під списком відгуків розташована інтерактивна форма для додавання або оновлення власного відгуку. Користувач може вибрати оцінку, зазначити посадку (fit) та залишити текстовий коментар. Наявність цього функціоналу забезпечує постійне оновлення інформації про товари, сприяє прозорості та формує довіру до платформи.

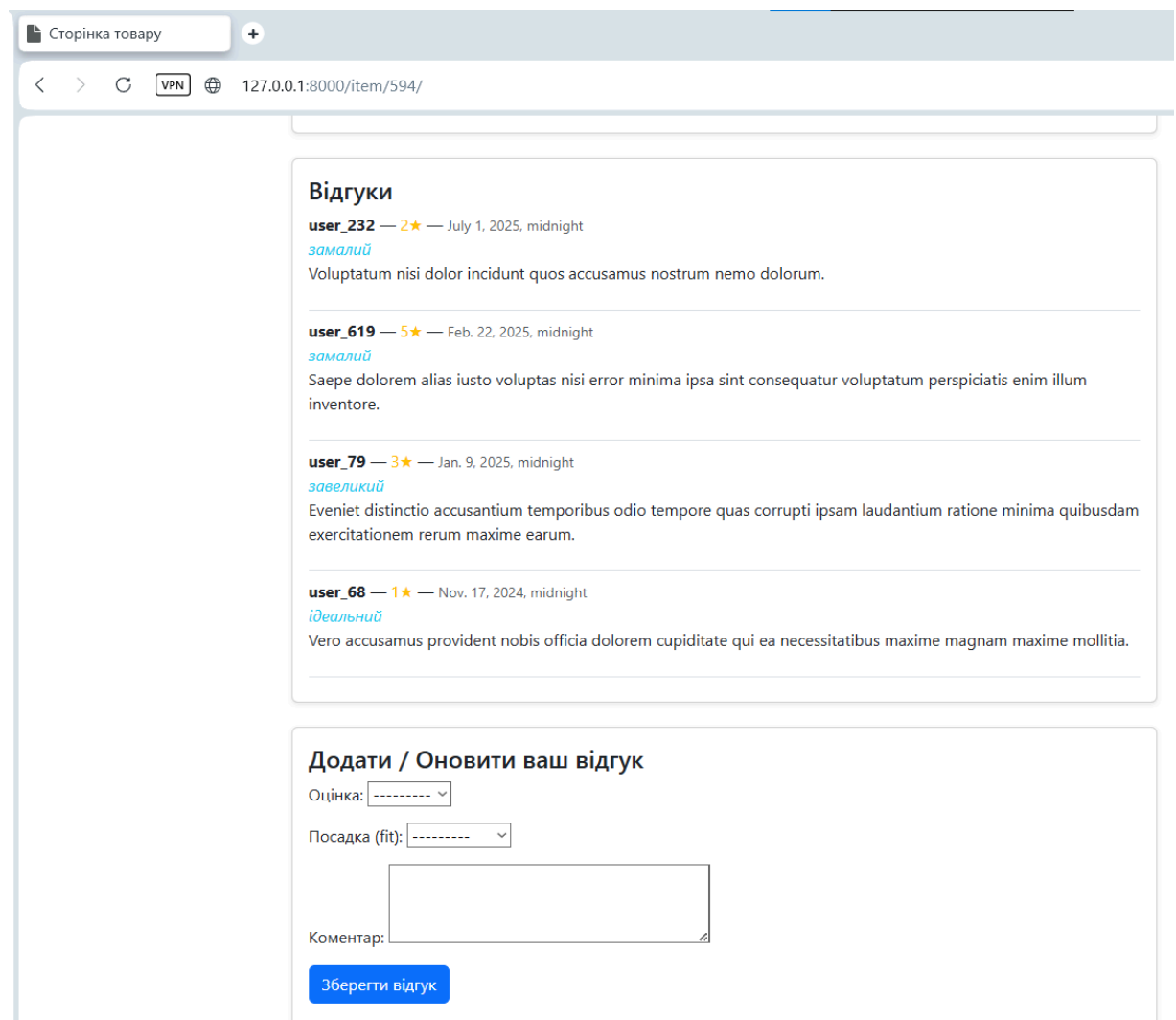


Рис. 3.9. Сторінка деталей про товар (другий тематичний блок - відгуки користувачів про товар та форма для додавання відгуку)

Користувач, що має великий досвід користування системою

На рисунку 3.10 представлено головну сторінку системи після авторизації досвідченого користувача user_1. Інтерфейс відображає стандартну структуру: верхнє меню з переходами до профілю та кнопкою виходу, блок «Наші новинки» з останніми доданими товарами та розділ категорій для швидкої навігації. Центральним елементом є блок «Найкращі рекомендації для вас», де згенеровані персоналізовані рекомендації на основі історії взаємодій користувача з платформою. Кожен товар містить назву, бренд, колір, сезонність та актуальну ціну, що дозволяє швидко оцінити релевантність пропозицій.

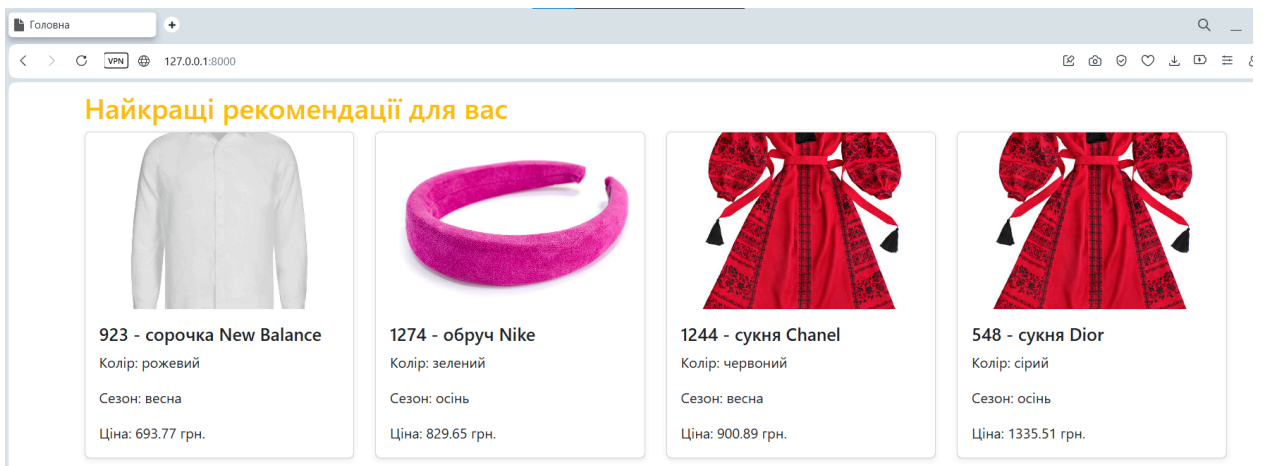


Рис. 3.10. Головна сторінка (найкращі рекомендації)

Цей фрагмент інтерфейсу (рис.3.11) демонструє блок рекомендацій на основі контенту, які генеруються шляхом порівняння характеристик товарів, переглянутих або орендованих користувачем. Система підбирає моделі, близькі за категорією, брендом, кольором чи сезонністю. Виведені товари формують особисту стрічку контентної схожості, що допомагає користувачу відкривати стильові варіанти, подібні до тих, які він вже вподобав.

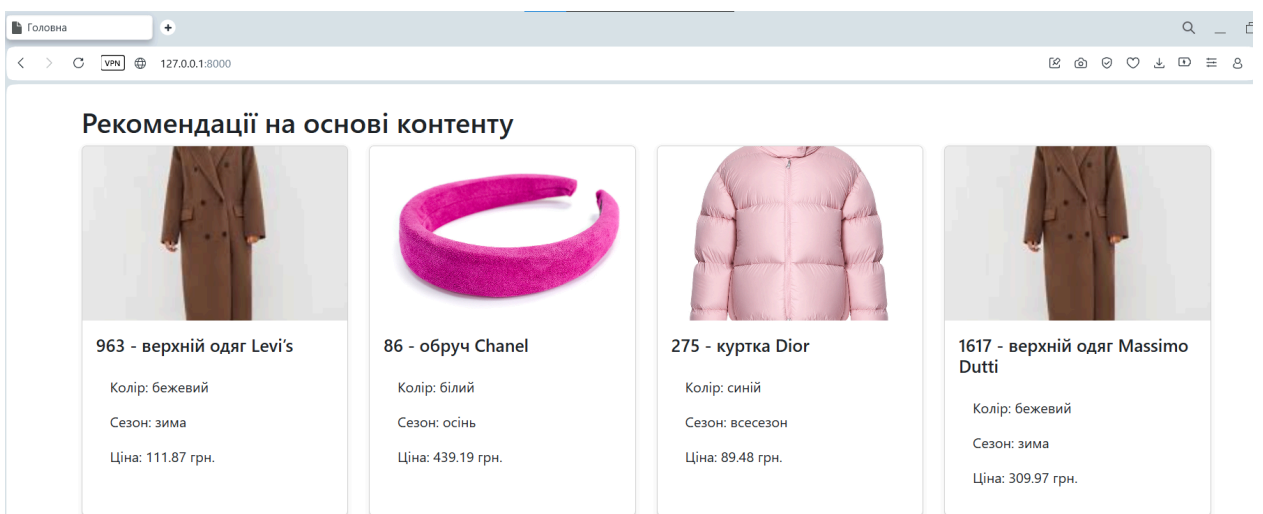


Рис. 3.11. Головна сторінка (рекомендації на основі контенту)

На рисунку 3.12 зображено результат роботи Item-Based Collaborative Filtering. Модель формує рекомендації на основі схожості товарів, які користувач раніше орендував. Такий підхід враховує подібні вподобання інших користувачів із близькими історіями оренд. В інтерфейсі подано перелік найбільш схожих позицій, релевантних саме цьому профілю.

