Національний університет «Полтавська політехніка імені Юрія Кондратюка»
(повне найменування вищого навчального закладу)
Навчально-науковий інститут інформаційних технологій та робототехніки
(повна назва факультету)
Кафедра комп'ютерних та інформаційних технологій і систем
(повна назва кафедри)

Пояснювальна записка
до дипломного проекту (роботи)
магістра
(освітньо-кваліфікаційний рівень)
на тему
Розробка системи рекомендацій для персоналізації контенту на веб-сайті із використанням методів машинного навчання

Виконав: студент 6 курсу, групи 602-ТН
спеціальності
122 Комп'ютерні науки та інформаційні технології
(шифр і назва напряму)
Хоменко А.О.
(прізвище та ініціали)
Керівник Фесенко Т.Г.
(прізвище та ініціали)
Рецензент Хоменко М.В.
(прізвище та ініціали)

Полтава – 2025 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»

НАВЧАЛЬНО НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ

КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

спеціальність 122 «Комп'ютерні науки»

на тему

«Розробка системи рекомендацій для персоналізації контенту на веб-сайті із використанням методів машинного навчання»

Студента групи 602-ТН Хоменка Антона Олексійовича

Керівник роботи
доктор технічних наук,
професор Фесенко Т.Г

Завідувач кафедри
кандидат фізико-математичних наук,
Двірна О.А.

Полтава – 2025

РЕФЕРАТ

Кваліфікаційна робота магістра: 89 с., 15 малюнків, 1 додатки, 15 джерел.

Об’єкт дослідження: процеси персоналізації контенту на веб-сайтах, алгоритми рекомендацій, їх вплив на користувацький досвід.

Мета роботи: створення ефективної системи рекомендацій, яка забезпечує персоналізацію контенту веб-сайту, використовуючи сучасні алгоритми машинного навчання.

Методи: аналіз даних користувачів, розробка рекомендаційних алгоритмів із застосуванням методів колаборативної та контентної фільтрації, інтеграція розробленої системи у веб-сайт.

Ключові слова: система рекомендацій, персоналізація, машинне навчання, колаборативна фільтрація, контентна фільтрація, алгоритми.

ABSTRACT

Master's Qualification Thesis: 89 pages, 15 figures, 1 appendix, 15 references.

Object of research: processes of content personalization on websites, recommendation algorithms, and their impact on user experience.

Purpose of the research: to develop an effective recommendation system that ensures content personalization on a website using modern machine learning algorithms.

Methods: user data analysis, development of recommendation algorithms utilizing collaborative and content-based filtering methods, and integration of the developed system into a website.

Keywords: recommendation system, personalization, machine learning, collaborative filtering, content-based filtering, algorithms.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – штучний інтелект

MVC – архітектурний шаблон Model-View-Controller.

ПЗ – програмне забезпечення.

ПП – програмний продукт.

JSON – формат обміну даними (JavaScript Object Notation).’

JS – JavaScript, мова програмування.

HTML – мова розмітки гіпертексту (HyperText Markup Language).

CSS – каскадні таблиці стилів (Cascading Style Sheets).

CFD – колаборативна система рекомендацій (Collaborative Filtering Dataset).

SQL – мова структурованих запитів (Structured Query Language).

БД – база даних.

CF – колаборативна фільтрація (Collaborative Filtering).

CBF – контентна фільтрація (Content-Based Filtering).

HTTP – протокол передачі гіпертексту (HyperText Transfer Protocol).

RS – система рекомендацій (Recommender System).

UX – досвід користувача (User Experience).

UI – користувацький інтерфейс (User Interface).

ВСТУП

У сучасному світі обсяг інформації, доступної користувачам через інтернет, зростає з надзвичайною швидкістю. В таких умовах постає необхідність забезпечити ефективну персоналізацію контенту, яка дозволяє користувачам отримувати саме ту інформацію, яка є для них найбільш релевантною. Системи рекомендацій є ключовим інструментом для досягнення цієї мети. Вони дозволяють адаптувати контент під індивідуальні потреби користувачів, забезпечуючи їм кращий досвід взаємодії з платформою та сприяючи зростанню її популярності.

Системи рекомендацій займають центральне місце у багатьох сферах сучасного життя. Вони стали незамінними в електронній комерції, допомагаючи користувачам знайти потрібні товари серед тисяч доступних варіантів. Відеостримінгові платформи, такі як Netflix чи YouTube, використовують рекомендаційні алгоритми, щоб запропонувати контент, який максимально відповідає вподобанням глядачів. У музичних сервісах, таких як Spotify, рекомендації грають ключову роль у формуванні плейлистів та відкритті нових треків.

Важливість систем рекомендацій також простежується у сфері освіти. Персоналізовані рекомендації допомагають студентам знаходити матеріали, що відповідають їхньому рівню знань та навчальним цілям, що підвищує ефективність навчального процесу. У новинних порталах рекомендації сприяють фільтрації великого обсягу інформації, дозволяючи кожному користувачеві отримувати новини, які відповідають його інтересам, і таким чином уникати інформаційного перевантаження.

Метою цієї роботи є розробка інноваційної системи рекомендацій для персоналізації контенту веб-сайту із застосуванням передових методів машинного навчання. Такі системи забезпечують можливість адаптувати контент до індивідуальних потреб користувачів, що дозволяє значно підвищити їх задоволеність і залученість.

Розвиток систем рекомендацій є критично важливим у сучасному цифровому світі. Вони здатні не лише вдосконалювати користувацький досвід, але й відігравати ключову роль у підвищенні ефективності бізнес-процесів. У електронній комерції, наприклад, рекомендаційні алгоритми допомагають пропонувати клієнтам персоналізовані товари та послуги, що безпосередньо впливає на зростання конверсій і доходів компанії. У стримінгових платформах, таких як Netflix чи Spotify, персоналізовані рекомендації сприяють утриманню клієнтів і підвищенню їхньої активності, пропонуючи їм саме той контент, який відповідає їхнім смакам.

Розробка системи рекомендацій у цій роботі базується на використанні сучасних підходів машинного навчання, таких як колаборативна та контентна фільтрація, а також їх гібридні моделі. Колаборативна фільтрація дозволяє створювати рекомендації на основі поведінкових даних інших користувачів, що мають схожі інтереси. Контентна фільтрація базується на аналізі характеристик контенту, з яким користувач взаємодіяв раніше. Комбінація цих підходів дає змогу значно підвищити точність рекомендацій та забезпечити користувачам більш релевантний контент.

Інтеграція рекомендаційного механізму у веб-сайт потребує ретельного проектування. У рамках цієї роботи створюється база даних для зберігання поведінкових даних користувачів, розробляються алгоритми для генерації рекомендацій, а також забезпечується інтеграція системи у веб-інтерфейс. Основними принципами, які враховуються при розробці, є масштабованість, продуктивність та простота використання системи як для кінцевих користувачів, так і для адміністраторів платформи.

Результати цієї роботи мають широкий спектр можливостей для застосування. У сфері електронної комерції рекомендаційні системи сприяють збільшенню продажів, пропонуючи клієнтам персоналізовані товари та послуги. Стримінгові сервіси, такі як музичні чи відеоплатформи, можуть використовувати рекомендації для покращення утримання користувачів,

пропонуючи їм цікаві матеріали на основі їхніх уподобань. Новинні портали можуть персоналізувати новинний контент, щоб кожен користувач отримував саме ті матеріали, які його найбільше цікавлять. Освітні платформи можуть використовувати подібні системи для пропозиції навчальних матеріалів, адаптованих до рівня знань та інтересів студента.

Таким чином, розробка системи рекомендацій для персоналізації контенту є важливим кроком для покращення якості обслуговування користувачів та підвищення ефективності бізнесу. Результати цієї роботи сприятимуть подальшому розвитку персоналізованих платформ у різних сферах та створенню нового рівня взаємодії між користувачами та цифровими сервісами.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД

1.1 Опис предметної області

Інтернет змінив наш спосіб здійснення покупок, завдяки величезному вибору продуктів, доступних для покупки онлайн. Однак ця зручність має свою ціну, оскільки споживачам доводиться перебирати незліченну кількість варіантів, що робить це непосильним і виснажливим завданням. З іншого боку, завдання для інтернет-магазинів полягає в тому, як продати більше товарів за вищою ціною та швидше, ніж їхні конкуренти. Одним із рішень є використання системи рекомендацій, яка використовує штучний ШІ для надання персоналізованих рекомендацій користувачам. Така система використовує алгоритми машинного навчання, які аналізують дані користувача, такі як історія пошуку, поведінка покупців і вподобання, щоб передбачити, які продукти, ймовірно, зацікавлять користувача.

Для споживачів переваги персоналізованих рекомендацій продукту очевидні. Вони економлять час і зусилля, пропонуючи індивідуальні пропозиції, які швидше за все відповідають інтересам і вподобанням споживачів. Вони також допомагають онлайн-продавцям збільшити дохід і прибуток, надаючи клієнтам персоналізовані рекомендації, заохочуючи їх купувати більше товарів. Крім того, вони допомагають розвинути лояльність і довіру клієнтів, покращуючи загальний досвід покупок для споживачів.

Персоналізовані рекомендації в реальному часі стали основною частиною нашого онлайн-досвіду. Системи рекомендацій у режимі реального часу зробили революцію в тому, як ми відкриваємо вміст і продукти та взаємодіємо з ними: від сервісів потокового передавання, які пропонують наш наступний серіал, який заслуговує на розпивання, до платформ електронної комерції, що пропонують індивідуальні рекомендації щодо продуктів. Системи рекомендацій є одним із ключових інструментів для персоналізації контенту в цифрових платформах. Їхнє

застосування охоплює різноманітні сфери, такі як електронна комерція, потокові сервіси, освітні платформи та новинні портали.

Персоналізовані системи рекомендацій щодо продуктів є цінними як для онлайн-продавців, так і для покупців. Незалежно від того, чи йдеться про електронну комерцію, потокове передавання медіа чи будь-який інший сектор, що пропонує користувачам контент, рекомендація нових матеріалів має вирішальне значення для успіху платформи. Навіть невелике збільшення відсотка доходу може перетворитися на мільйони доларів прибутку. Фактично, за оцінками McKinsey, рекомендації щодо продукту на основі алгоритмів спричиняють 35% споживчих покупок на Amazon і впливають на 75% контенту, який переглядають на Netflix [2].

Системи рекомендацій, пропонують переконливу альтернативу традиційним алгоритмам пошуку. Вони допомагають користувачам відкривати нові продукти, які вони могли пропустити, пропонують персоналізовані рекомендації на основі їхніх уподобань і, зрештою, роблять покупки ефективнішими та приємнішими. Ці системи можуть значно підвищити задоволеність клієнтів і стимулювати продажі за рахунок автоматизації процесу пошуку та економії часу клієнтів. Не дивно, чому сьогодні кожна велика платформа має систему рекомендацій.

Система рекомендацій на основі штучного інтелекту – це алгоритм машинного навчання, навчений ранжувати або оцінювати продукти чи користувачів. Він призначений для передбачення оцінок, які користувач може дати певному елементу, а потім повертає ці прогнози користувачеві в ранжованому списку. Ця технологія використовується багатьма популярними компаніями, такими як Google, Amazon і Netflix, щоб збільшити залучення користувачів до своїх платформ. Наприклад, Spotify може рекомендувати пісні, схожі на ті, які ви слухали раніше або сподобалися, щоб ви продовжували використовувати їхню платформу для прослуховування музики. Amazon може пропонувати користувачам продукти на основі даних, які вони зібрали про цього конкретного користувача.

Існуючі системи рекомендацій демонструють значний прогрес у різних галузях, сприяючи як підвищенню ефективності бізнесу, так і покращенню користувацького досвіду. Подальший розвиток цих систем спрямований на вдосконалення алгоритмів, підвищення точності рекомендацій та оптимізацію обчислювальних ресурсів.

Існує багато способів створити систему рекомендацій, і підходи можуть варіюватися від алгоритмічних і формульних до орієнтованих на моделювання. Ці підходи включають рейтинг сторінки, спільне фільтрування, на основі вмісту та прогнозування посилань. Однак важливо зазначити, що складність не обов'язково означає хорошу продуктивність, і часто прості рішення та реалізації дають найсильніші результати. Наприклад, такі великі компанії, як Reddit, Hacker News і Google, використовували прості шаблонні реалізації механізмів рекомендацій для просування вмісту на своїй платформі.

Визначення того, що визначає гарну рекомендацію, є проблемою, з якою досі борються багато компаній. Визначення «хороших» рекомендацій допомагає оцінити ефективність створеної системи рекомендацій. Якість рекомендації можна оцінити за допомогою різних тактик, які вимірюють охоплення та точність. Точність – це частка правильних рекомендацій із загальної кількості можливих рекомендацій, тоді як охоплення вимірює частку об'єктів у просторі пошуку, для яких система може надати рекомендації. Метод оцінки рекомендації залежить виключно від набору даних і підходу, використаного для створення рекомендації.

Системи рекомендацій мають кілька концептуальних подібностей із проблемою класифікації та регресійного моделювання. В ідеальній ситуації компанії хотіли б побачити, як реальні користувачі реагують на рекомендації, і відстежувати показники навколо користувача, щоб покращити свої рекомендації. На відміну від традиційних групових систем рекомендацій, які використовують тривалі робочі процеси вилучення, перетворення та завантаження над статичними наборами даних, системи рекомендацій у реальному часі динамічно

адаптуються до взаємодії користувача, надаючи рекомендації з низькою затримкою протягом сеансу користувача.

Системи рекомендацій у реальному часі часто використовують моделі машинного навчання або вдосконалені методи обробки даних, щоб передбачити переваги користувачів і надати персоналізовані пропозиції. Найдосконаліша система запису в режимі реального часу використовуватиме онлайн-сховище функцій або систему визначення моделі в режимі реального часу в поєднанні з довгостроковою системою навчання моделі для надання найкращих рекомендацій у режимі реального часу та постійного перенавчання та оптимізації моделі на основі користувачів зворотній зв'язок.

Загальною перевагою систем рекомендацій у реальному часі є їх здатність справлятися з динамічною та непередбачуваною поведінкою користувачів. Традиційним системам рекомендацій часто важко адаптуватися до раптових змін уподобань користувачів або тенденцій, оскільки вони покладаються на попередньо оброблені дані. Навпаки, системи рекомендацій у реальному часі, які використовують машинне навчання, можуть швидко коригувати свої рекомендації на основі нових взаємодій з користувачем, забезпечуючи своєчасність і ефективність пропозицій. Для розуміння сучасного стану систем рекомендацій важливо розглянути їхні типи, принципи роботи, переваги, недоліки та реальні приклади використання.

Технічно кажучи, системи рекомендацій використовують алгоритми машинного навчання, щоб надавати користувачам персоналізовані рекомендації щодо продуктів, послуг або інформації на основі їх поведінки, уподобань та історії. За даними Accenture, бренди, які визнають, пам’ятають і пропонують відповідні пропозиції, швидше за все, привернуть увагу 91% споживачів. Netflix є чудовим прикладом компанії, яка використала потужність механізмів рекомендацій для трансформації потокового передавання. Надаючи користувачам налаштований контент, Netflix досяг менших показників скасування, заощаджуючи компанії близько мільярда доларів на рік. Netflix є чудовим прикладом компанії, яка використала потужність механізмів

рекомендацій, щоб революціонізувати потокове передавання. Надаючи користувачам налаштований контент, Netflix досяг менших показників скасування, заощаджуючи компанії близько мільярда доларів на рік.