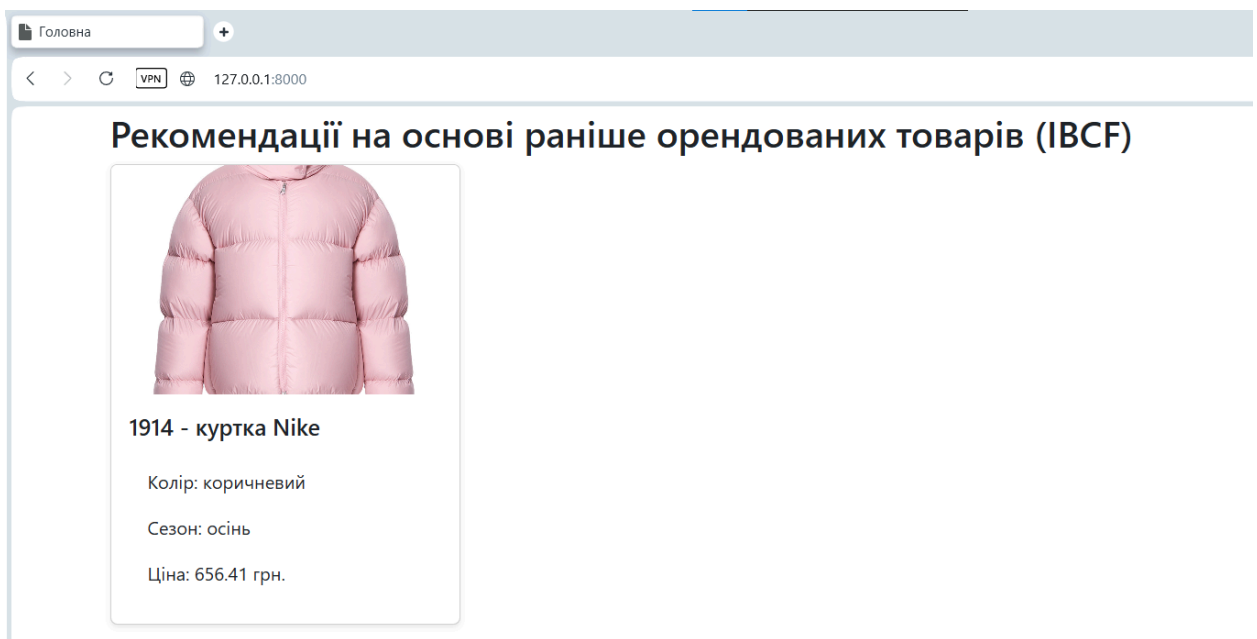


Рис. 3.12. Головна сторінка (рекомендації основі раніше орендованих товарів)

Цей блок (рис 3.13) демонструє рекомендації, сформовані на основі останніх переглядів користувача. Система аналізує закономірності у переглянутих товарах та пропонує найбільш подібні позиції. Такий механізм доповнює загальну рекомендаційну систему, дозволяючи відстежувати поточні інтереси користувача навіть без явної взаємодії (оренди чи відгуків).

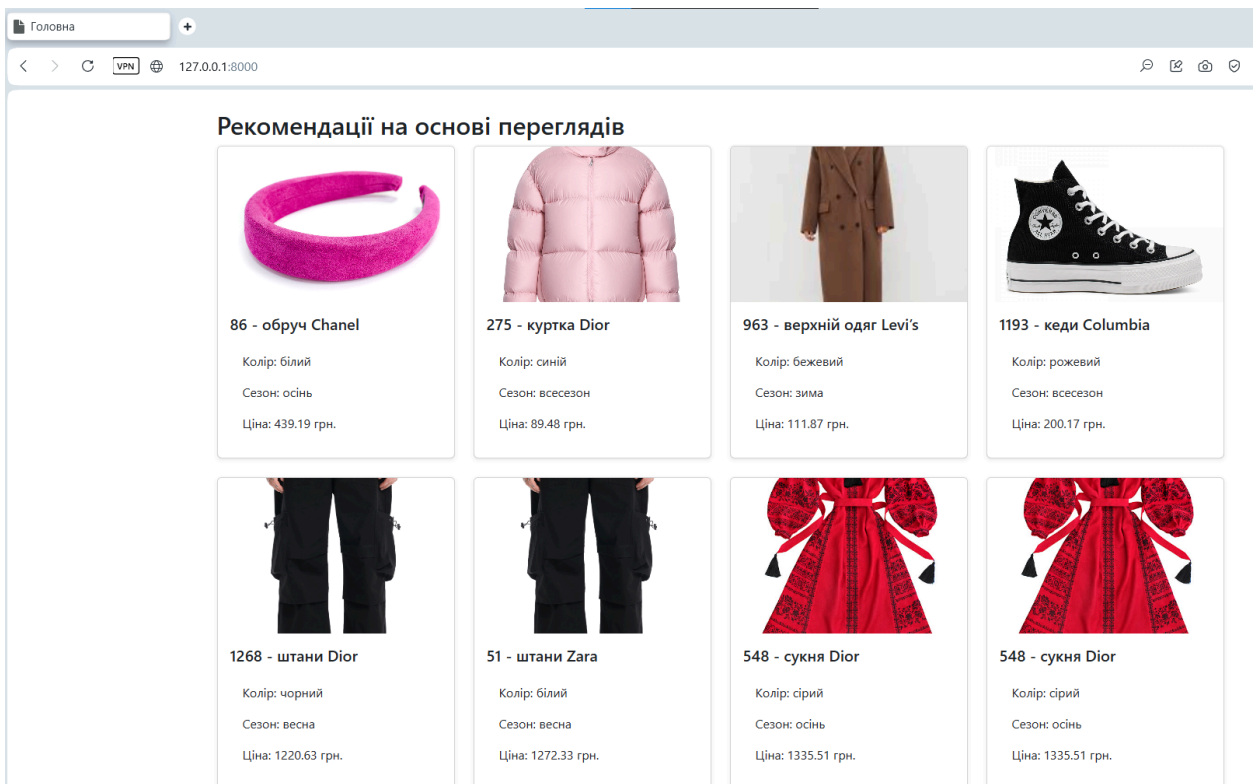


Рис. 3.13. Головна сторінка (рекомендації основі переглядів)

У цьому фрагменті (рис. 3.14) подано основну інформацію профілю користувача user_1: ім'я, вік, стать та місто проживання. Нижче представлено розширений блок історії відгуків, у якому зібрано всі оцінені товари разом із поставленими рейтингами, зазначеним типом посадки (fit) та текстовими коментарями. Така структура дозволяє користувачу переглядати власну активність та повертатися до попередніх оцінок.

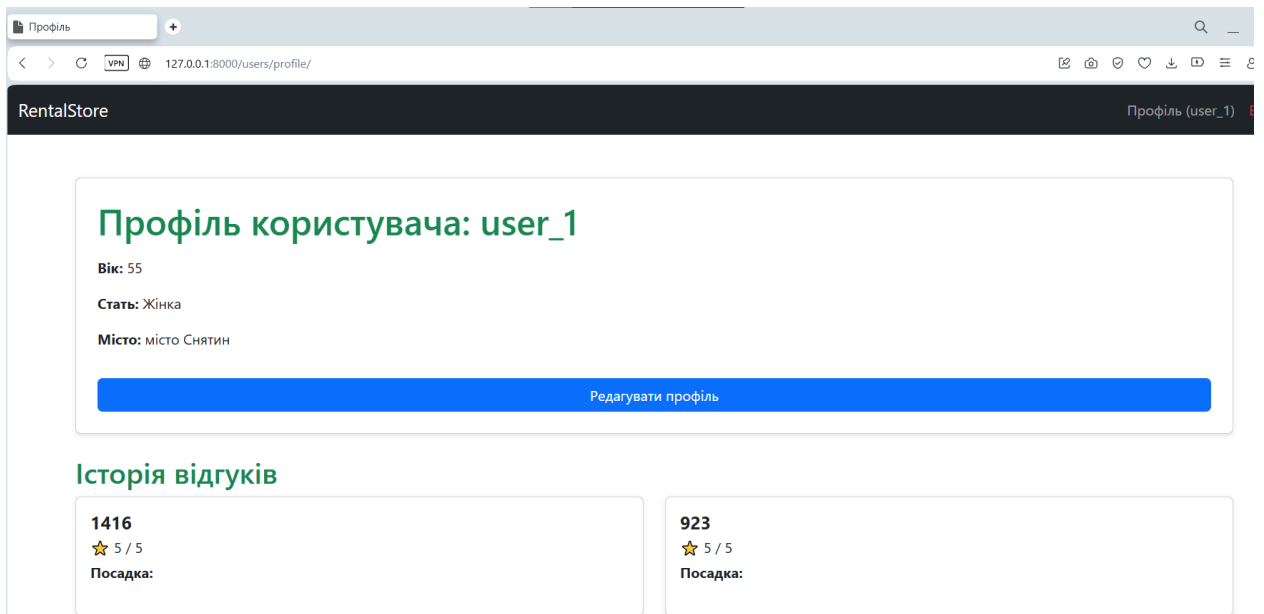


Рис. 3.14. Профіль користувача

На даному рисунку 3.15 представлено блок з переліком товарів, які користувач орендував раніше. Для кожної позиції вказано бренд, коротку характеристику та точну дату оренди. Це забезпечує зручний доступ до власної історії взаємодії з сервісом.