Система рекомендацій та персоналізація контенту відіграють важливу роль у сучасних веб-сайтах, особливо в умовах зростаючого обсягу інформації та різноманіття користувацьких вподобань. Вони дозволяють забезпечити користувачів релевантним контентом, підвищуючи ефективність взаємодії з платформою, що є критично важливим у конкурентних середовищах, таких як платформи для оренди нерухомості. В рамках цієї дипломної роботи предметною областю є створення рекомендаційної системи для сайту з оренди нерухомості, що використовує методи машинного навчання для персоналізації контенту.

Системи рекомендацій стали невід'ємною частиною багатьох веб-сайтів і платформ. Основна їхня мета полягає у допомозі користувачам швидше знаходити релевантний контент серед великої кількості варіантів. З розвитком інтернету і збільшенням кількості користувачів, пошук та вибір стають все складнішими, і це створює попит на автоматизовані рішення, що можуть передбачити інтереси користувачів.

1.2 Огляд і аналіз існуючих систем рекомендацій

Рекомендаційні системи відіграють ключову роль у сучасних цифрових платформах, зокрема на сайтах для оренди нерухомості, інтернет-магазинах, стримінгових сервісах та соціальних мережах. Вони допомагають користувачам знайти потрібний контент, товари або послуги, грунтуючись на їхніх попередніх взаємодіях, інтересах та поведінці інших користувачів.

Існує безліч програмних рішень, що використовують різноманітні методи машинного навчання для персоналізації контенту та поліпшення користувацького досвіду. У цьому розділі розглянемо найпоширеніші платформи та технології, які успішно використовуються для реалізації систем рекомендацій, зокрема у сфері нерухомості.

Airbnb є однією з провідних платформ, яка надає послуги оренди житла по всьому світу. Основною особливістю платформи є її рекомендаційна система, яка активно використовує методи машинного навчання для персоналізації пошукових результатів і рекомендацій житла для користувачів. Airbnb поєднує кілька типів алгоритмів, включаючи:

- колаборативну фільтрацію для аналізу поведінки користувачів і пропозиції варіантів на основі дій подібних користувачів;
- контентну фільтрацію, що враховує атрибути житла (наприклад, місце розташування, кількість кімнат, ціну, рейтинг) та інтереси користувача;
- гібридні методи, що поєднують колаборативну та контентну фільтрацію для надання більш точних і релевантних рекомендацій.

Airbnb також використовує аналіз природної мови (NLP) для оцінки відгуків та описів нерухомості, що дозволяє пропонувати варіанти, які більше відповідають конкретним вимогам користувачів. Наприклад, система може виділяти об'єкти з позитивними відгуками про чистоту, зручність або розташування.

Zillow — популярна платформа для купівлі, продажу та оренди житла у США. Вона також використовує систему рекомендацій, що допомагає користувачам швидше знайти нерухомість на основі їхніх уподобань. Основними технологіями, що використовуються на Zillow, є:

- моделі машинного навчання для прогнозування вартості об'єктів нерухомості та рекомендації варіантів на основі ринкових тенденцій;
- алгоритми колаборативної фільтрації, що аналізують поведінку користувачів, які здійснювали схожі пошуки, і рекомендують аналогічні об'єкти іншим користувачам;
- персоналізовані фільтри на основі попередніх пошукових запитів та уподобань користувача, що дозволяє швидше знайти варіанти нерухомості з

бажаними характеристиками (наприклад, житло в конкретному районі або з певною кількістю кімнат).

Zillow активно використовує великі дані (Big Data) для аналізу ринку нерухомості, прогнозування змін у вартості та надання точних рекомендацій користувачам на основі великої кількості історичних та актуальних даних.

— ще одна платформа, що надає послуги з пошуку нерухомості для купівлі та оренди. Її система рекомендацій також використовує індивідуальний підхід до користувача:

- контентна фільтрація дозволяє пропонувати об'єкти нерухомості на основі їхніх характеристик, таких як ціна, розташування, тип житла тощо;
- колаборативна фільтрація аналізує дії користувачів, які переглядали або взаємодіяли з певними оголошеннями, та пропонує схожі об'єкти новим користувачам;
- нейронні мережі застосовуються для прогнозування ймовірності того, що користувач зацікавиться певним об'єктом на основі його взаємодії з попередніми оголошеннями.

Однією з особливостей Realtor є інтеграція з картографічними сервісами (Google Maps), що дозволяє показувати об'єкти нерухомості на карті та враховувати близькість до ключових об'єктів інфраструктури, що також є важливим фактором для рекомендацій.

Trulia, як і Zillow, є потужною платформою для пошуку нерухомості. Trulia застосовує гібридні алгоритми рекомендацій, що використовують:

- колаборативну фільтрацію для вивчення поведінки користувачів, які здійснювали аналогічні пошукові запити;
- контентну фільтрацію для аналізу характеристик житла та рекомендації об'єктів, схожих на ті, які вже зацікавили користувача;

- геолокаційні алгоритми, що враховують місце розташування нерухомості та зручності, доступні у конкретному районі (школи, магазини, транспорт).

Trulia також використовує аналітику великих даних для надання рекомендацій не тільки стосовно житла, але й загальної інформації про райони, включаючи безпеку, рівень шуму, наявність шкіл та інші фактори, що можуть впливати на рішення користувача.

Окрім готових платформ, існують програмні рішення та бібліотеки, які можна використовувати для створення власних систем рекомендацій на веб-сайтах, включаючи сайти для оренди нерухомості.

є потужною платформою для масштабованого машинного навчання, що пропонує широкий спектр готових алгоритмів для побудови рекомендаційних систем, а також для виконання кластеризації та класифікації. Основною перевагою Mahout є його здатність обробляти великі обсяги даних, що робить платформу ідеальною для веб-проектів, які потребують персоналізованих рекомендацій на основі значних обсягів інформації. Mahout забезпечує високу продуктивність і гнучкість, що дозволяє легко інтегрувати його у проекти різного масштабу та складності.

— це сучасний фреймворк від компанії Google, який спеціально розроблений для побудови рекомендаційних систем на основі глибокого навчання. Завдяки цьому фреймворку розробники можуть створювати як моделі колаборативної, так і контентної фільтрації, що відкриває можливості для створення гібридних систем рекомендацій. TensorFlow Recommenders дозволяє ефективно працювати з великими наборами даних, забезпечуючи високу точність рекомендацій завдяки потужним інструментам глибокого навчання. Це робить його підходящим рішенням для проектів, де потрібні високоточні персоналізовані рекомендації.

— це Python-бібліотека, орієнтована на створення систем колаборативної фільтрації. Бібліотека надає вбудовані алгоритми для побудови моделей, що ґрунтуються на взаємодії користувачів із контентом, дозволяючи швидко генерувати точні рекомендації. Однією з головних переваг Surprise є її простота

у використанні, що дає змогу розробникам легко створювати і тестувати рекомендаційні системи без необхідності занурюватися у складні деталі реалізації алгоритмів. Ця бібліотека є чудовим вибором для проєктів, де швидкість впровадження та тестування систем рекомендацій має велике значення.

є потужним гібридним алгоритмом, який поєднує методи колаборативної та контентної фільтрації для побудови рекомендаційних систем. Завдяки цьому підходу, LightFM здатний враховувати як взаємодії користувачів із контентом, так і характеристики самих об'єктів, що дозволяє створювати більш точні та релевантні рекомендації. Цей алгоритм добре підходить для платформ з великою кількістю різнорідного контенту, зокрема таких як сайти для оренди нерухомості, де існує необхідність враховувати багато параметрів одночасно: місце розташування, вартість, тип житла, зручності тощо.

Таблиця 1.1 - Порівняльна таблиця існуючих програмних рішень

Характеристика				
Тип алгоритму	Колаборативна фільтрація, кластеризація, класифікація	Глибоке навчання, колаборативна та контентна фільтрація	Колаборативна фільтрація	Гібридний (колаборативна + контентна фільтрація)
Масштабованість	Висока, підходить для великих обсягів даних	Висока, завдяки TensorFlow і глибокому навчанню	Середня, підходить для менших проектів	Висока, підходить для великих проектів
Складність впровадження	Висока, вимагає глибоких знань і налаштувань	Відносно висока, потребує знань	Низька, проста у використанні	Середня, потребує розуміння гібридних моделей

Продовження таблиці 1.1

Основні алгоритми	Колаборативна фільтрація, кластеризація	Нейронні мережі, гібридні моделі	Колаборативна фільтрація	Гібридні моделі, колаборативна та контентна фільтрація
Використання великих даних	Підходить для великих обсягів даних	Підходить для обробки великих даних	Може працювати з меншими наборами даних	Підходить для великих та різнорідних наборів даних
Переваги	Масштабованість, підтримка кластеризації та класифікації	Висока точність, потужність глибокого навчання	Простота і швидкість реалізації	Гнучкість гібридного підходу
Недоліки	Складність налаштування і впровадження	Потребує великих обчислювальних ресурсів	Обмежена масштабованість	Може бути складним для налаштування
Сфера застосування	Великі проекти з великими обсягами даних	Проекти з високими вимогами до персоналізації та точності	Швидка реалізація на менших проектах	Платформи з різнорідним контентом, як-от сайти для оренди нерухомості

Огляд існуючих програмних рішень показує, що рекомендаційні системи є важливим інструментом для платформ, які надають послуги оренди нерухомості. Вони значно покращують користувацький досвід, надаючи індивідуальні рекомендації та допомагаючи користувачам швидше знаходити релевантні варіанти житла.

Вивчення існуючих платформ, таких як Airbnb, Zillow та Realtor, показує, що більшість із них використовують складні системи персоналізації контенту, базовані на машинному навчанні. Наприклад, Airbnb активно використовує колаборативну фільтрацію для рекомендації житла на основі вподобань схожих

користувачів, а також методи обробки природної мови (NLP) для аналізу відгуків та описів об'єктів нерухомості. Застосування алгоритмів машинного навчання дозволяє цим платформам підвищувати рівень залучення користувачів, забезпечувати високий рівень персоналізації і збільшувати коефіцієнт конверсії, тобто кількість користувачів, які знаходять і орендують нерухомість через платформу.

Розробка системи рекомендацій для сайту з оренди нерухомості із використанням методів машинного навчання є актуальним завданням, яке дозволяє підвищити ефективність платформи, забезпечити персоналізований підхід до кожного користувача та покращити загальний користувацький досвід. Використання різних методів, таких як контентна та колаборативна фільтрація, гібридні підходи та технології глибокого навчання, дозволяють досягти високої точності рекомендацій та зробити платформу зручною для широкого кола користувачів.

етоди машинного навчання в системах рекомендацій

Машинне навчання відіграє ключову роль у розробці рекомендаційних систем. Воно дозволяє будувати моделі, що можуть «вчитися» на великих наборах даних і робити прогнози про майбутні вподобання користувачів. Системи рекомендацій поділяються на кілька основних типів, кожен з яких використовує різні методи для аналізу даних і надання персоналізованих рекомендацій.

У цьому розділі розглянуто основні типи систем рекомендацій, їхні особливості, а також галузі, в яких кожен із методів може бути найбільш ефективним. Такий огляд дає змогу зрозуміти, який підхід до побудови рекомендаційної системи буде найбільш доцільним у конкретному контексті, наприклад, на платформі для оренди нерухомості, де важливо враховувати як індивідуальні вподобання користувачів, так і характеристики об’єктів нерухомості.

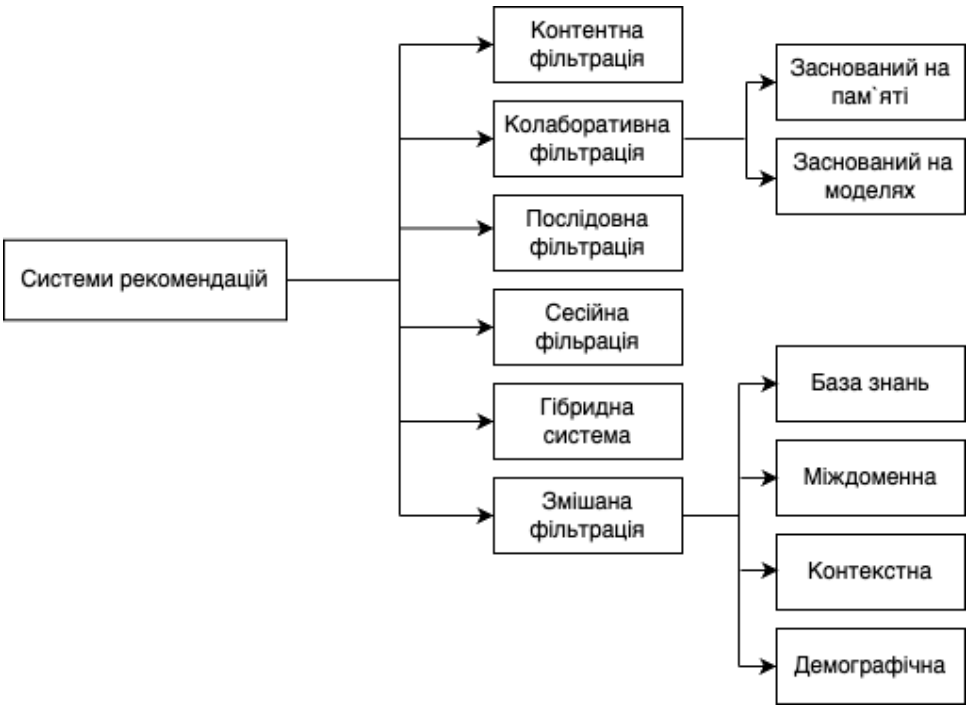

Рисунок 1.1 - Типи систем рекомендацій

Колаборативна фільтрація

Колаборативна фільтрація— це техніка, яка використовується в системах рекомендацій для прогнозування інтересів і вподобань користувача на основі даних і шаблонів багатьох користувачів. Основний принцип колаборативної фільтраціїполягає в тому, що якщо два користувачі мають схожі смаки в одному продукті, вони також, імовірно, матимуть подібні смаки в інших продуктах. Існує два основних типи підходів до спільної фільтрації: на основі пам’яті та на основі моделі.

Підходи на основі пам’яті, також відомі як фільтрація спільного сусідства, використовують рейтинги комбінацій користувачів і елементів, щоб передбачити їхні переваги на основі їхнього сусідства. Колаборативна фільтраціяна основі користувачів рекомендує продукти користувачеві на основі вподобань подібних користувачів, тоді як колаборативна фільтраціяна основі елементів рекомендує продукти на основі подібності між елементами, обчисленої з використанням оцінок користувачів цих товарів.

З іншого боку, підходи, засновані на моделях, використовують прогнозні моделі, які включають машинне навчання для параметризації функцій, пов'язаних із набором даних як вхідних даних моделі. Це допомагає вирішити проблему, пов'язану з оптимізацією. Підходи на основі моделі включають дерева рішень, підходи на основі правил і моделі латентних факторів [3].

Моделі колаборативної фільтраціїє перевагами, оскільки їх легко реалізувати, вони забезпечують охоплення високого рівня та фіксують тонкі характеристики, не вимагаючи знання вмісту елемента. Деякі приклади алгоритмів колаборативної фільтраціївключають рекомендації контенту YouTube на основі користувачів, які підписалися або переглянули подібні відео, і рекомендації курсів CourseEra на основі інших осіб, які закінчили існуючі курси, які закінчив користувач.