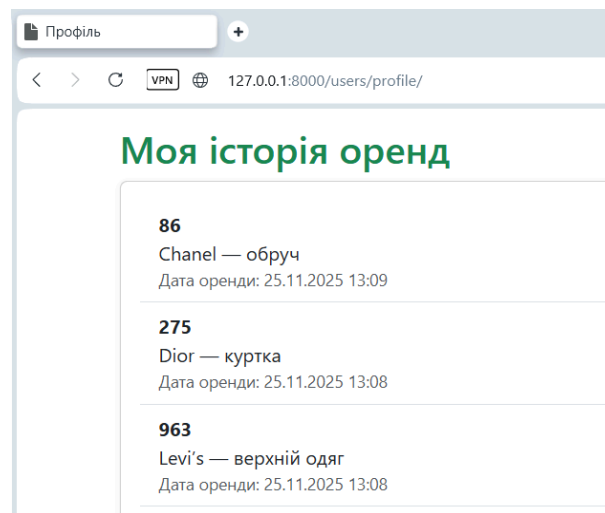


Рис. 3.15. Профіль користувача (історія оренд)

Цей розділ демонструє історію останніх переглядів користувача. Окрім назви та бренду товару, подається дата та час перегляду, що дозволяє швидко повернутися до моделей, які зацікавили користувача під час навігації платформою.

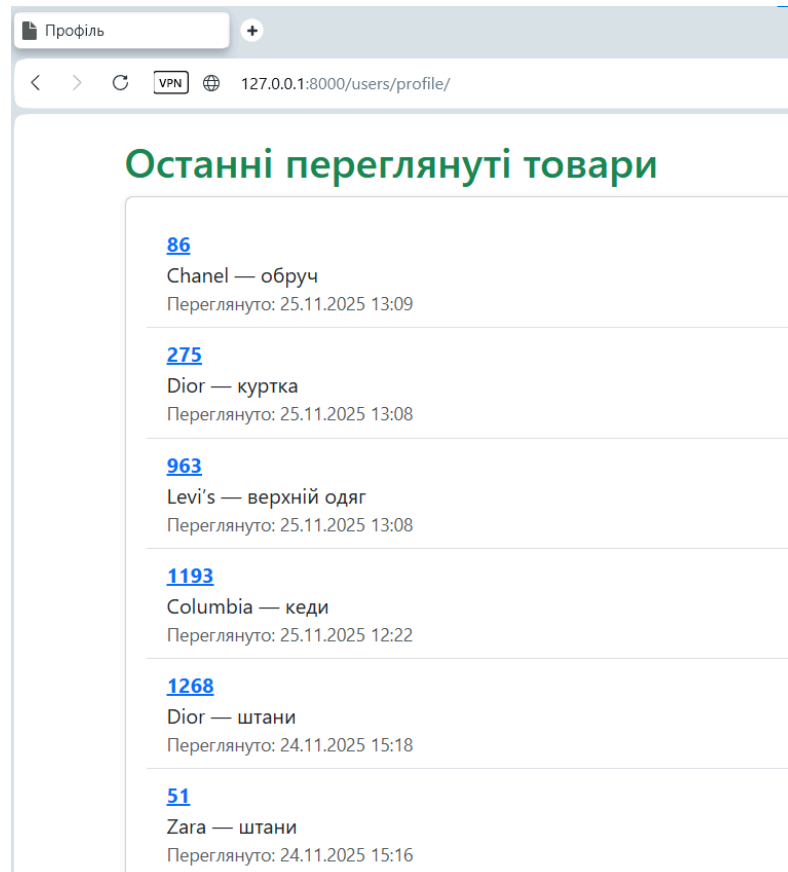


Рис. 3.16. Профіль користувача (переглянуті товари)

На рисунку демонструється сторінка детального перегляду товару. Інтерфейс містить повний опис вибраної моделі: назву, бренд, колір, сезонність, ціну та середній рейтинг. Нижче подано текстовий опис товару та розділ із відгуками інших користувачів, де зазначено оцінку, тип посадки та коментар.

Окремим блоком представлено відгук, залишений поточним користувачем (user_1), який має можливість не лише оновити власну оцінку, але й повністю видалити її — що є важливою особливістю системи. У нижній частині сторінки розміщено рекомендовані товари, сформовані на основі поведінки інших користувачів, які цікавилися подібними моделями.

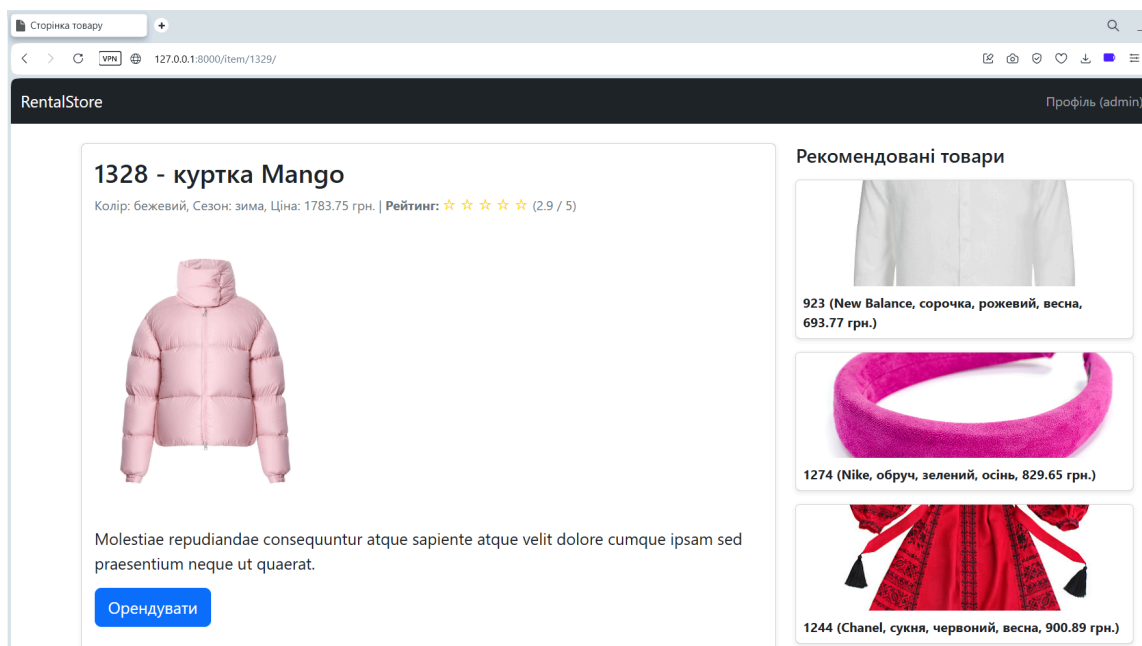


Рис. 3.17. Детально про товар

Головна сторінка адміністративної панелі Django (рис. 3.18) призначена для керування вмістом платформи та доступна лише користувачу з роллю *superuser*. Після авторизації адміністратор бачить привітання, навігаційні посилання (перехід на сайт, зміна пароля, вихід) та перемикач теми інтерфейсу.

Нижче розміщено структурований список доступних моделей додатку: *Item*, *Item view history*, *Rental history*, *User profiles*, Відгуки, а також системні моделі Django — *Users* та *Groups*. Для кожної моделі доступні дії *Add* та *Change*, що дозволяють додавати нові записи або переглядати й редагувати існуючі.

У нижній частині сторінки показано блок *Recent actions*, де відображаються нещодавні операції адміністратора, зокрема перегляди історії товарів або видалення записів. Цей журнал допомагає контролювати зміни, виконані в системі, та забезпечує прозорість роботи з даними.

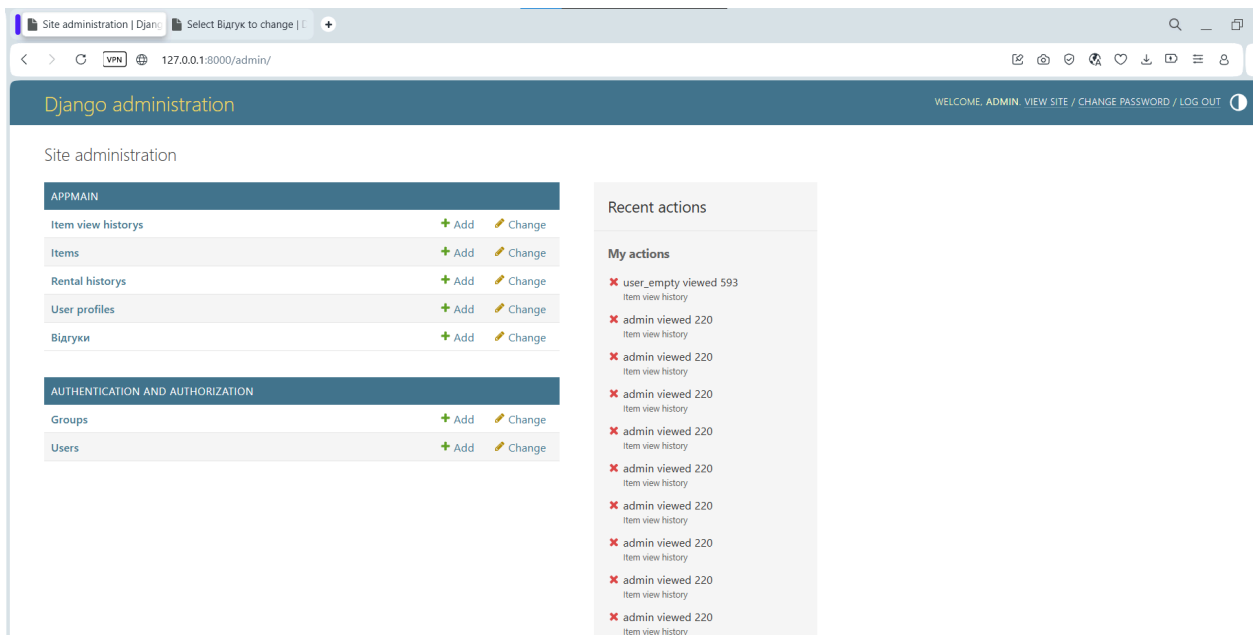


Рис. 3.18. Адміністративна панель (головна сторінка)

Сторінка управління відгуками (рис. 3.19) відкривається після переходу до моделі Відгуки в адмін-панелі. У верхній частині розташовані елементи навігації, що дозволяють повернутися на головну сторінку адмін-панелі, а також кнопка Add, яка відкриває форму створення нового відгуку.