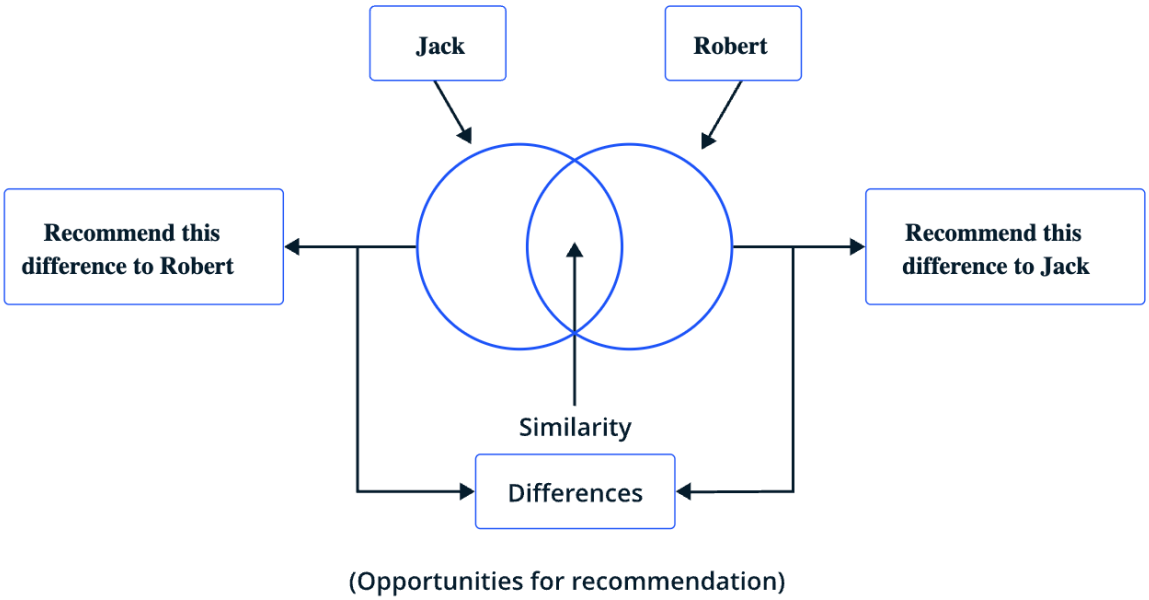

Рисунок 1.2 - Схема роботи колаборативної фільтрації

Контентна фільтрація

Система рекомендацій на основі вмісту — це система, яка генерує рекомендації на основі вподобань і профілю користувача, зіставляючи їх із елементами, які їм раніше сподобалися. Замість того, щоб використовувати оцінки між цільовим користувачем та іншими користувачами, моделі на основі

вмісту зосереджуються на оцінках, які надає цільовий користувач. Ці моделі встановлюють рівень подібності між елементами на основі атрибутів елементів, які сподобалися користувачеві. Щоб побудувати систему на основі вмісту, вам потрібне надійне джерело даних на рівні товару, пов’язаних з атрибутами товару, такими як ціна, рік публікації тощо, разом із деякими відгуками користувачів про товар, які можуть бути неявними. або явні. Моделі на основі вмісту є особливо вигідними, коли доступно недостатньо рейтингових даних, оскільки вони можуть використовувати рейтинги та атрибути предметів для створення рекомендацій [1].

Приклади систем, що базуються на вмісті, включають канал продуктів Amazon, який рекомендує продукти, подібні до тих, які користувач раніше придбав, і музичні рекомендації Spotify. Деякі компанії, такі як Hacker Rank і Reddit, також використовували алгоритмічні підходи, щоб рекомендувати нові публікації користувачам на основі таких факторів, як час публікації, кількість оцінок «подобається», «не подобається» та коментарів, які можна врахувати у формулі для генерування балів для публікації. а отже і рекомендація.

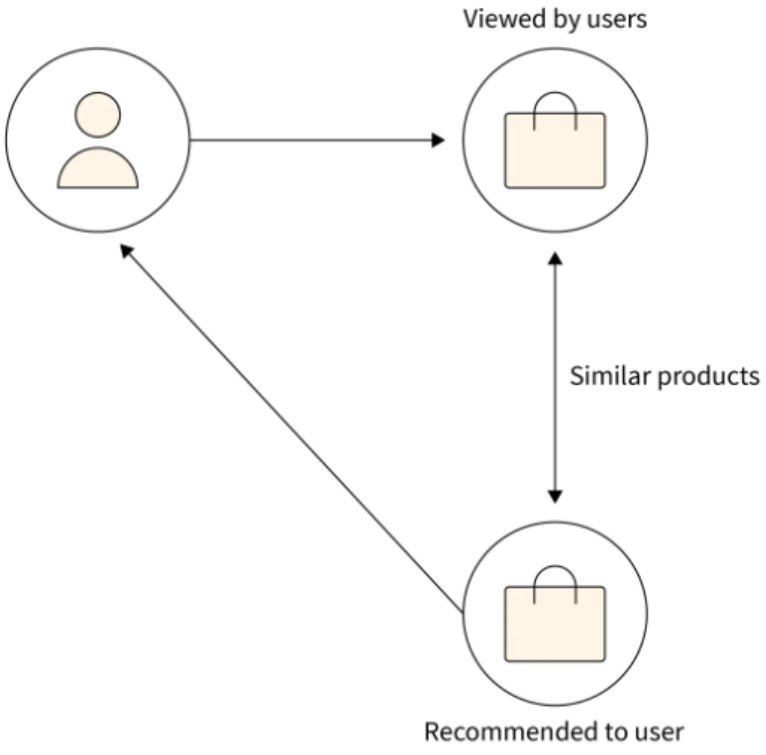

Рисунок 1.3 – Схема роботи контентної фільтрації

Гібридні методи

Гібридні системи рекомендацій створені для подолання обмежень індивідуальних систем рекомендацій шляхом об'єднання кількох джерел даних, що може бути досягнуто за допомогою двох різних дизайнів: паралельного та послідовного. У паралельному дизайні кілька систем рекомендацій використовуються паралельно для створення рекомендацій, а їхні результати об'єднуються для отримання кінцевого результату. Один механізм рекомендацій використовується в послідовному проектуванні, і його вихідні дані передаються наступному рекомендацію в послідовності [11].

Гібридні системи пропонують кілька переваг, зокрема підвищену надійність і персоналізацію в рекомендаціях користувача. Поєднуючи різні моделі, гібридні системи можуть пом'якшити слабкі сторони окремих моделей, що призведе до більш точних і різноманітних рекомендацій. Netflix є добре відомим прикладом компанії, яка використовує гібридну систему рекомендацій, поєднуючи спільну фільтрацію (на основі поведінки користувачів) із фільтрацією на основі вмісту (на основі характеристик елементів), щоб надавати своїм користувачам більш релевантні рекомендації щодо фільмів.

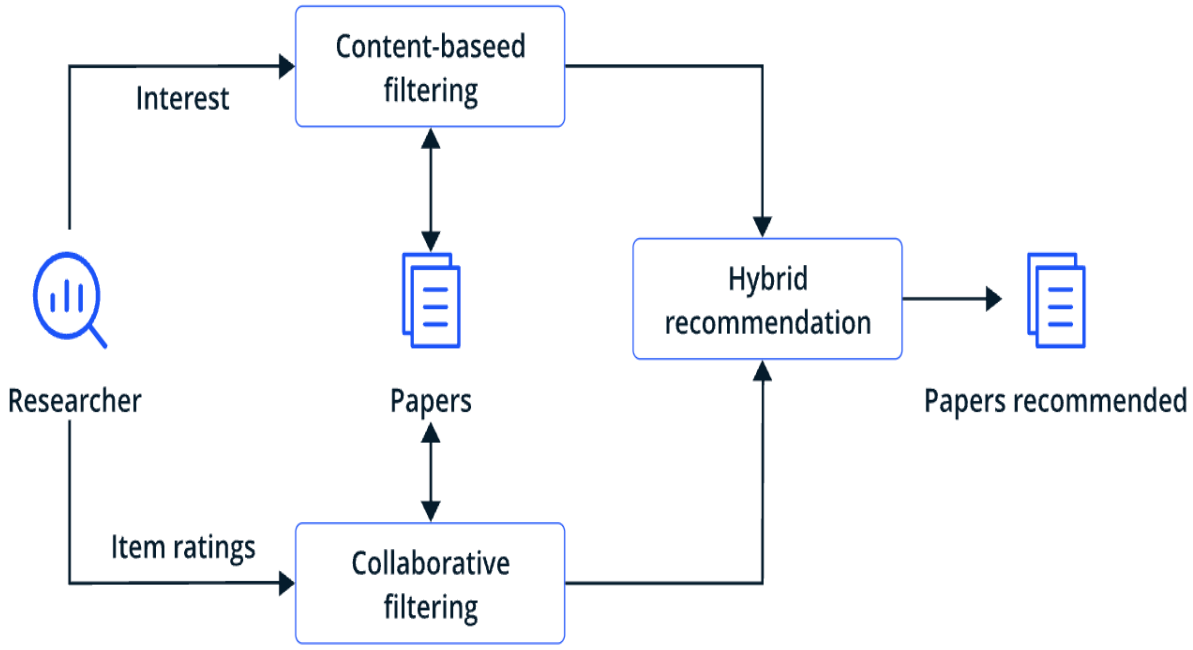

Рисунок 1.4 - Схема роботи гібридної фільтрації

Система бази-знань

Система, заснована на знаннях, у системі рекомендацій генерує рекомендації на основі потреб користувача та досвіду домену. Це передбачає визначення правил, які встановлюють контекст для кожної рекомендації, наприклад, критерії того, коли конкретний продукт або послуга принесе користь користувачеві. На відміну від підходу, заснованого на вмісті, ці правила не обов'язково покладаються на історію взаємодії користувача, але можуть містити іншу експертну інформацію або атрибути продуктів і послуг клієнта.

Однією з переваг системи, заснованої на знаннях, є те, що рекомендації можна легко пояснити, що полегшує користувачам зрозуміти, чому була зроблена певна рекомендація. Однак створення такого типу фреймворку може бути дорогим, і воно краще підходить для складних доменів, де елементи купуються рідко, а даних може бути браковано. Однією з переваг цього підходу є те, що він не страждає від тих самих проблем холодного запуску, як інші методи.

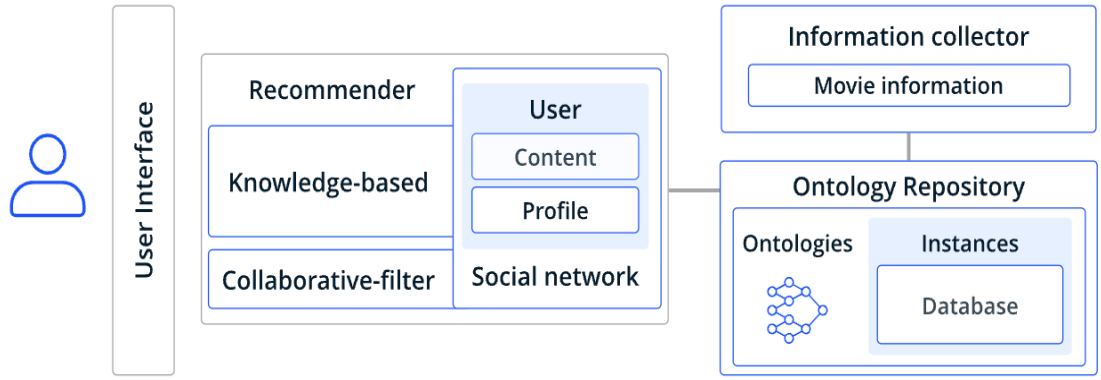

Рисунок 1.5 - Схема роботи системи бази знань

Система на основі демографічних даних

Рекомендації у таких системах ґрунтуються на демографічній інформації про користувачів, такій як вік, стать, місце проживання, соціальний статус тощо. Цей метод може бути корисним для сегментування користувачів і надання рекомендацій групам, які мають схожі демографічні характеристики. Наприклад, якщо користувачі певного віку або статі частіше вибирають конкретний тип

житла, система може використовувати ці дані для надання персоналізованих рекомендацій новим користувачам з аналогічними характеристиками.

Алгоритми на основі глибокого навчання

Останнім часом рекомендаційні системи все більше використовують методи глибокого навчання, зокрема нейронні мережі. Такі системи здатні враховувати складні багатофакторні взаємозв'язки між користувачами та об'єктами, що дозволяє забезпечити більш точні й персоналізовані рекомендації. Глибоке навчання може комбінувати дані про взаємодії, контент, демографічні дані і навіть дані зі сторонніх джерел, що робить цей підхід універсальним для складних систем, зокрема в нерухомості.

Системи на основі соціальних зв'язків

Ці системи використовують інформацію про соціальні зв'язки користувачів (наприклад, дані з соціальних мереж) для побудови рекомендацій. Ідея полягає в тому, що користувачі, які мають спільних друзів або підписки, можуть мати схожі інтереси і вподобання. Для платформи оренди нерухомості цей підхід може бути корисним для рекомендацій на основі соціальних уподобань чи дій друзів користувача.

Таблиця 1.2 - Порівняння систем рекомендацій

Тип системи рекомендацій	**Метод**	**Переваги**	**Недоліки**	**Сфера застосування**
Колаборативна фільтрація	Рекомендації на основі поведінки схожих користувачів або об'єктів	Висока точність на основі взаємодій користувачів Добре працює з великими даними	Проблема "холодного старту" Залежність від великих наборів даних	Веб-сайти з великою аудиторією, наприклад, стримінгові сервіси

Продовження таблиці 1.2

Контентна фільтрація	Рекомендації на основі характеристик контенту	Може рекомендувати новий контент Не потребує даних про інших користувачів	Обмеженість лише схожим контентом Менша гнучкість рекомендацій	Інтернет-магазини, медіа-платформи, новинні портали
Гібридна система	Поєднання кількох методів, зокрема колаборативної та контентної фільтрації	Поєднує переваги різних методів Вирішує проблему "холодного старту"	Складність впровадження Потребує більше обчислювальних ресурсів	Великі платформи (сайти нерухомості, онлайн-ринок)
Система на основі знань	Використовує заздалегідь визначені знання про об'єкти та вимоги користувачів	Може працювати без великих обсягів даних Відповідає конкретним вимогам	Менш адаптивна Потрібні детальні знання про об'єкти	Спеціалізовані сервіси (нерухомість, туристичні послуги)
Демографічна система	Рекомендації на основі демографічних даних	Простота сегментації користувачів Підходить для нових користувачів	Не враховує індивідуальні інтереси користувача	Реклама, новинні платформи
Система на основі глибокого навчання	Використовує нейронні мережі для виявлення складних взаємозв'язків	Висока точність Можливість комбінування кількох джерел даних	Високі вимоги до обчислювальних ресурсів Складність налаштування	Великі онлайн-платформи, соціальні мережі

Продовження таблиці 1.2

Соціальна система	Використовує дані соціальних мереж для рекомендацій	Використовує інформацію зі спільнот Підвищує соціальну релевантність	Обмежується лише соціальними зв'язками Менша персоналізація	Соціальні мережі, сервіси з інтеграцією соцмереж

Сайти для оренди нерухомості відрізняються від інших платформ через певні специфічні особливості. Такими особливостями можна вважити: динамічність ринку - оголошення швидко змінюються, нові об'єкти з'являються, а старі зникають, це створює додаткові вимоги до актуальності рекомендацій; Різноманіття критеріїв - користувачі можуть мати різні вподобання щодо нерухомості, від місця розташування до вартості оренди та додаткових зручностей.

Отже, існуючі системи рекомендацій демонструють значний прогрес у різних галузях, сприяючи як підвищенню ефективності бізнесу, так і покращенню користувацького досвіду. Подальший розвиток цих систем спрямований на вдосконалення алгоритмів, підвищення точності рекомендацій та оптимізацію обчислювальних ресурсів. У перспективі, гібридні підходи залишатимуться найбільш ефективним рішенням, забезпечуючи адаптивність і універсальність для різноманітних сфер застосування.

РОЗДІЛ 2
ПОСТАНОВКА ЗАДАЧІ

2.1 Визначення проблеми

У сучасному цифровому середовищі користувачі стикаються з величезним обсягом інформації, що ускладнює пошук релевантного контенту. Це створює значну проблему як для користувачів, так і для компаній, які прагнуть забезпечити якісний користувацький досвід і утримувати клієнтів. Надлишок інформації часто призводить до перевантаження, що унеможливлює швидкий доступ до потрібних матеріалів і знижує ефективність використання цифрових платформ.

Існуючі системи рекомендацій частково вирішують це завдання, використовуючи алгоритми персоналізації контенту. Однак вони стикаються з низкою обмежень. Наприклад, "проблема холодного старту" виникає у випадках, коли про нових користувачів або контент недостатньо даних для побудови рекомендацій. Додатково, розрідженість даних у великих базах створює труднощі для аналізу, а висока обчислювальна складність алгоритмів може уповільнювати роботу системи. Інші проблеми, такі як недостатня точність рекомендацій і обмеженість моделей до певних категорій даних, також знижують ефективність поточних рішень.

Ці виклики вимагають розробки нової, більш ефективної системи рекомендацій, яка могла б враховувати індивідуальні вподобання користувачів та адаптуватися до змінних умов. Така система має бути здатною працювати з великими обсягами даних, забезпечувати масштабованість і гарантувати високу продуктивність навіть за умов зростання навантаження. Успішна реалізація такого підходу дозволить не тільки покращити користувацький досвід, але й сприятиме підвищенню ефективності бізнесу, який залежить від утримання клієнтів та їхньої залученості.

2.2 Мета роботи

Метою роботи є розробка та впровадження системи рекомендацій для персоналізації контенту на веб-сайті із використанням сучасних методів машинного навчання. Основний акцент зроблено на побудові алгоритмів, які можуть працювати з різними типами даних (взаємодія користувачів, характеристики контенту) і забезпечувати високу точність рекомендацій.

2.3 Завдання роботи

Для досягнення поставленої мети необхідно виконати наступні завдання:
наліз предметної області та існуючих рішень. Провести огляд сучасних систем рекомендацій, визначити їх переваги та недоліки.
изначення вимог до системи. Розробити функціональні та нефункціональні вимоги, які система повинна задовольняти.
озробка архітектури системи. Спроектувати компоненти системи, включаючи базу даних, серверні алгоритми та інтеграцію з веб-інтерфейсом.
ибір методів рекомендацій. Визначити підходи, які будуть використані (контентна фільтрація, колаборативна фільтрація, гібридні моделі).
еалізація системи. Розробити прототип системи, включаючи базу даних та алгоритми рекомендацій.
кспериментальна оцінка. Провести тестування системи на реальних або синтетичних даних, оцінити точність і продуктивність рекомендацій.

2.4 Вхідні дані

Система рекомендацій буде використовувати дані, що надходять від користувачів, а також характеристики контенту. Основними джерелами вхідних даних є:

- взаємодія користувачів: Дані про перегляди оголошень, додавання до обраного, кліки по оголошеннях та інші дії. Ці дані будуть зберігатися у вигляді подій із зазначенням користувача, часу та типу дії. Основний акцент робиться на динаміці взаємодій, щоб забезпечити актуальність рекомендацій;
- характеристики об’єктів нерухомості: Дані про оголошення, такі як розташування (місто, район), кількість кімнат, площа, вартість оренди та додаткові характеристики (наприклад, наявність меблів або близькість до транспорту). Ці метадані дозволяють краще зрозуміти переваги користувачів і формувати рекомендації на основі подібності об’єктів;
- історія вподобань: Для кожного користувача зберігаються раніше обрані або переглянуті об’єкти, що дозволяє використовувати контентну фільтрацію для визначення його інтересів.

Усі ці дані будуть інтегровані в базу даних для подальшої обробки алгоритмами рекомендацій, які враховують як індивідуальні вподобання, так і загальну динаміку взаємодій на платформі.

2.5 Вихідні результати

Очікувані результати системи рекомендацій є критичними для оцінки її ефективності та практичної цінності. Одним із ключових результатів буде формування списку персоналізованих рекомендацій для кожного користувача. Цей список забезпечить релевантність запропонованих об'єктів нерухомості на основі історії взаємодій користувача, його вподобань та загальної поведінки на платформі.

Крім того, система повинна забезпечувати інтеграцію рекомендацій у веб-інтерфейс сайту. Веб-сайт матиме спеціальний розділ, де користувачі зможуть переглядати свої персоналізовані рекомендації. Це дозволить створити безперервний користувацький досвід і підвищити залученість користувачів до платформи.

Для оцінки ефективності системи будуть застосовані метрики якості рекомендацій, такі як точність, повнота та F1-міра. Точність визначатиме, наскільки релевантними є запропоновані об'єкти, повнота оцінюватиме здатність системи врахувати всі вподобання користувача, а F1-міра забезпечуватиме збалансований підхід до аналізу результатів. Також планується використання бізнес-орієнтованих метрик, таких як зростання кліків або конверсій у реальні запити.

Загалом, результати системи не тільки покращать користувацький досвід, але й забезпечать підвищення ефективності роботи платформи, сприятимуть утриманню клієнтів і залученню нових.

2.6 Вимоги до системи

Система рекомендацій повинна мати широкий набір функцій, які покращують користувацький досвід і підвищують ефективність платформи. Одним із ключових завдань є генерація персоналізованих рекомендацій на основі історії взаємодій користувача, таких як перегляди, кліки та додавання об'єктів до обраного. Ця функція забезпечує формування релевантних пропозицій об'єктів нерухомості. Водночас система повинна враховувати різноманітні типи взаємодій користувачів, включаючи взаємодії з фільтрами пошуку та вибір об'єктів для порівняння. Ще однією важливою функцією є інтеграція з веб-інтерфейсом, яка дозволяє рекомендаціям бути доступними у динамічному режимі через API-запити та веб-інтерфейс. Крім того, система повинна бути адаптивною, враховуючи змінні вподобання користувачів і оперативно оновлюючи рекомендації у разі змін в інформації про об'єкти нерухомості.

Для стабільної роботи системи важливо врахувати низку нефункціональних аспектів. Генерація рекомендацій повинна відбуватися швидко, не перевищуючи однієї секунди для кожного запиту, навіть за умов високого навантаження. Масштабованість системи є ще однією важливою вимогою: вона повинна обробляти великі обсяги даних, включаючи мільйони

активних користувачів і об'єктів нерухомості. Система також має бути стійкою до збоїв. Наприклад, у разі помилок у базі даних або серверах вона повинна частково зберігати функціональність, забезпечуючи доступ до рекомендацій з наявних даних. Безпека даних є критичним аспектом: конфіденційна інформація користувачів повинна бути захищена сучасними стандартами шифрування.

Реалізація системи рекомендацій базується на перевірених і ефективних технологіях. Для серверної логіки використовується мова програмування PHP. Реляційна база даних MySQL буде зберігати всю інформацію про користувачів, об'єкти нерухомості та історію взаємодій. Веб-інтерфейс розроблятиметься з використанням HTML, CSS і JavaScript для відображення рекомендацій у зручному для користувачів вигляді. Алгоритми рекомендацій будуть реалізовані нативно, без використання сторонніх бібліотек, що забезпечить більшу гнучкість і можливість адаптації під специфічні вимоги проекту.

Розробка системи рекомендацій спрямована на досягнення конкретних бізнес-цілей. Персоналізовані рекомендації підвищать залученість користувачів, збільшуючи ймовірність тривалого використання платформи. Також система сприятиме зростанню конверсій, оскільки рекомендації мотивуватимуть користувачів частіше залишати запити на оренду об'єктів. Крім того, оптимізація пошуку завдяки швидкому доступу до потрібного контенту зменшить витрати часу користувачів і підвищить їхню задоволеність платформою.

Ці вимоги формують всебічний підхід до розробки системи рекомендацій, орієнтуючись на забезпечення високої якості роботи платформи як для користувачів, так і для досягнення бізнес-цілей.

РОЗДІЛ 3
ПРОЕКТНІ РІШЕННЯ

3.1 Опис архітектури системи

Архітектура системи рекомендацій для платформи оренди нерухомості базується на моделі MVC (Model-View-Controller), що забезпечує чітке розділення логіки програми, представлення даних та управління користувацькими запитами. Така структура сприяє модульності, підтримуваності та масштабованості системи, дозволяючи ефективно обробляти великі обсяги даних і забезпечувати високий рівень продуктивності.

Модель відповідає за роботу з даними. Це включає зберігання, доступ, обробку та маніпулювання даними в базі. У нашій системі модель реалізована через класи PHP, які взаємодіють із базою даних MySQL. База даних організована у вигляді таблиць, які зберігають інформацію про користувачів, об'єкти нерухомості, історію взаємодій та результати рекомендацій. Для оптимізації роботи бази даних використовуються індекси, що прискорюють пошук, та реплікація для підвищення доступності.

Рівень представлення відповідає за відображення даних користувачеві. Веб-інтерфейс створений за допомогою HTML, CSS і JavaScript, що забезпечує інтуїтивно зрозумілий дизайн і зручність використання. Представлення використовує динамічні шаблони, які дозволяють відображати результати рекомендацій у вигляді списків, карток або слайдерів. Асинхронні запити, реалізовані через AJAX, забезпечують оновлення даних у реальному часі без необхідності перезавантаження сторінки.

Контролер виступає посередником між користувачем, моделлю та представленням. Він обробляє запити від користувача, викликає відповідні методи моделі для роботи з даними і передає результати до рівня представлення. Наприклад, при запиті рекомендацій контролер звертається до моделі для

обчислення релевантних об'єктів і повертає їх у вигляді JSON-об'єкта, який потім обробляється на рівні представлення.

Взаємодія між компонентами системи здійснюється через чітко визначені інтерфейси. Користувач взаємодіє із системою через веб-інтерфейс, який надсилає запити до контролера. Контролер обробляє ці запити, звертається до моделі для виконання бізнес-логіки або отримання даних і передає результат назад до представлення. Така організація дозволяє легко розширювати функціональність системи, додаючи нові моделі або представлення без змін у наявній архітектурі.

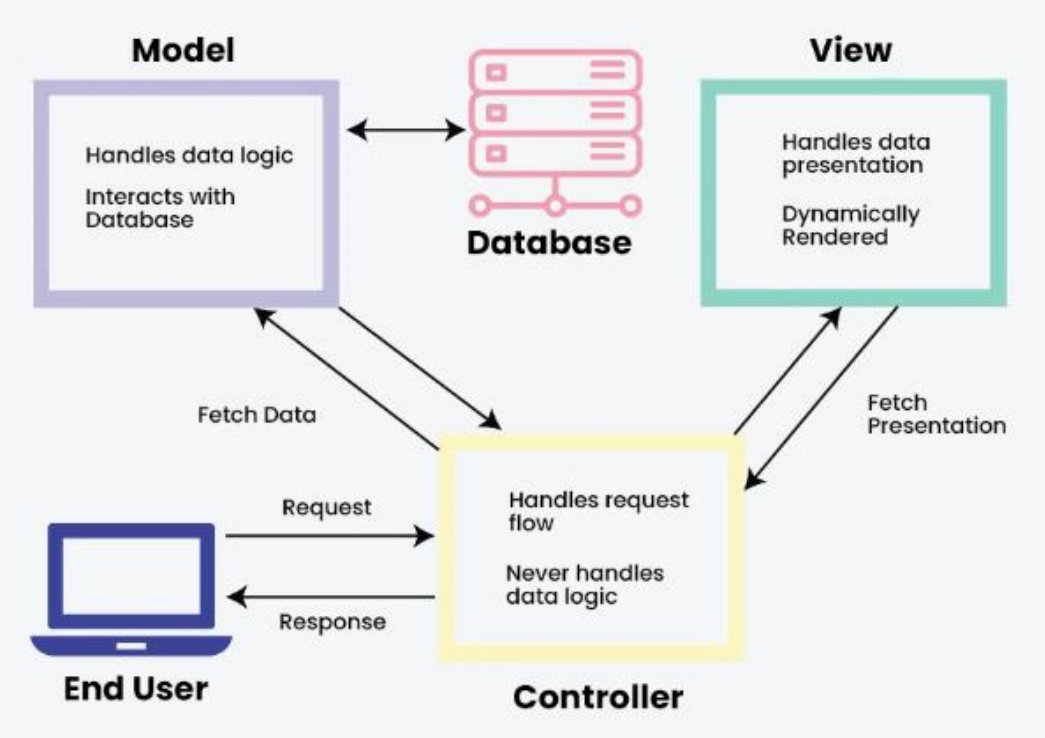

Рисунок 3.1 Дизайн-схема MVC архітектури

У рамках MVC архітектури логіка рекомендацій реалізована в моделі. Вона використовує поєднання контентної та колаборативної фільтрації. Контентна фільтрація аналізує характеристики об'єктів, такі як місцезнаходження, площа, ціна та кількість кімнат, тоді як колаборативна фільтрація враховує взаємодії інших користувачів із платформою. Це поєднання дозволяє створювати рекомендації, які враховують як індивідуальні вподобання, так і загальні тренди.

Використання архітектури MVC у системі рекомендацій забезпечує низку переваг. По-перше, чітке розділення відповідальностей між компонентами дозволяє спростити підтримку і тестування системи. По-друге, модульність архітектури сприяє легкому впровадженню нових функцій, таких як розширені алгоритми рекомендацій або нові форми представлення даних. По-третє, MVC сприяє масштабованості, оскільки дозволяє незалежно розвивати клієнтський, серверний і модельний компоненти.

Для забезпечення продуктивності система використовує оптимізації на кожному рівні. На рівні моделі впроваджені індекси бази даних і механізми кешування, що знижують навантаження на сервер. На рівні контролера застосовуються асинхронні обчислення, які дозволяють швидко обробляти запити. Представлення оптимізовано для швидкого завантаження сторінок і мінімізації використання ресурсів клієнта.

Таким чином, архітектура MVC забезпечує стабільність, продуктивність і гнучкість системи рекомендацій, створюючи надійну основу для подальшого розвитку платформи.