Основну частину інтерфейсу становить таблиця зі списком усіх відгуків. Для кожного запису відображається користувач, пов'язаний товар, рейтинг, тип відповідності розміру (fit) та дата публікації. Адміністратор може відкрити будь-який запис для редагування.

У правій частині присутні фільтри за рейтингом та параметром fit, що дозволяє швидко звузити вибірку, знаходити проблемні відгуки або перевіряти якість взаємодій користувачів з товарами. Для великих наборів даних реалізовано пагінацію, що дає змогу зручно переглядати тисячі відгуків. Таким чином, інтерфейс забезпечує ефективне адміністрування відгуків і контроль над контентом платформи.

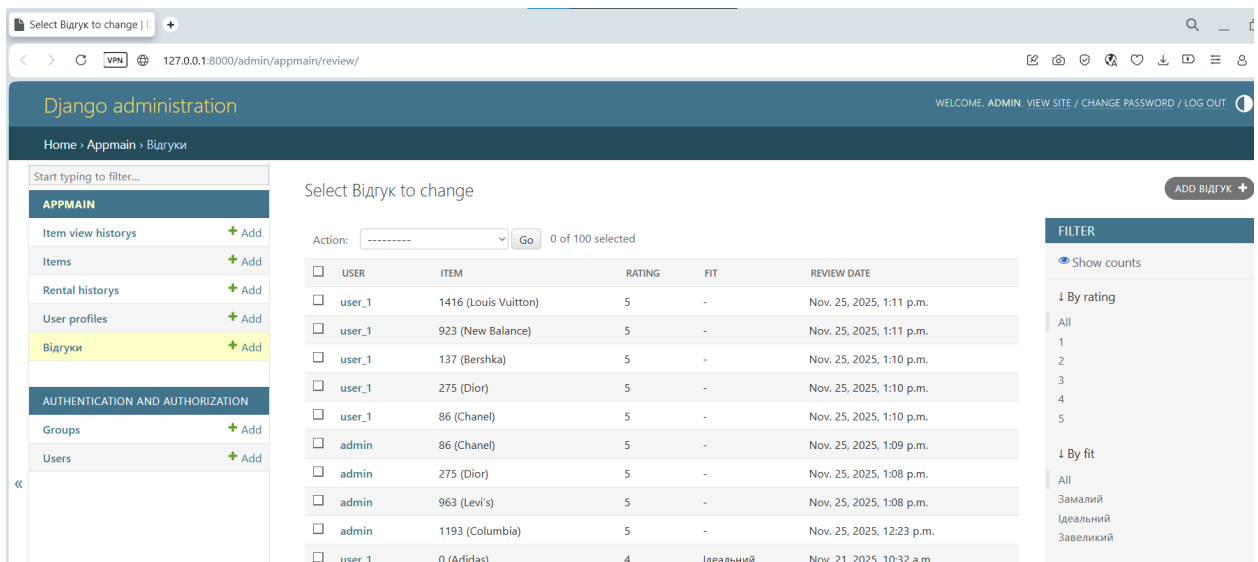


Рис. 3.19. Адміністративна панель (сторінка відгуків)

3.2. Відображення результатів рекомендацій

Після автентифікації користувач отримує доступ до персоналізованих рекомендацій, сформованих різними алгоритмами системи. Інтерфейс відображення результатів структуровано у вигляді окремих блоків, кожен з яких відповідає певному підходу до рекомендацій: гібридному методу, контентній фільтрації, колаборативній фільтрації на основі товарів (IBCF) та рекомендаціям на основі переглядів. Такий розподіл дозволяє візуально порівняти роботу алгоритмів і оцінити різні аспекти персоналізації.

У першому блоці представлено «Найкращі рекомендації для вас» – результати гібридного методу, який поєднує контентну схожість товарів, інформацію про перегляди й попередні оренди користувача. Для користувача user_1 система пропонує широкий набір різних категорій товарів, наприклад: кеди Columbia рожевого кольору за ціною 200,17 грн, білі чобітки Levi's, бежеву кепку Puma чи рожеву куртку Dior. Такий набір добре демонструє здатність гібридного методу узагальнювати переваги користувача, комбінуючи одночасно стиль, сезонність та історію активності. Рекомендації охоплюють різні типи товарів, що вказує на варіативність моделей та їхню здатність шукати релевантні об'єкти у великому каталозі.

Контентний блок демонструє роботу алгоритму, який враховує текстові та категоріальні характеристики товарів, зокрема бренд, колір, сезон та тип. У

цій секції відображаються результати порівняння описів товарів, що раніше зацікавили користувача, з іншими позиціями каталогу. Як приклад, користувачеві пропонуються штани Zara білого кольору, куртка Adidas чорного кольору та кардиган Armani. Система підбирає їх на основі подібності характеристик, тому тут переважають товари зі схожими кольорами, стилем або сезонністю, що підсилює контентну релевантність рекомендацій.

Метод колаборативної фільтрації на основі товарів (IBCF) представлений окремим невеликим блоком. Він формує рекомендації на основі поведінки інших користувачів, які орендували ті самі товари, що й поточний користувач. Для user_1 система видала рекомендацію у вигляді коричневої куртки Nike за 656,41 грн. Це типова поведінка IBCF-алгоритму: він знаходить товар, який найчастіше зустрічався у взаємодіях схожих користувачів, та пропонує його як потенційно релевантний.

Окремо у системі представлено блок рекомендацій на основі переглядів. Це найдинамічніший метод, який аналізує елементи, що користувач переглядав останнім часом. У цьому блоці користувачу пропонується набір товарів, серед яких куртка Adidas чорного кольору, штани Zara білого кольору та білі чобітки Louis Vuitton. Такі рекомендації демонструють врахування безпосередніх інтересів користувача, які він виявив під час останніх взаємодій з платформою.

У нижній частині інтерфейсу наведено результати оцінювання точності роботи системи за метриками precision@15 та recall@15 . Значення показують, що гібридний метод має найкращі показники ($\text{precision} \approx 0.0556$, $\text{recall} \approx 0.8345$), що підтверджує ефективність комбінування різних джерел інформації. Колаборативна фільтрація має вищий recall , тоді як контентний метод демонструє помірні значення обох метрик. Ці результати підтверджують роль гібридного підходу як найбільш збалансованого для персоналізації.

Таким чином, інтерфейс рекомендаційної системи забезпечує зрозуміле та зручне відображення результатів роботи різних алгоритмів, дозволяє користувачу оцінити запропоновані товари за їх характеристиками та демонструє інтеграцію складних моделей у функціональний веб-додаток.

Висновки до розділу 3

У третьому розділі було представлено структуру та функціональні можливості інтерфейсу розробленої рекомендаційної системи на платформі оренди одягу. Розділ демонструє, як користувач взаємодіє з веб-додатком — від автентифікації до отримання персоналізованих рекомендацій — та підкреслює цілісність і логічність створеного інтерфейсу.

Початкові сторінки входу й реєстрації забезпечують простий і швидкий доступ до системи, що є важливим для залучення нових користувачів. Мінімалістичний дизайн і чітке структурування полів дозволяють створити позитивний перший досвід взаємодії з платформою. Після успішного входу користувач потрапляє на головну сторінку, яка поєднує загальний каталог товарів і персоналізований контент, зокрема блоки «Наші новинки», категорії товарів та індивідуальні рекомендації.

У розділі детально показано роботу різних алгоритмів рекомендацій. Для досвідчених користувачів система відображає результати гібридного методу, контентної фільтрації, колаборативної фільтрації та рекомендацій на основі переглядів. Такий підхід забезпечує різнобічний аналіз уподобань та дозволяє формувати точніші пропозиції. Показано й роботу системи у випадку нових користувачів, коли рекомендації ґрунтуються переважно на переглядах та загальних характеристиках каталогу.

Таким чином, у третьому розділі сформовано комплексне бачення веб-системи, де інтерфейс і рекомендаційні алгоритми взаємодіють як єдиний функціональний продукт, спрямований на забезпечення персоналізованого досвіду для кожного користувача.

ВИСНОВКИ

У ході виконання роботи було проведено повний цикл дослідження, проєктування та створення рекомендаційної системи для платформи оренди одягу та аксесуарів, що охоплює теоретичні засади, алгоритмічну реалізацію та інтеграцію моделі у веб-інтерфейс. Узагальнення результатів трьох розділів дозволяє оцінити досягнутий рівень розв'язання поставленої проблеми та визначити перспективи подальшого розвитку системи.

У першому розділі були сформовані теоретичні основи, необхідні для створення якісної рекомендаційної системи. Проведений аналіз сучасних платформ та існуючих алгоритмів продемонстрував, що ринок fashion-оренди має свою специфіку, яка потребує високого рівня персоналізації, урахування фізичних параметрів користувача, стилістичних характеристик товарів і поведінкових сигнатур. Дослідження прикладів Rent the Runway та By Rotation показало різні моделі рекомендацій — алгоритмічно складну та соціально-орієнтовану, — що дозволило визначити вимоги до майбутньої системи та обґрунтувати вибір гібридного підходу як найбільш ефективного. Також у межах цього розділу було обґрунтовано використання Django як технологічної основи проєкту.