3.2 Процес розробки сайту

Розробка сайту з використанням архітектури Model-View-Controller (MVC) є ключовим підходом, що дозволяє забезпечити чітке розділення логіки програми, представлення даних і обробки запитів. Архітектура MVC спрощує підтримку, тестування та розширення коду, що є критично важливим у сучасній веб-розробці. У рамках цього проєкту архітектура MVC стала основою для створення платформи оренди нерухомості, яка відповідає вимогам продуктивності, масштабованості та зручності для користувачів.

Для реалізації сайту використовувалися такі технології, як HTML, CSS і для роботи з базою даних. Крім того, було використано фреймворк Bootstrap, який забезпечив швидку реалізацію адаптивного дизайну та дотримання

сучасних стандартів веб-дизайну. Сайт включає основний набір функціональних можливостей, таких як реєстрація користувачів, перегляд рекомендацій, збереження обраних об'єктів і зміна налаштувань профілю.

Для реалізації сайту використовувалися сучасні веб-технології. Основою фронтенд-частини стали HTML і CSS, які забезпечують базову структуру та стилізацію сторінок. Для створення інтерактивності застосовувався JavaScript, який дозволяє реалізовувати функції, такі як зміна теми з світлої на темну та динамічне оновлення даних без перезавантаження сторінки.

Серверна частина була реалізована за допомогою PHP, що забезпечило обробку запитів і взаємодію з базою даних MySQL. Для зберігання інформації про користувачів, їхні обрані об'єкти та історію взаємодій використовувалася реляційна база даних. Її структура була спроєктована з урахуванням принципів нормалізації для забезпечення цілісності даних і оптимізації запитів.

Одним із важливих елементів проєкту став фреймворк Bootstrap, який забезпечив адаптивність дизайну. Завдяки цьому сайт коректно відображається на різних пристроях, включаючи комп'ютери, планшети та смартфони. Компоненти Bootstrap, такі як навігаційні панелі, форми та кнопки, були активно використані для створення зручного інтерфейсу.

Функціональні сторінки сайту

Сайт включає кілька основних сторінок, які забезпечують необхідний функціонал для користувачів.

Головна сторінка (Home) служить точкою входу для відвідувачів сайту. Вона містить перелік останніх доданих об'єктів нерухомості, а також найпопулярніші пропозиції. Інтуїтивний дизайн цієї сторінки дозволяє користувачам швидко ознайомитися з доступними об'єктами. Крім того, головна сторінка містить зручний сайдбар для вибору фільтрів, який дозволяє звузити пошук об'єктів за такими критеріями, як ціна, кількість кімнат, тип нерухомості та місцезнаходження. Сайдбар створений із використанням HTML, CSS і основі обраних фільтрів. При виборі параметрів у сайдбарі дані передаються

через HTTP-запит, після чого сервер обробляє запит і генерує відповідь із новими результатами, які відображаються на сторінці. Такий підхід забезпечує точність фільтрації та дозволяє користувачам отримувати актуальну інформацію без зайвих складнощів.

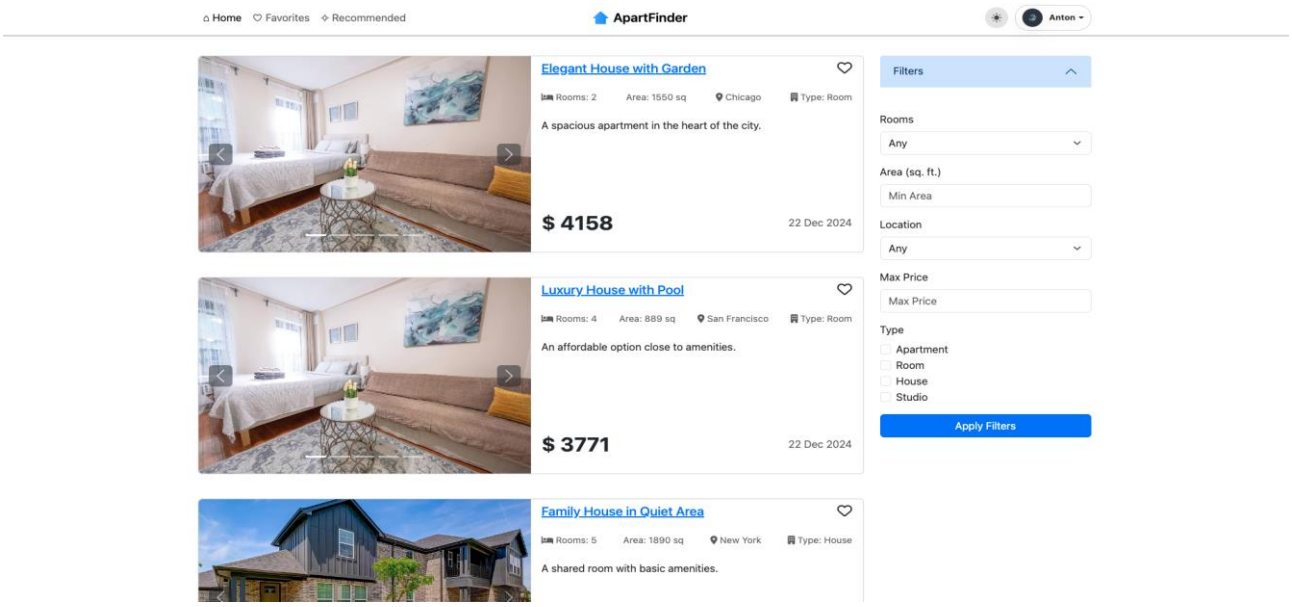

Рисунок 3.2 – Сторінка Home

На сторінці Favorites користувачі можуть переглядати свої обрані об'єкти. Ця функція реалізована через таблицю, яка зберігає зв'язок між користувачами та їхніми збереженими об'єктами. Інтерактивний інтерфейс дозволяє легко видаляти або додавати нові об'єкти до списку обраних.

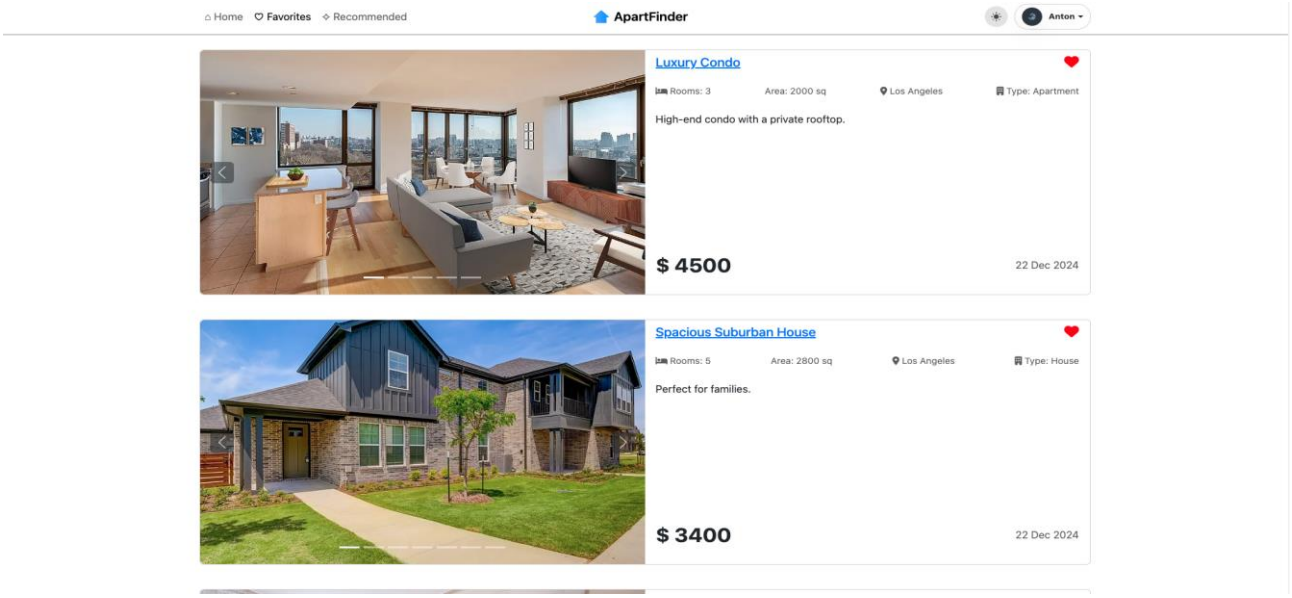

Рисунок 3.3 – Сторінка Favorites

Сторінка Recommended забезпечує персоналізовані рекомендації, які генеруються на основі історії взаємодій користувача. Алгоритм рекомендацій використовує дані про перегляди, кліки та додавання до обраного для формування списку найбільш релевантних пропозицій. Цей процес включає аналіз усіх взаємодій користувача, таких як кількість кліків по оголошеннях у певній категорії чи регіоні, а також додавання до обраного об'єктів з конкретними характеристиками (наприклад, певна кількість кімнат або діапазон цін).

Рекомендації відображаються у вигляді карток, що містять основну інформацію про об'єкт і фотографії. Кожна картка включає короткий опис об'єкта, його ключові характеристики, такі як площа, ціна, тип і розташування, а також головне зображення для створення візуального враження. Окрім цього, картки оснащені кнопками для взаємодії, наприклад, можливістю додати об'єкт до обраного або перейти до детального перегляду.

Сторінка також підтримує інтерактивні функції, такі як пагінація, що дозволяє користувачам зручно переглядати великий список рекомендацій. Завдяки ретельно продуманій структурі сторінки користувачі мають змогу швидко знаходити об'єкти, які відповідають їхнім потребам, і взаємодіяти з ними в кілька кліків. Це підвищує загальний рівень зручності використання платформи.

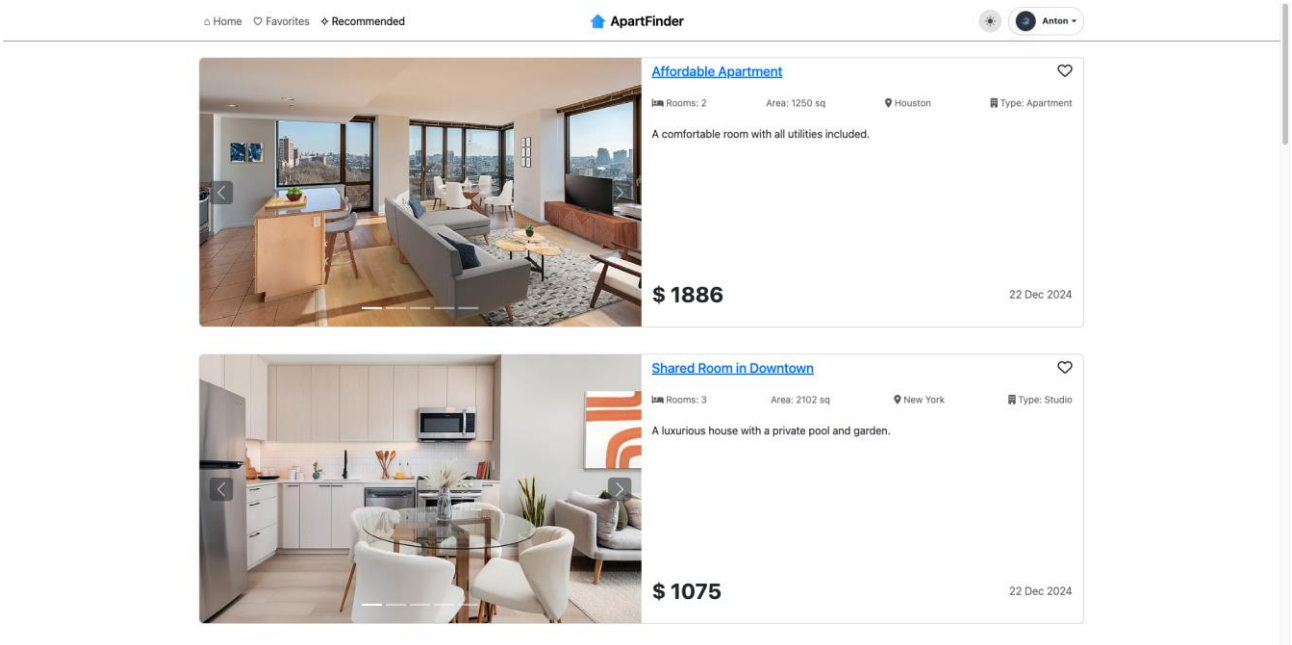

Рисунок 3.4 – Сторінка Recommended

Сторінки Login і Registration забезпечують критично важливий функціонал для роботи сайту, зокрема входу та реєстрації нових користувачів. Для реєстрації користувачі вводять свою електронну пошту, пароль, ім'я та іншу необхідну інформацію, яка забезпечує ідентифікацію в системі. Дані ретельно перевіряються на етапі введення: валідація відбувається як на клієнтській, так і на серверній стороні. Наприклад, перевіряється відповідність пароля встановленим вимогам, таким як довжина та складність, що гарантує високий рівень безпеки.

При вході в систему користувачі вводять свою електронну пошту та пароль, після чого сервер здійснює перевірку відповідності введених даних збереженим у базі даних. Для захисту паролів використовується хешування за допомогою сучасних алгоритмів, що виключає можливість компрометації навіть у разі витоку бази даних.