Другий розділ зосереджено на практичній реалізації рекомендаційних моделей. Побудовано матрицю взаємодій користувачів і товарів, застосовано TF-IDF-векторизацію текстових атрибутів, реалізовано алгоритми колаборативної, контентної та гібридної фільтрації. Значну увагу приділено оптимізації обчислень за допомогою розріджених матриць і ефективних методів обробки даних. Розроблений модуль оцінювання якості рекомендацій на основі time-based holdout та метрик precision@k і recall@k дав змогу провести об'єктивне порівняння алгоритмів і зробити висновок про переваги гібридного підходу. На цьому етапі сформовано повноцінне рекомендаційне ядро, здатне працювати на реальних даних.

У третьому розділі реалізовану модель інтегровано у веб-платформу з повноцінним інтерфейсом, що забезпечує повний цикл взаємодії користувача

з рекомендаційною системою. Описано роботу модулів автентифікації, головної сторінки з індивідуальними рекомендаціями, сторінок товарів і профілю, а також адміністративної панелі Django. Структура веб-інтерфейсу підтвердила, що рекомендаційна система не лише функціонує коректно, а й здатна динамічно адаптуватися до різних сценаріїв взаємодії. У результаті створено інтегрований продукт, де алгоритмічне ядро тісно пов'язане з користувацьким досвідом.

У сукупності три розділи демонструють завершений приклад розробки системи рекомендацій — від теоретичного обґрунтування до повністю працюючого веб-рішення. Встановлено, що використання гібридного підходу забезпечує найвищу якість персональних рекомендацій, а застосування Django дозволяє інтегрувати алгоритмічну частину в масштабовану та зручну для користувачів платформу. Таким чином, у роботі створено сучасну, адаптивну та функціональну систему рекомендацій, яка відповідає вимогам ринку оренди одягу й може бути розширена у подальших дослідженнях та практичних застосуваннях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Raza, Shaina, Mizanur Rahman, Safiullah Kamawal, Armin Toroghi, Ananya Raval, Farshad Navah, and Amirmohammad Kazemeini. A comprehensive review of recommender systems: Transitioning from theory to practice. Computer Science Review Vol. 59. 2026. 97 p.
<https://arxiv.org/pdf/2407.13699>
2. X. Su, T. M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. Advances in Artificial Intelligence. Vol. 2009. 2009. 19 p.
<https://onlinelibrary.wiley.com/doi/pdf/10.1155/2009/421425>
3. Kolisnyk O., Vakaliuk T. A. Integrating Behavioral Data for Tailored Outfit Recommendations VII Всеукраїнської науково-технічної конференції «Комп'ютерні технології: інновації, проблеми, рішення», (2-3 грудня, 2024 p.) <https://conf.ztu.edu.ua/wp-content/uploads/2025/01/47.pdf>
4. Kolisnyk O., Vakaliuk T. A comparative study of hybrid recommender systems in fashion and accessory rentals. Сучасні інформаційні системи та технології: матеріали VII Всеукр. наук.-практ. інтернет-конф. за тематикою «Сучасні комп'ютерні системи та мережі в управлінні» (29 листопада 2024 р., м. Херсон, м. Хмельницький) / за ред. А. А. Григорової. – Херсон: Книжкове видавництво ФОП Вишемирський В. С., 2024. С. 67-70.
5. Буяло А. Б., Пономаренко І. В. Особливості побудови маркетингової стратегії компанії на ринку оренди одягу, взуття та аксесуарів в Україні. Економіка і Суспільство. № 17. 2018. С. 334–340.
<https://chmnu.edu.ua/wp-content/uploads/2019/06/Ekonomika-i-suspilstvo-17-2018.pdf#page=334>
6. Lerche, L. Using implicit feedback for recommender systems: characteristics, applications, and challenges. Dissertation to obtain the degree of Doctor of Natural Sciences. Dortmund, 2016. 104 p.
<https://eldorado.tu-dortmund.de/server/api/core/bitstreams/7f57b3cb-a7c0-4956-a63d-beb447862eaf/content>

7. Wang, H., Chen, K., Zhu, W. and Song, Z. A process model on P2P lending. Financial Innovation. 2015. Vol. 1, no. 1, p.3.
<https://link.springer.com/content/pdf/10.1186/s40854-015-0002-9.pdf>
8. Oluwatosin, H.S.. Client-server model. IOSR Journal of Computer Engineering. 2014. Vol. 16, no. 1. pp.67-71.
https://www.researchgate.net/profile/Shakirat-Sulyman/publication/271295146_Client-Server_Model/links/5864e11308ae8fce490c1b01/Client-Server-Model.pdf
9. Forcier, J., Bissex, P., & Chun, W. J.. Python web development with Django. Addison-Wesley Professional. 2008.
https://books.google.com.ua/books?hl=en&lr=&id=M2D5nnYlmZoC&oi=fnd&pg=PT21&dq=Django+&ots=v_YIEodRLU&sig=tB8kRf_Gb5NyOXrTTtYkEX-XwVA&redir_esc=y#v=onepage&q=Django&f=false
10. Harrington L. Relational database design and implementation. Morgan Kaufmann. 2016. 597 p.
https://books.google.com.ua/books?hl=en&lr=&id=yQgfCgAAQBAJ&oi=fnd&pg=PP1&dq=Relational+database&ots=qQHvkXUG0s&sig=FygLUmfSwnJRosL1fZW9It1TuGY&redir_esc=y#v=onepage&q=Relational%20database&f=false
11. Padhy, Rabi Prasad, Manas Ranjan Patra, and Suresh Chandra Satapathy. RDBMS to NoSQL: reviewing some next-generation non-relational database's. International Journal of Advanced Engineering Science and Technologie. Vol. 11, no. 1. 2011. p 15-30.
https://d1wqtxts1xzle7.cloudfront.net/61537369/2011_RDBMS_to_NoSQL_Reviewing_Some_Next-Generation_Non-20191217-38072-pk5ltv.pdf?1738391697=&response-content-disposition=inline%3B+filename%3DRDBMS_to_NoSQL_Reviewing_Some_Next_Gener.pdf&Expires=1765125307&Signature=SmH9yEKC9WC30cxbSizxbpXZnZCQ5fGWgK5XnK0pSI0hYH QxgGNRaomrveWoJ8KzVGdrZ35Az1znNpu0VReS8x8aP9kTwKLEoybVl fYpTuHxA~z3XTuGlGD-aYm0fKxrUEwxJYjB~TbkJ5JblM7XqSElJ2myi

UwOzk3Y3s4I~pEDPwTT1cqgl6jnz0R6pusv2WHj2enDqbCvCa6eRD75b
wi~tHL76-gfY8X6NlBH6Sbf0F0lP9~lXC4VzLIXfSWtlfGx1TxUqsyHYC
~Yu4MC6RNdQvoIDfZJBpumoAceaJ-pWHaumQrbvs1u-uSGAb~FzrLSSy
k2xjnjKZLMP84wAg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

12. Jatana, Nishtha, Sahil Puri, Mehak Ahuja, Ishita Kathuria, and Dishant Gosain. A survey and comparison of relational and non-relational database. International Journal of Engineering Research & Technology. Vol. 1, no. 6. 2012. p 1-5.

https://d1wqtxts1xzle7.cloudfront.net/76957411/a-survey-and-comparison-of-relational-and-non-relational-database-libre.pdf?1640094061=&response-content-disposition=inline%3B+filename%3DA_Survey_and_Comparison_of_Relational_an.pdf&Expires=1765125122&Signature=ZCWDb77cKOUqhIEykE6z8fCfRtXyt2Dy19IwrYqujKHUgb1PWptM-2ujDo2NS1DKVxgU0-EB62I6drC8PrY9w-TI9cbb35BXQp8BG0PDuJaY~avSvhGyRMuCd3UvsGeBQ8H4lOlIL6Yt84fLsiwpOLB8iU~~KxNMiHVANeRzBDDxCDkXGh7876XCyxqewB32F0iSasRt8R5ZSIsz4Z8VWUY6GjuGTPhj9eKR~4yiUvIKCzh3KqtqAgwiJn4oXPsqygyic1cB3HVmiMRHc-ES2f5xByqQoVtNprEd9Syvk3k7-NesZ4GTdrwN6exNZyiAqtfz3kR6vboV-5asfdm-g__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

13. Kreibich, Jay. Using SQLite. " O'Reilly Media, Inc.", 2010. 409 p.

https://books.google.com.ua/books?hl=en&lr=&id=HFIM47wp0X0C&oi=fnd&pg=PR7&dq=sqlite&ots=Fj2zfSxpbT&sig=A0naWeJVZIzcHjCcdMmVieDXTCA&redir_esc=y#v=onepage&q=sqlite&f=false

14. Лобур М.В., Шварц М.Є, Стех Ю.В. Моделі і методи прогнозування рекомендацій для колаборативних рекомендаційних систем. Вісник Національного університету «Львівська політехніка» «Інформаційні системи та мережі». № 901. 2018. С. 68–75.
<https://science.lpnu.ua/sites/default/files/journal-paper/2019/feb/15581/181912maket-68-75.pdf>

15. Rahutomo F., Kitasuka T., Aritsugi M. Semantic cosine similarity. The 7th

- international student conference on advanced science and technology ICAST. South Korea: University of Seoul, 2012. p. 1.
https://www.researchgate.net/profile/Faisal-Rahutomo/publication/262525676_Semantic_Cosine_Similarity/links/0a85e537ee3b675c1e000000/Semantic-Cosine-Similarity.pdf
16. Cohen, I., Huang, Y., Chen, J., Benesty, J., Benesty, J., Chen, J., Cohen, I. Pearson correlation coefficient. Noise reduction in speech processing, 2009, p. 1-4.
https://www.researchgate.net/profile/Philip-Sedgwick/publication/275470782_Pearson's_correlation_coefficient/links/569e7e3e08ae2c638eb56c50/Pearsons-correlation-coefficient.pdf
 17. Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web, 2001, April, (pp. 285-295).
https://d1wqtxts1xzle7.cloudfront.net/106844907/p519-libre.pdf?1698016290=&response-content-disposition=inline%3B+filename%3DItem_based_collaborative_filtering_recom.pdf&Expires=1738861734&Signature=D7R8LTpIc0uE~friWVvahiLcwa3e4dLuoolRSEd6XolSbTdLRQySj72N5YmSH8W9~sCbePHRrb0X70tQWtfLWCEqu9hV44fYWcgM4npF9S7u~1UoGcL-ec5BakbxEtVlmVkn1bBn0k35~BNVzFyt6bVrcFp1z5lUx1yYkkBIhK1ys4UQeW~x4iVTUgwibOzSjIffGIrEJDi7CqEjXt~pyw2H1uWQyAwJ-Z6ZxhYZ9UiJn7U0aG7ABzEDICVlfMEGlsUythB5m0f8UqmvY-40cAwj6EGclUyJ~7mP0ZOyNPw66NBjznk2U5cHnaaec6mSqD3jZDeeJ-Eys5RULsJBA__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA
 18. Abdi, H. Singular value decomposition (SVD) and generalized singular value decomposition. Encyclopedia of measurement and statistics, 2007. 907(912), p. 44.
https://www.cimat.mx/~alram/met_num/clases/Abdi-SVD2007-pretty.pdf
 19. Takács, G., & Tikk, D. Alternating least squares for personalized ranking. In Proceedings of the sixth ACM conference on Recommender systems 2012,

- September, pp. 83-90.
<https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=dbe9d04bffb5c1df8eb721dab4f744ea81d9a4c1>
20. Thorat, P. B., Goudar, R. M., & Barve, S. Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, 110(4), 2015, pp. 31-36.
https://d1wqtxts1xzle7.cloudfront.net/59762468/10.1.1.695.642820190617-91457-z4s1rf-libre.pdf?1560756140=&response-content-disposition=inline%3B+filename%3DSurvey_on_Collaborative_Filtering_Conten.pdf&Expires=1738862098&Signature=Cwc-IEExqkrXSVyztb2ZWYGAsQ5Q8Nod6h5Xro8eUJOoXU~~x5BiMaqgTrV0U4L6311HIszVrsw5Wcp8oPLq4MgxfYPDmOKdWfYnJXNliVE9p15gZGVKhbndgWH4TNzTbgf7aMcMXA8oz0VwTf0LuAVoHiffkUqIPuEyMhp5ML1HMfn3B6-4KdoLg6YHh~R~5CFNtZ-OhhSd-d9oWIpx1mh-Naz1Mtw3G0tVum5YOPQHdZ83r0RUPLhS4ZF9L1wIwrM5FpW30bj~J2ayof0VAv3xBKmi9kOweuUKwinGizk3VDImNqWVNxQtIKW3RCnkzaDre5Hq~Su3-SVj6dZvQQ__&Key-Pair-Id=APKAJL OHF5GGSLRBV4ZA
 21. Ramos, J. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning* Vol. 242, No. 1, 2003, December, pp. 29-48.
<https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=b3bf6373ff41a115197cb5b30e57830c16130c2c>
 22. Church, K. W. Word2Vec. *Natural Language Engineering*, 23(1), 2017, pp. 155-162.
 23. Jawahar, G., Sagot, B., & Seddah, D. What does BERT learn about the structure of language?. In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*. 2019, July, pp 1-18.
<https://arxiv.org/pdf/2103.11943>
 24. Deldjoo, Yashar, Fatemeh Nazary, Arnau Ramisa, Julian McAuley, Giovanni Pellegrini, Alejandro Bellogin, and Tommaso Di Noia. A review of modern

- fashion recommender systems. ACM Computing Surveys. Vol. 56, no. 4. 2023, p 1-37. <https://arxiv.org/pdf/2202.02757>
25. Lika, Blerina, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. "Facing the cold start problem in recommender systems." Expert systems with applications. Vol. 41, no. 4. 2014. p 2065-2073.
https://git.dml.ir/mr.rezaei/haji_thesis/raw/commit/441a4583c37d00dea91bd8cabe98246b3bbc2440/References/6-%20Facing%20the%20cold%20start%20problem%20in%20recommender%20systems.pdf
 26. Wang, Hulong, Zesheng Shen, Shuzhen Jiang, Guang Sun, and Ren-Jie Zhang. User-based collaborative filtering algorithm design and implementation. In Journal of Physics: Conference Series, vol. 1757, no. 1. 2021. 7 p.
<https://iopscience.iop.org/article/10.1088/1742-6596/1757/1/012168/pdf>
 27. Xue, Feng, Xiangnan He, Xiang Wang, Jiandong Xu, Kai Liu, and Richang Hong. Deep item-based collaborative filtering for top-n recommendation. ACM Transactions on Information Systems (TOIS). Vol. 37, no. 3. 2019. 25 p. <https://arxiv.org/pdf/1811.04392>
 28. Geetha, G., Safa, M., Fancy, C. and Saranya, D. A hybrid approach using collaborative filtering and content based filtering for recommender system. Journal of physics: conference series. Vol. 1000. 2018, April. 8 p.
<https://iopscience.iop.org/article/10.1088/1742-6596/1000/1/012101/pdf>
 29. Harris, C.R., Millman, K.J., Van Der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J. and Kern, R.. Array programming with NumPy. Vol. 585.. 2020. pp.357-362.
<https://www.nature.com/articles/s41586-020-2649-2.pdf>
 30. McKinney, W. Pandas: a foundational Python library for data analysis and statistics. Python for high performance and scientific computing. Vol. 14, no. 9. 2011. 9 p
https://d1wqtxts1xzle7.cloudfront.net/117768488/pyhpc2011_submission_9-libre.pdf?1724818056=&response-content-disposition=inline%3B+filename

%3DAnaplastic_Lymphoma_Kinase_ALK_positive.pdf&Expires=1765126731&Signature=ghWrc4c2gQCMo~TTfcXDzm3-2OTnlBdiXrLqS5gihQv~rkXZXs~QgvYekCm-Dly8sj2OtFwc4~WnG~X9cWNlA~QzCL-U~BQYxMQhWgtzdiO3tm5zpoxIdpOLV-AGkNvvNErKKlsw5hO2Ads80ycf8NfA3VX3DPwzKIpYXJOEA4DOgqfRftONNr3JTr~o3aDOFjb89PimGSOurp~1TtES57DT0YGcuLdJZyvu5zDoEOpriUG6GqZsCJ5BX2qL2DiNQUOayNpRSHxqajuceB1pnMkXxkbkmS-vLOVZv6y98-Ws1DyhG9kf3MGLKYbiBh2ouaDJGcq5eCouudIbjO1yA__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA.

31. Rayhan, Abu, and Robert Kinzler. Advancing scientific computing with Python's SciPy library. 2023. 19 p.

https://d1wqtxts1xzle7.cloudfront.net/105130832/Advancing_Scientific_Computing_with_Python_s_SciPy_Library-libre.pdf?1692461217=&response-content-disposition=inline%3B+filename%3DAdvancing_Scientific_Computing_with_Pyth.pdf&Expires=1765126867&Signature=XACKJqf8YA-s0kzPBVB2LK7~xkUDFOREum-iMC8qryViKFCiG00zohg-QmkbtoPbwCpUudP5vnSOggvceLI~qMY6Jl2oZA9MeNiCTpio94ElF6SE8pumQxiu4VXIIXqw6JVkz9msSbSc9EUVKQaqIHmHh~ml5PtuUwVXluataVO54qc0SxgxXuRiz~lq~ZF-PXZYlSuoAaNvfZYM RBHRta458ZbDzlOp7d8rGs~n5f9ZN2phKtdyQlKnfl~WP6J5-BXKQhNJ6Xjc5wI4wEkkIpFizuQbhqvOnLD38qmyMKxbAxt9IXM2NUvA6yZ3k4yjQ7CorDDGm~lJMBfrUstvHg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

32. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J... Scikit-learn: Machine learning in Python. the Journal of machine Learning research, Vol. 12. 2011. pp.2825-2830.
https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf?source=post_page