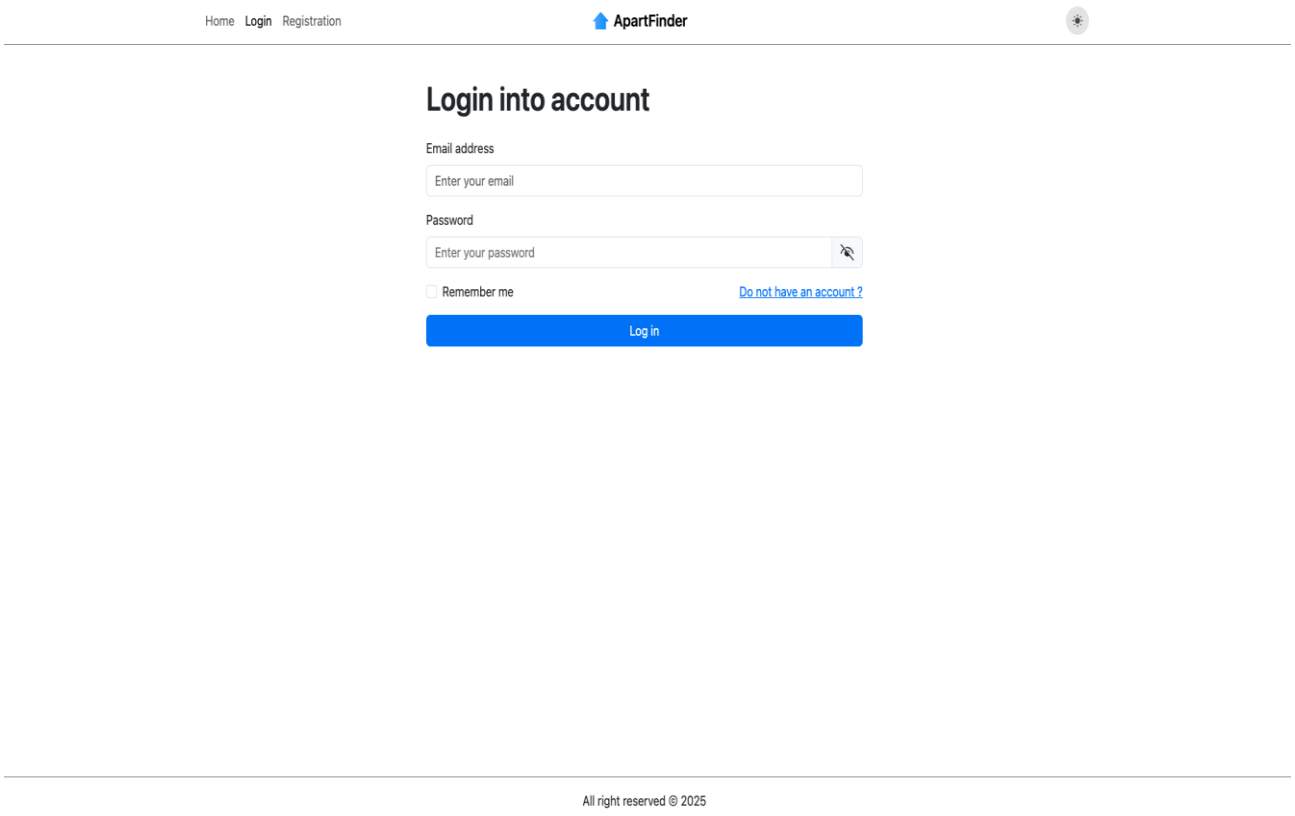

Рисунок 3.5 – Сторінка Login

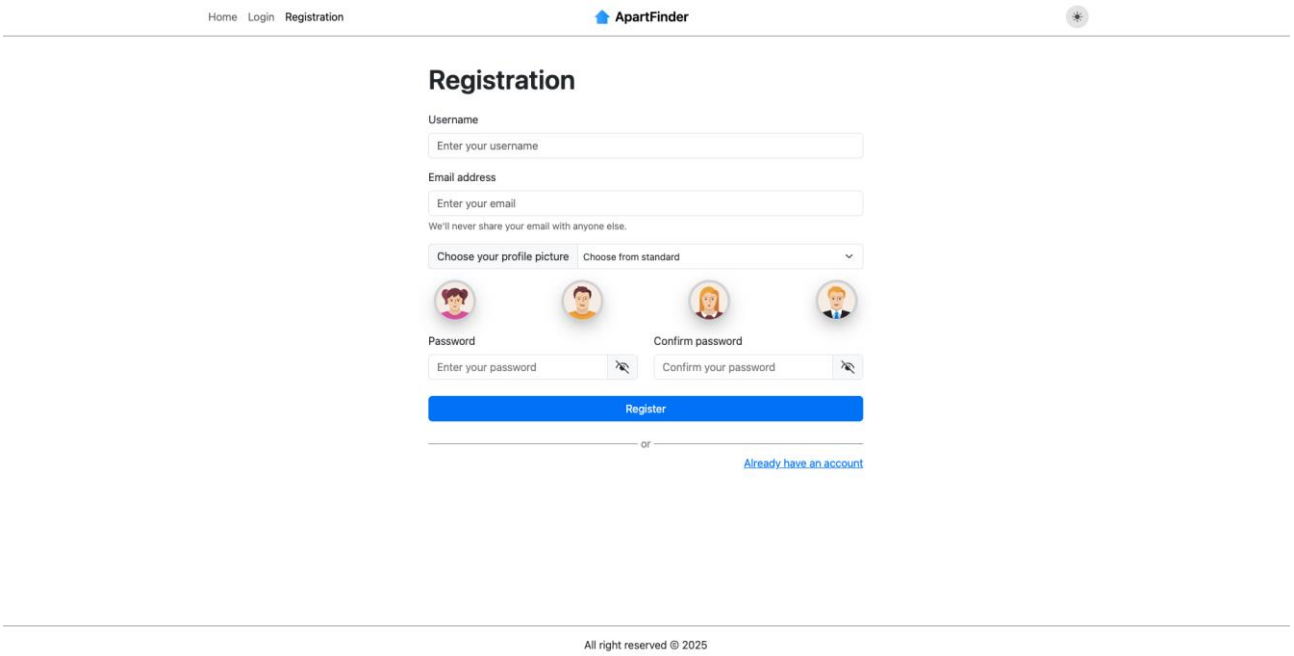

Рисунок 3.6 – Сторінка Registration

Сторінка Profile є важливою частиною функціоналу сайту, яка надає користувачам можливість керувати своїми особистими налаштуваннями. Однією з ключових функцій є зміна аватара. Завантаження зображень реалізовано із застосуванням механізмів перевірки формату файлу (наприклад, JPEG, PNG) і обмеження розміру файлу, що забезпечує коректність завантаження та зменшує ризик додавання некоректних або шкідливих даних до системи. Після завантаження зображення обробляється сервером, а його зменшена копія зберігається в базі даних для оптимізації відображення на сторінках.

Іншою важливою функцією є зміна пароля, яка реалізована з акцентом на безпеку. Перед тим як користувач може оновити свій пароль, він повинен підтвердити поточний, що мінімізує ризик несанкціонованого доступу. Новий пароль перевіряється на відповідність встановленим вимогам, таким як мінімальна довжина, використання цифр і символів. Після валідації пароль хешується за допомогою сучасного алгоритму bcrypt і зберігається у базі даних, що гарантує високий рівень безпеки навіть у разі витоку даних.

Сторінка профілю має зручний та інтуїтивно зрозумілий інтерфейс, створений за допомогою Bootstrap. Елементи керування, такі як кнопки та форми,

адаптовані для мобільних пристроїв, що забезпечує доступ до функцій профілю незалежно від типу пристрою. Завдяки інтерактивності та простоті використання сторінка Profile підвищує загальний рівень задоволеності користувачів та ефективність роботи платформи.

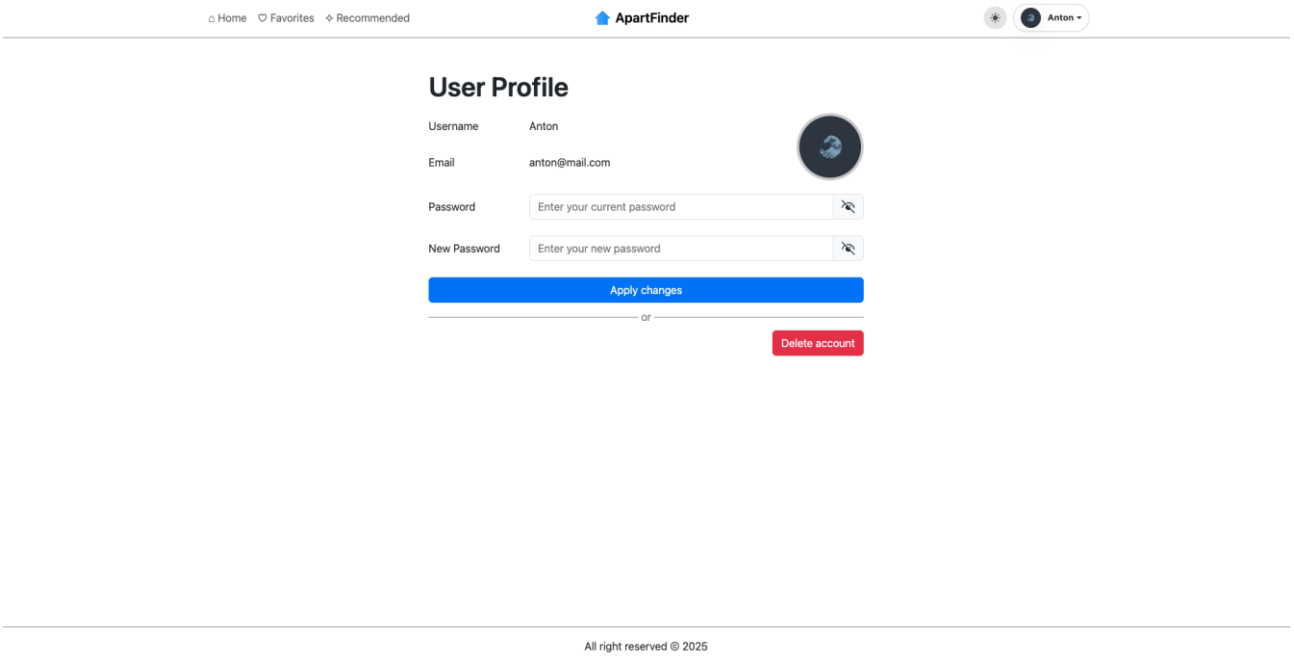

Рисунок 3.7 – Сторінка Profile

Тематичний режим

Одна з ключових функцій сайту — можливість змінювати тему з світлої на темну. Ця опція була реалізована за допомогою механізмів Bootstrap у поєднанні з JavaScript і CSS змінними, що дозволяють динамічно змінювати кольорову схему залежно від вподобань користувача. Bootstrap надає попередньо налаштовані класи для кольорових тем, що значно прискорило реалізацію даної функціональності. Завдяки гнучкості CSS змінних, теми можна легко адаптувати для специфічних потреб або додати нові стилі.

Вибір теми інтегровано в інтерфейс сайту через зручний перемикач, розташований у навігаційній панелі. Коли користувач обирає темну або світлу тему, JavaScript динамічно змінює відповідні класи на елементах сторінки, а налаштування зберігаються у файлах cookie. Це забезпечує, що вибір теми

зберігається між сеансами, і користувач бачить сайт у бажаній кольоровій схемі при наступному відвідуванні.

Реалізація темного режиму покращує користувацький досвід, особливо для людей, які працюють у нічний час або віддають перевагу менш яскравому інтерфейсу. Крім того, функція була протестована на різних пристроях і браузерах, щоб забезпечити її стабільну роботу. Темний режим не лише додає естетичної привабливості сайту, а й відповідає сучасним трендам у веб-дизайні, сприяючи зручності використання платформи.