ДОДАТКИ

Реалізація персоналізованих рекомендаційних алгоритмів

generate_recs.py

```

def build_interaction_df():
    """Завантажуємо review → DataFrame (user_id, item_id, rating,
    review_date)."""
    qs = Review.objects.select_related("user", "item").all().values(
        "user_id", "item_id", "rating", "review_date"
    )
    df = pd.DataFrame(list(qs))
    if df.empty:
        return df
    # Переконаємось у типах
    df["user_id"] = df["user_id"].astype(int)
    df["item_id"] = df["item_id"].astype(int)
    df["rating"] = df["rating"].astype(float)
    df["review_date"] = pd.to_datetime(df["review_date"])
    return df

def build_item_df():
    """Завантажуємо item → DataFrame з полями, які використовуємо в
    content-based."""
    qs = Item.objects.all().values("id", "name", "category", "brand",
    "description", "color", "season", "size",
    "target_group")

    df = pd.DataFrame(list(qs))
    if df.empty:
        return df
    df = df.rename(columns={"id": "item_id"})
    # Заповнюємо NaN рядками
    text_cols = ["name", "category", "brand", "description", "color", "season"]
    for c in text_cols:
        if c in df.columns:
            df[c] = df[c].fillna("").astype(str)
    # Створимо поле 'text' для TF-IDF
    df["text"] = df["brand"] + " " + df["category"] + " " + df["color"] + " " +
    df["season"] + " " \
        + df["description"] + " " + df["size"] + " " +
    df["target_group"]
    return df

def build_user_item_matrix(interactions_df, min_rating=1.0):
    """
    Повертає:
    - sparse матрицю (items в стовпцях, users в рядках) або навпаки,
    - mapping: user_id → row idx, item_id → col idx
    Ми робимо users x items (rows=user, cols=item) – зручніше для user-based,
    але тут нам потрібно item-based => беремо транспозицію пізніше.
    """
    users = interactions_df["user_id"].unique()
    items = interactions_df["item_id"].unique()
    user_map = {u: i for i, u in enumerate(users)}
    item_map = {it: j for j, it in enumerate(items)}

    rows = interactions_df["user_id"].map(user_map)
    cols = interactions_df["item_id"].map(item_map)
    data = interactions_df["rating"]

```

```

mat = csr_matrix((data, (rows, cols)), shape=(len(users), len(items)))
return mat, user_map, item_map

def item_item_similarity_from_interactions(interactions_df):
    """Побудова item-item cosine similarity з матриці взаємодій."""
    if interactions_df.empty:
        return None, {}, {}
    mat, user_map, item_map = build_user_item_matrix(interactions_df)
    # Для item-based нам потрібна матриця items x users
    item_user = mat.T.tocsr()
    # cosine similarity між товарами (рядки item_user)
    sim = cosine_similarity(item_user, dense_output=False)
    # сим буде квадратною матрицею (n_items x n_items)
    # Повернемо sim як sparse/ndarray; також mapping item_id->col_index та
    зворотній
    item_id_by_idx = {v: k for k, v in item_map.items()}
    return sim, item_map, item_id_by_idx

def build_user_user_similarity(interactions_df):
    """User-based CF: cosine similarity між користувачами."""
    if interactions_df.empty:
        return None, {}, {}
    mat, user_map, item_map = build_user_item_matrix(interactions_df)
    # cosine similarity між користувачами
    sim = cosine_similarity(mat, dense_output=False)
    user_id_by_idx = {v: k for k, v in user_map.items()}
    return sim, user_map, user_id_by_idx

def recommend_user_based_for_user(user_id, interactions_df, top_k=10):
    """
    User-based CF: рекомендації на основі схожих користувачів.
    Повертає список item_ids.
    """
    if interactions_df.empty:
        return []

    mat, user_map, item_map_ub = build_user_item_matrix(interactions_df)
    if user_id not in user_map:
        return []

    u_idx = user_map[user_id]
    user_vecs = mat.toarray() # users x items
    target_vec = user_vecs[u_idx]

    # косинусна подібність з іншими користувачами
    norms = np.linalg.norm(user_vecs, axis=1) * np.linalg.norm(target_vec)
    norms[norms == 0] = 1e-9
    sims = (user_vecs @ target_vec) / norms # similarity vector

    # агрегуємо оцінки від схожих користувачів
    agg_scores = sims @ user_vecs # shape=(num_items,)

    # map до item_ids
    inv_item_map = {v: k for k, v in item_map_ub.items()}
    scores_dict = {inv_item_map[i]: agg_scores[i] for i in
    range(len(agg_scores))}

    # приховуємо вже оцінені
    rated = interactions_df[interactions_df["user_id"] ==
    user_id]["item_id"].unique()
    for r in rated:

```

```

        if r in scores_dict:
            scores_dict[r] = -np.inf

    # топ-K
    top_items = sorted(scores_dict, key=lambda x: -scores_dict[x])[:top_k]
    return top_items

def build_content_matrix(items_df):
    """TF-IDF матриця по полю text в items_df. Повертає матрицю та map
    item_id->idx"""
    if items_df.empty:
        return None, {}
    vectorizer = TfidfVectorizer(max_features=5000, ngram_range=(1, 2))
    tfidf = vectorizer.fit_transform(items_df["text"].values)
    item_map = {item_id: idx for idx, item_id in
    enumerate(items_df["item_id"].values)}
    return tfidf, item_map

def recommend_item_based_for_user(user_id, interactions_df, sim_matrix,
item_map, item_id_by_idx, top_k=10):
    """
    Для заданого user_id повертає top_k рекомендацій на основі item-item
    similarity.
    Логіка: беремо всі товари, які юзер оцінив, беремо їх індекси,
    підсумовуємо схожість до інших товарів, відкидаємо вже оцінені.
    """
    if interactions_df.empty or sim_matrix is None:
        return []

    # Які товари оцінив користувач
    user_reviews = interactions_df[interactions_df["user_id"] == user_id]
    if user_reviews.empty:
        return []

    # Map item_id -> idx
    rated_item_ids = user_reviews["item_id"].unique()
    # Переклад у індекси
    rated_idxes = []
    for it in rated_item_ids:
        if it in item_map:
            rated_idxes.append(item_map[it])

    if not rated_idxes:
        return []

    # Створимо score vector довжиною num_items
    # sim_matrix can be sparse; обчислюємо сумарний score = sum(sim[item_idx,
:] * rating)
    num_items = sim_matrix.shape[0]
    scores = np.zeros(num_items, dtype=float)

    for _, row in user_reviews.iterrows():
        it = row["item_id"]
        rating = row["rating"]
        if it not in item_map:
            continue
        idx = item_map[it]
        # similarity row до всіх інших товарів
        sim_row = sim_matrix.getrow(idx).toarray().ravel() if
hasattr(sim_matrix, "getrow") else sim_matrix[idx]
        # акумулюємо зважено
        scores += sim_row * rating

```

```

# Приховуємо вже оцінені
for it in rated_item_ids:
    if it in item_map:
        scores[item_map[it]] = -np.inf

# Беремо топ-К по індексах
top_indices = np.argpartition(-scores, range(min(top_k,
len(scores))))[:top_k]
top_indices = top_indices[np.argsort(-scores[top_indices])]
result_item_ids = [item_id_by_idx[idx] for idx in top_indices if not
np.isinf(scores[idx])]
return result_item_ids[:top_k]

def recommend_content_based_for_user(user_id, interactions_df, items_df,
tfidf_matrix, tfidf_item_map, top_k=10):
    """
    Рекомендуємо на основі схожості контенту:
    - беремо товари, які користувач оцінив > 0 (наприклад rating >=4),
    - усереднюємо їх TF-IDF вектори → профіль користувача,
    - знаходимо косинусну подібність між профілем та всіма товарами,
    - приховуємо вже оцінені товари.
    """
    if interactions_df.empty or tfidf_matrix is None:
        return []

    user_reviews = interactions_df[interactions_df["user_id"] == user_id]
    if user_reviews.empty:
        return []

    # Виберемо "позитивні" товари: rating >= 4
    pos = user_reviews[user_reviews["rating"] >= 4]["item_id"].unique()
    if len(pos) == 0:
        # якщо немає позитивних – візьмемо найвищі рейтинги
        pos = user_reviews.sort_values("rating",
ascending=False)["item_id"].unique()[:3]

    pos_idx = [tfidf_item_map[it] for it in pos if it in tfidf_item_map]
    if not pos_idx:
        return []

    # Профіль користувача = середній вектор TF-IDF
    user_vec = tfidf_matrix[pos_idx].mean(axis=0)
    # Косинус подібність профілю до всіх товарів
    user_vec = np.asarray(user_vec)
    tfidf_matrix = np.asarray(tfidf_matrix.toarray())
    sims = cosine_similarity(user_vec, tfidf_matrix).ravel()
    # Приховуємо оцінені товари
    rated = user_reviews["item_id"].unique()
    for it in rated:
        if it in tfidf_item_map:
            sims[tfidf_item_map[it]] = -np.inf

    top_idx = np.argpartition(-sims, range(min(top_k, len(sims))))[:top_k]
    top_idx = top_idx[np.argsort(-sims[top_idx])]
    # Повертаємо item_ids
    inv_map = {v: k for k, v in tfidf_item_map.items()}
    result_item_ids = [inv_map[idx] for idx in top_idx if not
np.isinf(sims[idx])]
    return result_item_ids[:top_k]

def recommend_hybrid_for_user(user_id, interactions_df, sim_matrix, item_map,