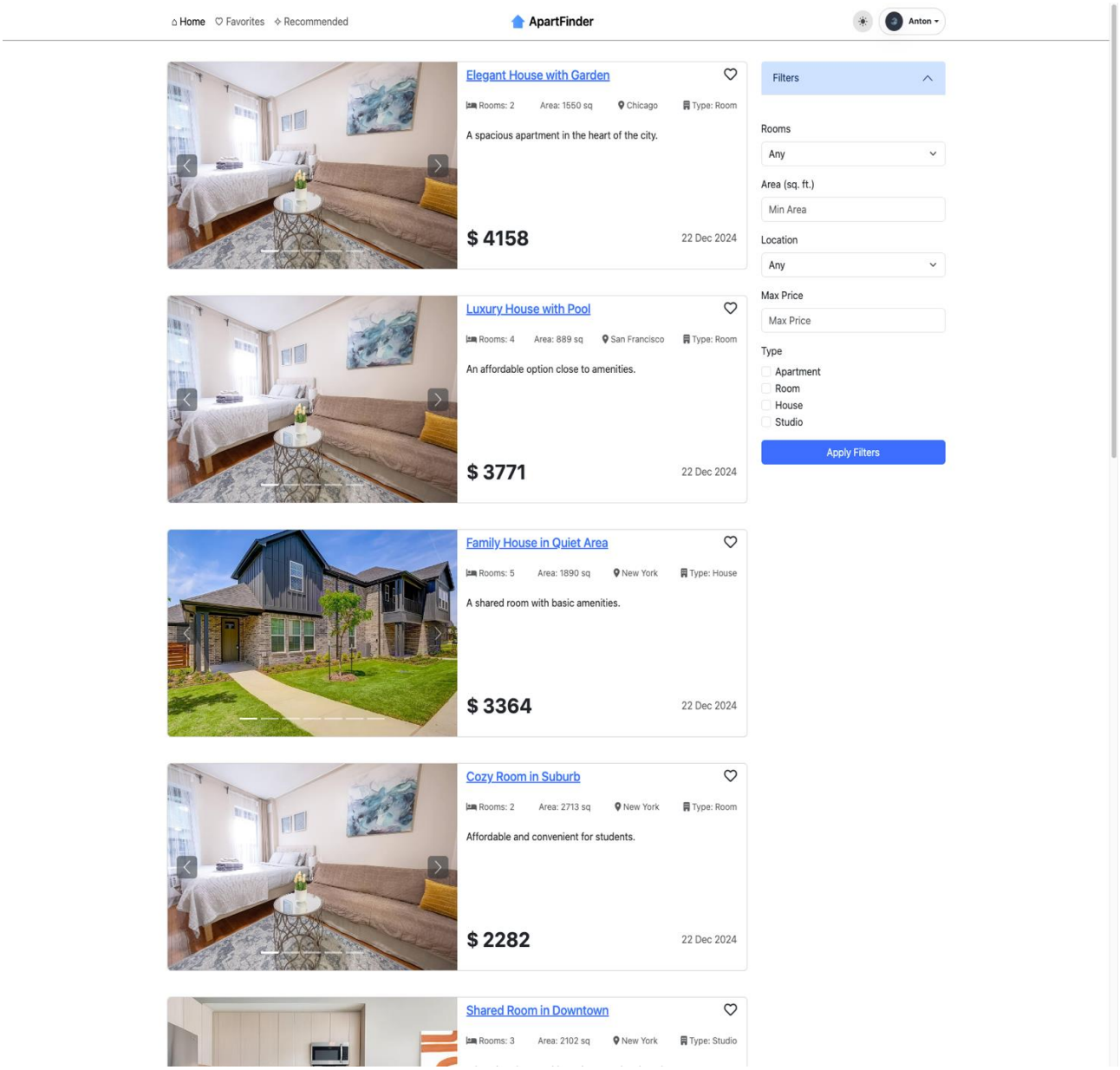

Рисунок 3.8 – Світла тема сторінки Home

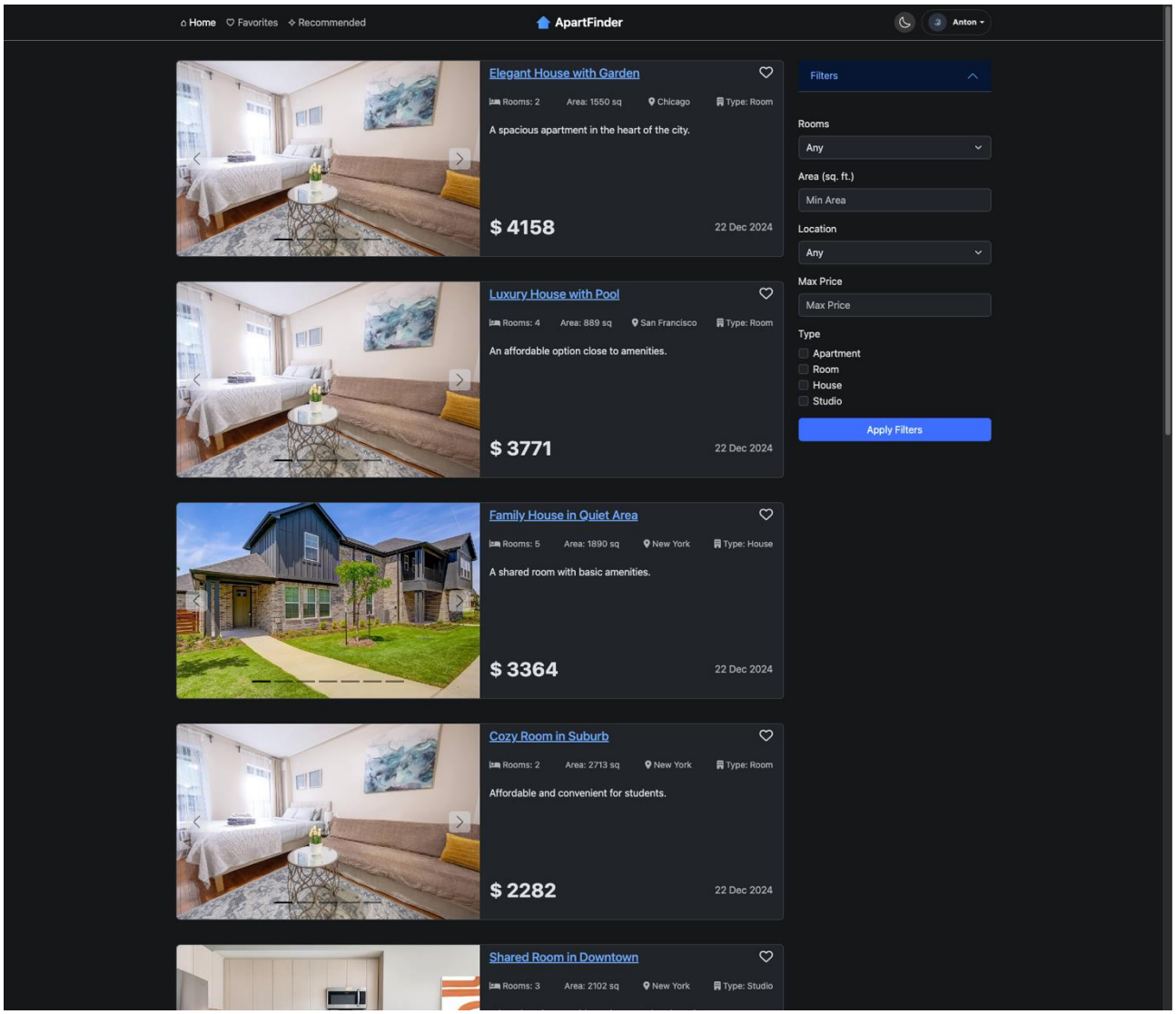

Рисунок 3.9 – Темна тема сторінки Home

Розробка сайту складалася з кількох етапів. Спочатку було спроєктовано базу даних, яка включає таблиці для зберігання інформації про користувачів, об'єкти нерухомості, їхні зображення, взаємодії та рекомендації. На наступному етапі були реалізовані основні компоненти архітектури MVC: моделі, контролери та вигляди. Особлива увага приділялася безпеці даних, зокрема хешуванню паролів і захисту від SQL-ін'єкцій.

Для кожної сторінки були створені окремі шаблони, які інтегрувалися з контролерами для отримання динамічних даних із моделі. Наприклад, на сторінці рекомендацій контролер отримує дані з бази, обробляє їх за допомогою алгоритмів і передає вигляду для відображення.

Заключним етапом стало тестування сайту, яке включало функціональне тестування (перевірка роботи кожної сторінки), тестування адаптивності (перевірка коректного відображення на різних пристроях) і навантажувальне тестування для оцінки продуктивності під час великої кількості запитів.

Розробка сайту на архітектурі MVC дозволила створити платформу, яка відповідає вимогам сучасної веб-розробки. Завдяки використанню PHP і MySQL у поєднанні з HTML, CSS, JavaScript і Bootstrap вдалося забезпечити високу продуктивність, адаптивність і зручність для користувачів. У подальшому планується розширення функціоналу, зокрема впровадження додаткових алгоритмів рекомендацій і оптимізація продуктивності системи для роботи з великими обсягами даних.

3.3 Реалізація алгоритмів рекомендацій

Система рекомендацій для платформи оренди нерухомості реалізує алгоритми контентної фільтрації, колаборативної фільтрації та гібридного підходу з урахуванням специфіки проекту. Реалізація цих методів базується на використанні нативного PHP без сторонніх бібліотек, що дозволяє гнучко адаптувати алгоритми до вимог системи та структури даних.

Контентна фільтрація

Контентна фільтрація базується на аналізі характеристик об'єктів нерухомості, таких як розташування, площа, кількість кімнат і ціна. Цей підхід дозволяє порівнювати властивості об'єктів із вподобаннями користувачів і пропонувати ті об'єкти, які відповідають їхнім інтересам. У реалізації контентної фільтрації використовується механізм порівняння векторів характеристик. Кожен об'єкт представлений як набір атрибутів у вигляді вектора, що включає такі поля бази даних, як city, area, rooms, price, type. Алгоритм визначає схожість між об'єктами за допомогою косинусної схожості, що дозволяє ефективно знаходити найближчі за властивостями об'єкти. Наприклад, якщо користувач переглядав або додавав до обраного об'єкти з певною кількістю кімнат та ціною, система

знаходить схожі об'єкти в базі даних, використовуючи SQL-запити для відбору відповідних записів.

На цьому етапі відбувається ранжування результатів, що забезпечує максимальну релевантність рекомендацій. Контентна фільтрація є особливо ефективною для користувачів із чіткими уподобаннями, оскільки враховує конкретні характеристики об'єктів. Проте вона стикається з проблемою "холодного старту", коли недостатньо інформації про нових користувачів або об'єкти, що обмежує її застосування в таких випадках.

Колаборативна фільтрація

Колаборативна фільтрація враховує дані про взаємодії інших користувачів із платформою. Вона базується на припущенні, що користувачі зі схожими вподобаннями цікавляться подібними об'єктами. Реалізація цього підходу в системі виконується через аналіз історії взаємодій користувачів, таких як перегляди, кліки та додавання до обраного. Всі ці дії записуються в таблицю interaction, яка містить наступні поля:

- u
- s
- a
- p
- action – тип взаємодії;
- a
- – час взаємодії.
- i
- d – у
- Система ідентифікує користувачів зі схожими діями, використовуючи
- алгоритми обчислення схожості, такі як косинусна схожість або коефіцієнт
- I
- d
- Жаккара. На основі цієї інформації генеруються рекомендації, які пропонують об'єкти, обрані іншими користувачами з подібними вподобаннями. SQL-запити до таблиці interaction дозволяють виділити об'єкти, які мають найбільшу кількість взаємодій серед схожих користувачів.

Цей підхід дозволяє створювати несподівані рекомендації, які виходять за межі звичних уподобань користувача, збагачуючи його досвід. Однак, як і у випадку контентної фільтрації, колаборативна фільтрація має свої обмеження. Наприклад, "проблема холодного старту" для нових користувачів або об'єктів, а

також розрідженість даних у великих системах можуть ускладнювати генерацію точних рекомендацій.

Гібридні моделі

Гібридні моделі поєднують переваги контентної та колаборативної фільтрації, мінімізуючи недоліки кожного з методів. У проекті цей підхід реалізовано через інтеграцію результатів обох методів. Контентна фільтрація використовується для роботи з новими користувачами або об'єктами, пропонуючи базові рекомендації, які враховують характеристики об'єктів. Для користувачів із достатньою історією взаємодій застосовується колаборативна фільтрація, яка додає соціальний контекст до рекомендацій.

Результати обох підходів комбінуються для створення остаточного списку рекомендацій. Наприклад, зважені бали з обох алгоритмів підсумовуються для кожного об'єкта, що дозволяє ранжувати результати за релевантністю. Ранжування об'єктів у цьому списку здійснюється на основі таких даних, як кількість взаємодій з об'єктом, схожість характеристик із профілем користувача та популярність серед схожих користувачів.

Технічні аспекти реалізації

Зберігання даних про взаємодії користувачів здійснюється через таблицю interaction, а інформація про об'єкти нерухомості — через таблицю apart. Поля таблиці apart включають:

– унікальний ідентифікатор об'єкта;
– назва об'єкта;
– розташування об'єкта;
– площа об'єкта;
– кількість кімнат;
– вартість оренди;
– тип об'єкта.

Для обчислення схожості та генерації рекомендацій використовуються спеціальні методи PHP, що інтегрують алгоритми контентної та колаборативної фільтрації. SQL-запити до таблиці apart забезпечують вибір релевантних об'єктів,

а крос-аналіз із таблицею interaction дозволяє враховувати поведінку користувачів. Наприклад, для швидкого вибору популярних об'єктів використовуються індекси бази даних за полями apart_id і user_id.

Крім того, для підвищення продуктивності впроваджено механізми кешування результатів найчастіших запитів, що дозволяє зменшити навантаження на сервер під час роботи системи з великими обсягами даних.

Переваги реалізованих алгоритмів

Запропонована реалізація алгоритмів забезпечує високу точність і адаптивність системи. Контентна фільтрація дозволяє враховувати індивідуальні характеристики об'єктів, що особливо корисно для нових користувачів. Колаборативна фільтрація додає соціальний вимір до рекомендацій, дозволяючи користувачам відкривати для себе нові об'єкти. Гібридний підхід поєднує ці методи, забезпечуючи універсальність і високу якість рекомендацій навіть для великих обсягів даних і різноманітної аудиторії. Це дозволяє системі працювати ефективно як із новими користувачами, так і з тими, хто активно взаємодіє з платформою.

3.4 Структура бази даних: зберігання взаємодій користувачів, контенту та метаданих

Для реалізації системи рекомендацій використовується реляційна база даних, яка складається з кількох основних таблиць. Ця база забезпечує зберігання та організацію даних про користувачів, об'єкти нерухомості, взаємодії користувачів і результати роботи системи рекомендацій. Основна мета структури — забезпечити ефективну обробку запитів і високу продуктивність системи.

Основні таблиці бази даних

Таблиця users зберігає дані про користувачів платформи. Це включає унікальний ідентифікатор користувача (id), його ім'я, електронну пошту, дату реєстрації та час останнього оновлення профілю. Ця таблиця є центральною для

ідентифікації користувачів і забезпечення зв'язків з іншими таблицями бази даних.

Таблиця apart містить інформацію про об'єкти нерухомості, які доступні для оренди. Кожен запис включає такі поля, як унікальний ідентифікатор об'єкта (id), назва, розташування (city), площа (area), кількість кімнат (rooms), ціна (price), тип об'єкта (type) та детальний опис. Додатково зберігаються дати створення та оновлення запису, що дозволяє відстежувати зміни в об'єктах.

Для зберігання зображень об'єктів використовується таблиця apart_images. Вона містить посилання на таблицю apart через поле apart_id, яке визначає, до якого об'єкта належить зображення. Поле image_path зберігає шлях до зображення, а поле is_main визначає, чи є це зображення головним для відображення в списку чи деталізації об'єкта.

Таблиця interaction фіксує всі взаємодії користувачів із системою, такі як перегляди оголошень, кліки та додавання до обраного. Основними полями є user_id (ідентифікатор користувача), apart_id (ідентифікатор об'єкта), action (тип взаємодії) і created_at (час взаємодії). Ці дані є ключовими для аналізу поведінки користувачів і створення персоналізованих рекомендацій.

Таблиця recommendations містить попередньо згенеровані рекомендації для користувачів. Поля включають user_id, apart_id, рейтинг рекомендації (score) і час створення рекомендації (created_at). Ця таблиця дозволяє зберігати результати роботи алгоритмів і швидко надавати їх користувачам без необхідності повторного обчислення.

Таблиця 3.1 – Структура бази даних

Таблиця	**Опис**	**Основні поля**
	Зберігає інформацію про користувачів	

Продовження таблиці 3.1

	Дані про об'єкти нерухомості	
	Зображення об'єктів нерухомості	
	Взаємодії користувачів із платформою	
	Рекомендації для користувачів	

Основні зв’язки між таблицями забезпечують взаємодію між різними компонентами системи. Таблиця users пов’язана з таблицею interaction за принципом «один до багатьох», що означає, що один користувач може здійснювати багато взаємодій, які фіксуються у таблиці interaction. Це дозволяє відстежувати дії користувачів, такі як перегляди, кліки чи додавання до обраного.

Зв’язок між таблицями apart та apart_images відображає співвідношення «один до багатьох». Один об'єкт нерухомості може мати кілька зображень, які зберігаються в таблиці apart_images. Це забезпечує зручне управління медіафайлами для кожного оголошення.

Таблиця users також пов’язана з таблицею recommendations, де для кожного користувача зберігаються індивідуальні рекомендації. Це дозволяє системі персоналізувати контент і пропонувати об’єкти, що відповідають інтересам і вподобанням користувача.

Зв’язок між таблицями apart та interaction також є важливим. Кожен об'єкт нерухомості може бути частиною багатьох взаємодій користувачів. Це дає змогу аналізувати популярність і активність об'єктів, що впливає на точність рекомендацій.

Нарешті, таблиці apart та recommendations взаємодіють через ідентифікатор об’єкта. Рекомендації прив’язуються до конкретних об'єктів, що

дозволяє системі чітко визначати, які оголошення мають бути запропоновані користувачам.Такий підхід до моделювання зв’язків забезпечує гнучкість і ефективність системи в управлінні даними.

Для забезпечення цілісності даних усі зовнішні ключі налаштовані з обмеженнями ON DELETE CASCADE. Це означає, що при видаленні користувача автоматично видаляються всі його взаємодії та рекомендації. Використання індексів для полів user_id, apart_id і action у таблиці interaction забезпечує швидкий доступ до необхідних даних.

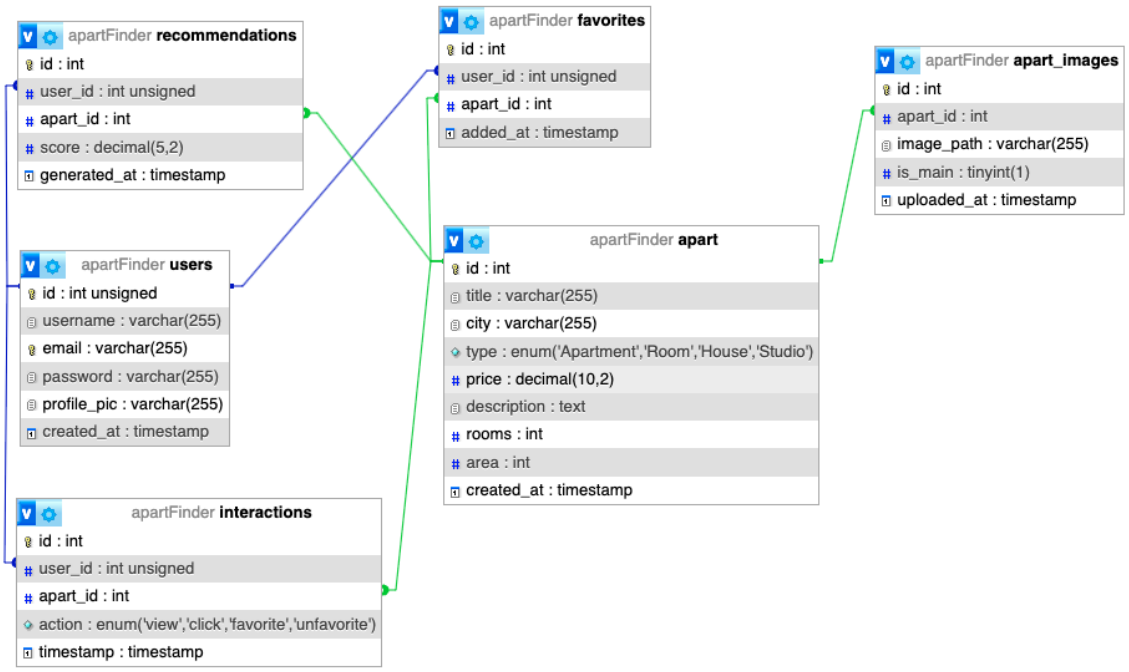

Рисунок 3.10 – Взаємозв’язок таблиць в базі даних

Кешування використовується для зменшення навантаження на базу даних при обробці часто виконуваних запитів. Наприклад, рекомендації для активних користувачів можуть бути збережені у кеші, що дозволяє зменшити час відповіді системи.