```

```

item_id_by_idx,
                                items_df, tfidf_matrix, tfidf_item_map,
                                top_k=10, alpha=0.4):
    """
    Гібрид:  $\alpha \cdot \text{item-based} + \beta \cdot \text{user-based} + (1 - \alpha - \beta) \cdot \text{content}$ 
    Повертає top_k item_ids та список словників hybrid_recs.
    """
    master_items = items_df["item_id"].values
    master_idx_map = {it: idx for idx, it in enumerate(master_items)}
    n = len(master_items)

    item_scores = np.zeros(n, dtype=float)
    user_scores = np.zeros(n, dtype=float)
    content_scores = np.zeros(n, dtype=float)

    user_reviews = interactions_df[interactions_df["user_id"] == user_id]

    # --- Item-based частина ---
    if sim_matrix is not None:
        sim_size = sim_matrix.shape[0]
        sim_idx_for_master = {it: item_map[it] if it in item_map else None for
it in master_items}
        if not user_reviews.empty:
            scores = np.zeros(sim_size, dtype=float)
            for _, row in user_reviews.iterrows():
                it = row["item_id"]
                rating = row["rating"]
                if it in item_map:
                    idx = item_map[it]
                    sim_row = sim_matrix.getrow(idx).toarray().ravel() if
hasattr(sim_matrix, "getrow") else sim_matrix[
                    idx]
                    scores += sim_row * rating
            for master_item, master_idx in master_idx_map.items():
                sim_idx = sim_idx_for_master.get(master_item)
                if sim_idx is not None:
                    item_scores[master_idx] = scores[sim_idx]

    # --- User-based частина ---
    user_top_items = recommend_user_based_for_user(user_id, interactions_df,
top_k=len(master_items))
    # нормалізуємо оцінки для інтеграції
    for idx, item_id in enumerate(user_top_items):
        if item_id in master_idx_map:
            user_scores[master_idx_map[item_id]] = len(master_items) - idx #
ранг як оцінка

    # --- Content-based частина ---
    if tfidf_matrix is not None:
        user_pos = user_reviews[user_reviews["rating"] >= 4]["item_id"].unique()
        if len(user_pos) == 0:
            user_pos = user_reviews.sort_values("rating",
ascending=False)["item_id"].unique()[:3]

        pos_idxes = [tfidf_item_map[it] for it in user_pos if it in
tfidf_item_map]
        if pos_idxes:
            user_vec = tfidf_matrix[pos_idxes].mean(axis=0)
            user_vec = np.asarray(user_vec)
            tfidf_matrix_arr = np.asarray(tfidf_matrix.toarray())
            sims = cosine_similarity(user_vec, tfidf_matrix_arr).ravel()
            inv = {v: k for k, v in tfidf_item_map.items()}
            for tfidf_idx, s in enumerate(sims):
                item_id = inv[tfidf_idx]

```

```

        if item_id in master_idx_map:
            content_scores[master_idx_map[item_id]] = s

# --- Нормалізація ---
def norm(x):
    if np.all(x == 0):
        return x
    mn = np.nanmin(x[np.isfinite(x)])
    mx = np.nanmax(x[np.isfinite(x)])
    if mx == mn:
        return np.zeros_like(x)
    return (x - mn) / (mx - mn)

item_n = norm(item_scores)
user_n = norm(user_scores)
content_n = norm(content_scores)
alpha = 0.7
beta = 0.15

hybrid = alpha * item_n + beta * user_n + (1 - alpha - beta) * content_n

# --- Приховуємо вже оцінені ---
rated = user_reviews["item_id"].unique()
for r in rated:
    if r in master_idx_map:
        hybrid[master_idx_map[r]] = -np.inf

# --- Топ-К ---
top_idx = np.argpartition(-hybrid, range(min(top_k, len(hybrid))))[:top_k]
top_idx = top_idx[np.argsort(-hybrid[top_idx])]
result = [master_items[idx] for idx in top_idx if not np.isinf(hybrid[idx])]

# --- Словники для зручного відображення ---
hybrid_recs = []
for item_id in result:
    row = items_df[items_df["item_id"] == item_id]
    if not row.empty:
        rec = {
            "item_id": int(item_id),
            "name": row["name"].values[0] if "name" in row else f"Item
{item_id}",
            "brand": row["brand"].values[0] if "brand" in row else
"Unknown",
            "category": row["category"].values[0] if "category" in row else
"-",
            "score": float(hybrid[master_idx_map[item_id]])
        }
        hybrid_recs.append(rec)

return result[:top_k], hybrid_recs, user_top_items[:top_k]
def train_test_split_time_based(interactions_df):
    """
    Holdout: для кожного користувача беремо найпізніший review як тест, решту
    - train.
    Повертаємо train_df, test_df.
    """
    if interactions_df.empty:
        return interactions_df, interactions_df
    interactions_df = interactions_df.sort_values("review_date")
    # Найпізніший по кожному юзеру
    last = interactions_df.groupby("user_id")["review_date"].idxmax()
    test_idx = list(last.values)
    test_df = interactions_df.loc[test_idx].copy()
    train_df = interactions_df.drop(index=test_idx).copy()

```

```

    return train_df, test_df

def precision_recall_at_k(recommended, ground_truth, k):
    """recommended - list, ground_truth - set"""
    if len(recommended) == 0:
        return 0.0, 0.0
    recommended_k = recommended[:k]
    hits = len(set(recommended_k) & set(ground_truth))
    precision = hits / k
    recall = hits / len(ground_truth) if len(ground_truth) > 0 else 0.0
    return precision, recall

def evaluate_model(interactions_df, items_df, sim_matrix, item_map,
item_id_by_idx, tfidf_matrix, tfidf_item_map,
                    k=10, alpha=0.6, sample_users=200):
    """Оцінка на holdout: precision@k, recall@k серед sample_users."""
    if interactions_df.empty:
        return {}
    train_df, test_df = train_test_split_time_based(interactions_df)
    # Зважимо вибір користувачів, які мають >0 у тесті
    test_users = test_df["user_id"].unique()
    if len(test_users) == 0:
        return {}
    # можливий downsample для швидкості
    if sample_users and len(test_users) > sample_users:
        test_users = np.random.choice(test_users, sample_users, replace=False)

    precisions = []
    recalls = []
    for uid in test_users:
        gt_items = set(test_df[test_df["user_id"] == uid]["item_id"].values)
        # Генеруємо рекомендації з train_df
        # note: use train_df for interactions in recommend functions
        rec_item, _, _ = recommend_hybrid_for_user(
            uid, train_df, sim_matrix, item_map, item_id_by_idx,
            items_df, tfidf_matrix, tfidf_item_map,
            top_k=k, alpha=alpha
        )
        p, r = precision_recall_at_k(rec_item, gt_items, k)
        precisions.append(p)
        recalls.append(r)
    return {
        "precision_at_{}".format(k): float(np.mean(precisions)) if precisions
    else 0.0,
        "recall_at_{}".format(k): float(np.mean(recalls)) if recalls else 0.0,
        "evaluated_users": len(precisions)
    }
}

```

Реалізація методів рекомендацій для конкретного користувача, повертаючи найрелевантніші товари з деталями.

recommendation_service.py

```

from appmain.management.commands.generate_recs import (
    build_interaction_df,
    build_item_df,
    item_item_similarity_from_interactions,
    build_content_matrix,
    recommend_hybrid_for_user
)
from appmain.models import Item, ItemViewHistory
import pandas as pd

# ===== CONTENT-BASED =====
def get_content_recs(user_id, k=5):
    items_df = build_item_df()
    if items_df.empty:
        return []

    tfidf_matrix, tfidf_item_map = build_content_matrix(items_df)

    # якщо про користувача нічого не знаємо – нічого не повертаємо
    user_history =
ItemViewHistory.objects.filter(user_id=user_id).order_by("-viewed_at")[:3]
    if not user_history:
        return []

    liked_items_ids = [v.item.id for v in user_history]

    scores = {}
    from sklearn.metrics.pairwise import linear_kernel

    for liked in liked_items_ids:
        if liked not in tfidf_item_map:
            continue
        idx = tfidf_item_map[liked]
        sims = linear_kernel(tfidf_matrix[idx:idx+1], tfidf_matrix).flatten()
        for i, s in enumerate(sims):
            item_id = items_df.loc[i, "item_id"]
            scores[item_id] = scores.get(item_id, 0) + s

    scores = sorted(scores.items(), key=lambda x: x[1], reverse=True)
    results = [{"item_id": i, "score": s} for i, s in scores[:k]]

    # enrich with item metadata
    items = Item.objects.in_bulk([r["item_id"] for r in results])
    for r in results:
        it = items.get(r["item_id"])
        r.update(name=it.name, brand=it.brand, category=it.category)

    return results

# ===== ITEM-ITEM CF =====
def get_cf_recs(user_id, k=5):
    interactions = build_interaction_df()
    items_df = build_item_df()

```

```

if interactions.empty:
    return []

sim_matrix, item_map, item_id_by_idx =
item_item_similarity_from_interactions(interactions)

# Знаходимо, які товари користувач оцінював
user_data = interactions[interactions["user_id"] == user_id]
if user_data.empty:
    return []

interacted_ids = list(user_data["item_id"])
scores = {}

for iid in interacted_ids:
    if iid not in item_map:
        continue
    idx = item_map[iid]
    sims = sim_matrix[idx]

    for j, s in enumerate(sims):
        item_id = item_id_by_idx[j]
        scores[item_id] = scores.get(item_id, 0) + s

scores = sorted(scores.items(), key=lambda x: x[1], reverse=True)
results = [{"item_id": i, "score": s} for i, s in scores[:k]]

items = Item.objects.in_bulk([r["item_id"] for r in results])
for r in results:
    it = items.get(r["item_id"])
    r.update(name=it.name, brand=it.brand, category=it.category)

return results

# ===== VIEW-BASED =====
def get_view_recs(user_id, k=5):
    views = ItemViewHistory.objects.filter(user_id=user_id)[:k]
    results = []
    for v in views:
        results.append({
            "item_id": v.item.id,
            "name": v.item.name,
            "brand": v.item.brand,
            "category": v.item.category,
            "score": 1.0
        })
    return results

# ===== HYBRID =====
def get_hybrid_recs(user_id, k=5):
    interactions = build_interaction_df()
    items_df = build_item_df()

    if interactions.empty or items_df.empty:
        return []

    sim_matrix, item_map, item_id_by_idx =
item_item_similarity_from_interactions(interactions)
tfidf_matrix, tfidf_item_map = build_content_matrix(items_df)

ids, detailed, _ = recommend_hybrid_for_user(user_id, interactions,
sim_matrix, item_map, item_id_by_idx, items_df, tfidf_matrix, tfidf_item_map,

```

```
top_k=k,    )  
    return detailed
```