Одним із методів оптимізації є архівація старих даних. Наприклад, взаємодії, які були створені більше року тому, можуть бути переміщені до

окремої архівної таблиці. Це зменшує обсяг основних таблиць, прискорює роботу системи та полегшує адміністрування бази даних.

Масштабованість системи забезпечується через розділення бази даних на шардінг. Це означає, що таблиці можуть бути розділені за певними критеріями, такими як регіон або категорія об'єктів. Наприклад, таблиця apart може бути поділена на декілька частин залежно від географічного розташування об'єктів. Це знижує навантаження на кожен сервер і дозволяє системі ефективно працювати з великим числом одночасних запитів.

Структура бази даних є ключовим елементом функціонування системи рекомендацій. Завдяки ретельному проектуванню таблиць, налагодженню зв'язків між ними та використанню сучасних підходів до оптимізації, база даних підтримує високу продуктивність і масштабованість системи. Ця структура дозволяє обробляти великі обсяги інформації, забезпечувати персоналізований користувацький досвід і швидко реагувати на запити. У майбутньому структура може бути розширена або модифікована для підтримки нових функцій або зростання кількості користувачів.

3.5 Особливості обробки великих обсягів даних

Обробка великих обсягів даних є важливим аспектом розробки сучасних інформаційних систем, зокрема, систем рекомендацій. У контексті платформи оренди нерухомості ці дані включають інформацію про об'єкти, такі як ціна, площа та розташування, а також взаємодії користувачів із платформою. Зростаючий обсяг таких даних вимагає застосування спеціалізованих методів обробки для забезпечення стабільної роботи системи.

Однією з основних проблем є складність аналізу розріджених і різнорідних даних, оскільки взаємодії користувачів часто обмежуються лише кількома об'єктами з усього каталогу. Це створює виклики для ефективної побудови рекомендацій. Для вирішення цієї проблеми використовуються технології

оптимізації баз даних, розподілені обчислення та багаторівневе кешування, що дозволяє обробляти запити в реальному часі.

Крім того, складність даних підвищує вимоги до систем моніторингу. Постійне відстеження часу виконання запитів і аналіз використання ресурсів допомагає виявляти вузькі місця та забезпечувати безперебійну роботу системи навіть під час пікових навантажень.

Масштабованість архітектури бази даних

Один із ключових аспектів обробки великих обсягів даних — це масштабованість бази даних. Система повинна бути здатна обробляти дані, кількість яких постійно зростає через збільшення кількості користувачів, об'єктів нерухомості та взаємодій. Для досягнення масштабованості застосовуються такі методи:

- горизонтальне масштабування: Поділ бази даних на кілька частин (шардів) за географічними регіонами або іншими критеріями. Наприклад, таблиця apart може бути розділена на частини, де кожна частина містить дані про об'єкти певного міста;
- реплікація: Копіювання бази даних на кілька серверів для забезпечення високої доступності та балансу навантаження. Реплікація дозволяє одночасно обробляти більше запитів, що знижує ризик перевантаження одного сервера;
- індексування: Використання індексів для прискорення запитів, зокрема тих, які стосуються пошуку та сортування великих масивів даних. У системі використовуються індекси для полів user_id, apart_id, action тощо, що оптимізує обробку запитів.

Кешування даних

Кешування є важливим механізмом для зменшення навантаження на базу даних і сервер. Воно дозволяє зберігати результати запитів у тимчасовій пам'яті, що значно скорочує час відповіді системи на повторювані запити. Наприклад:

- кешування рекомендацій: Результати роботи алгоритмів рекомендацій можуть зберігатися у кеші для швидкого доступу під час повторних запитів від користувачів;
- кешування популярних об'єктів: Об'єкти нерухомості, які найчастіше переглядаються, можуть бути збережені у кеші для прискорення їх відображення;
- рівні кешування: Кешування може здійснюватися як на рівні сервера бази даних, так і на рівні веб-сервера (наприклад, за допомогою Redis або

Обробка потокових даних

У системах з високою частотою взаємодій, таких як перегляди та кліки, обробка потокових даних є критично важливою. Для цього використовуються наступні підходи:

- асинхронна обробка: Запити, що не потребують негайного повернення результатів (наприклад, збереження інформації про перегляди), виконуються у фоновому режимі;
- паралельна обробка: Використання багатопоточності або розподілених обчислень для обробки даних у реальному часі;
- буферизація: Дані спочатку зберігаються у тимчасових буферах, а потім пакетно записуються в основну базу даних. Це знижує кількість записів у режимі реального часу та підвищує продуктивність системи.

Оптимізація запитів

Ефективна обробка великих обсягів даних неможлива без оптимізації SQL-запитів. У проекті застосовуються такі методи оптимізації:

- обмеження обсягу даних: Використання конструкції LIMIT для скорочення кількості рядків, які обробляються в одному запиті;
- уникнення складних операцій: Мінімізація використання важких обчислювальних операцій, таких як JOIN між великими таблицями;
- попередньо обчислені дані: Результати складних обчислень зберігаються в окремих таблицях, що дозволяє уникнути повторних обчислень.

Використання логів і моніторингу

Моніторинг роботи системи та аналіз логів дозволяють своєчасно виявляти вузькі місця та оптимізувати їх. Основні аспекти моніторингу включають:

- відстеження часу виконання запитів: Ідентифікація запитів, які займають найбільше часу;
- аналіз навантаження на сервери: Контроль використання ресурсів, таких як CPU, пам'ять і дисковий ввід/вивід;
- виявлення дублюючих запитів: Оптимізація запитів, які виконуються повторно без змін у вихідних даних.

Інтеграція алгоритмів машинного навчання

Для обробки великих обсягів даних алгоритми машинного навчання, такі як колаборативна та контентна фільтрація, повинні бути інтегровані з урахуванням масштабованості. Основні аспекти інтеграції включають:

- паралельне виконання алгоритмів: Використання розподілених обчислювальних платформ для роботи з великими наборами даних;
- модульність: Кожен алгоритм реалізований як незалежний модуль, що дозволяє гнучко змінювати конфігурацію системи;
- зменшення розрідженості даних: Використання матричних факторизацій для роботи з рідкими даними.

Обробка великих обсягів даних у системі рекомендацій вимагає поєднання різних підходів і технологій, які забезпечують ефективність, масштабованість і стабільність роботи системи. Завдяки впровадженню методів оптимізації, кешування, паралельної обробки та інтеграції сучасних алгоритмів машинного навчання система може ефективно працювати навіть за умов інтенсивного використання та великого обсягу даних. У подальшому планується розширення функціональності за рахунок більш глибокої інтеграції потокових обробок і вдосконалення моніторингу продуктивності.

РОЗДІЛ 4
ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

4.1 Постановка експерименту для оцінки якості рекомендацій

Експериментальні дослідження є важливим етапом розробки системи рекомендацій, оскільки вони дозволяють оцінити ефективність, точність і продуктивність запропонованих алгоритмів у реальних умовах. Головна мета таких досліджень — забезпечити перевірку практичної працездатності методів персоналізації контенту та визначити найкращі підходи до рекомендацій для користувачів. Результати експериментів надають важливу інформацію для подальшого вдосконалення алгоритмів, оптимізації продуктивності та підвищення якості користувацького досвіду.

У цьому розділі викладено ключові аспекти експериментальної методики, включаючи підготовку тестових даних, вибір метрик для оцінки якості рекомендацій і аналіз отриманих результатів. Також розглянуто теоретичну основу обраних підходів і їхню практичну реалізацію. Особливу увагу приділено порівнянню контентної, колаборативної фільтрації та гібридної моделі, що забезпечує поєднання їхніх сильних сторін. Це дозволяє зробити висновки про ефективність кожного методу та визначити напрямки для майбутніх покращень.

Мета експерименту

Основна мета експерименту полягає в тому, щоб перевірити здатність системи надавати релевантні та персоналізовані рекомендації користувачам на основі їхньої історії взаємодій і характеристик контенту. Такий підхід дозволяє не лише оцінити точність і продуктивність запропонованих алгоритмів, а й виявити потенційні обмеження системи. Вивчення сильних і слабких сторін кожного методу, зокрема контентної фільтрації, колаборативної фільтрації та гібридного підходу, є ключовим для вдосконалення якості рекомендацій.

Експеримент також спрямований на аналіз адаптивності системи до різних сценаріїв використання. Наприклад, важливо зрозуміти, як система працює з

новими користувачами або об'єктами, для яких обмежена кількість історичних даних. Це стосується вирішення проблеми "холодного старту", яка є типовою для багатьох систем рекомендацій. Крім того, експеримент розглядає, як різні типи взаємодій (перегляди, кліки, додавання до обраного) впливають на якість рекомендацій і які аспекти потребують оптимізації.

Особливу увагу приділено оцінці швидкості обробки даних у режимі реального часу, оскільки продуктивність системи є критично важливою для користувачів. Затримки у формуванні рекомендацій можуть значно знизити залученість користувачів до платформи, тому в рамках експерименту аналізується час відповіді кожного алгоритму. Крім того, дослідження охоплює способи зниження обчислювальної складності системи, що стає важливим у контексті роботи з великими обсягами даних.

Опис експериментальної методики

Для проведення експерименту спочатку були підготовлені дані. Синтетичний набір даних включає 100 об'єктів нерухомості та 500 взаємодій користувачів. Взаємодії включають перегляди, кліки та додавання до обраного, що відповідає різним рівням зацікавленості користувачів. Метадані об'єктів містять такі атрибути, як місто, кількість кімнат, площа, ціна та тип об'єкта. Ці дані моделюють реальний сценарій роботи платформи, де об'єкти та взаємодії є неоднорідними.

Реалізація алгоритмів здійснювалася за допомогою нативного PHP із використанням MySQL для зберігання даних. Контентна фільтрація базувалася на порівнянні характеристик об'єктів, тоді як колаборативна фільтрація використовувала матричний аналіз взаємодій користувачів. Гібридна модель поєднувала обидва підходи шляхом комбінування результатів із використанням вагових коефіцієнтів, які налаштовувалися під час тестування.

Тестування здійснювалося шляхом генерації рекомендацій для кожного користувача на основі його історії взаємодій. Для оцінки точності та релевантності рекомендацій використовувалися стандартні метрики, описані нижче.

Метрики оцінки

Для оцінки якості рекомендацій було обрано кілька ключових метрик, які дозволяють комплексно оцінити продуктивність алгоритмів і релевантність запропонованих об'єктів. Precision є основним показником, який визначає частку релевантних об'єктів серед усіх рекомендованих. Ця метрика особливо корисна, коли потрібно уникнути рекомендації нерелевантних об'єктів, що може знижувати довіру користувачів до системи.

вимірює здатність системи знаходити всі релевантні об'єкти для користувача. Ця метрика важлива для оцінки повноти рекомендацій, оскільки вона показує, наскільки ефективно система враховує всі можливі варіанти, що можуть зацікавити користувача. Однак високе значення Recall може супроводжуватися зниженням Precision, якщо в список рекомендацій додається багато нерелевантних об'єктів.

F1-міра є гармонійним середнім між Precision і Recall, що забезпечує збалансовану оцінку. Ця метрика враховує як точність, так і повноту, дозволяючи уникнути упередженості до однієї з них. Mean Average Precision (MAP) використовується для розрахунку середньої точності на різних рівнях релевантності. Вона забезпечує більш детальний аналіз якості рекомендацій, зокрема, в сценаріях, де важлива послідовність і ранжування запропонованих об'єктів.

Ці метрики дозволяють об'єктивно оцінити роботу системи в різних сценаріях, включаючи нових користувачів, популярні об'єкти або ситуації з обмеженими даними про взаємодії. Вибір таких метрик дає можливість виявити слабкі сторони алгоритмів і внести необхідні корективи для підвищення якості рекомендацій.

4.2 Результати експерименту

Результати експерименту підтвердили ефективність розроблених алгоритмів у системі рекомендацій для платформи оренди нерухомості. Було

виявлено, що запропоновані методи забезпечують високий рівень персоналізації та задовольняють основні вимоги користувачів щодо точності та швидкості роботи.

У ході тестування було оцінено три основні алгоритми — контентну фільтрацію, колаборативну фільтрацію та гібридний підхід. Контентна фільтрація показала себе як ефективний інструмент для роботи з метаданими об'єктів, однак мала обмеження при недостатній кількості даних про взаємодії користувачів. Колаборативна фільтрація, навпаки, демонструвала кращі результати для користувачів із багатою історією взаємодій, але була вразлива до розрідженості даних.

Гібридна модель, яка поєднує елементи обох підходів, виявилася найбільш ефективною, забезпечуючи як високу точність рекомендацій, так і гнучкість у роботі з різними типами даних. Зокрема, цей підхід дозволив ефективно вирішити проблему холодного старту для нових користувачів і об'єктів, що є важливою перевагою в контексті масштабування системи. Детальний аналіз продуктивності та точності алгоритмів наведено в наступних підрозділах.

Загальні результати

Було проведено тестування трьох основних алгоритмів — контентної фільтрації, колаборативної фільтрації та гібридного підходу. Для цього використовувався підготовлений синтетичний набір даних, який включає 100 об'єктів нерухомості, представлених різними типами (квартири, будинки, офісні приміщення), а також 500 взаємодій користувачів, таких як перегляди, кліки та додавання до обраного. Дані були ретельно структуровані, щоб забезпечити сценарії, які максимально наближені до реального використання.

Для кожного алгоритму було визначено різні сценарії взаємодії, зокрема користувачів із великою історією дій, нових користувачів без жодної взаємодії та об'єктів, які тільки додані до бази даних. Такі сценарії дозволили оцінити здатність системи ефективно адаптуватися до різних умов і підтримувати стабільну продуктивність незалежно від обставин. Крім того, враховувалися

особливості розріджених даних, де значна частина об'єктів і взаємодій залишалися нерелевантними для більшості користувачів.

Тестування проводилося за декількома етапами: на першому — оцінювалася базова продуктивність алгоритмів на невеликій вибірці, а на наступних — аналізувалася масштабованість і вплив збільшення обсягу даних на швидкість і точність рекомендацій. Це дозволило отримати повне уявлення про сильні та слабкі сторони кожного методу в контексті платформи оренди нерухомості.

Для оцінки якості рекомендацій було використано такі метрики, як Precision, Recall, F1-міра та Mean Average Precision (MAP). Ці метрики забезпечують багатогранний підхід до аналізу ефективності системи рекомендацій, дозволяючи оцінити не лише точність запропонованих рекомендацій, але й їхню повноту та релевантність для користувачів.

Precision є важливим показником, який вимірює частку рекомендованих об'єктів, що дійсно відповідають вподобанням користувача. Висока точність означає, що система здатна уникати нерелевантних рекомендацій, забезпечуючи користувача лише тим контентом, який є корисним. Recall, своєю чергою, оцінює, наскільки система здатна знаходити всі релевантні об'єкти в базі даних. Ця метрика особливо важлива для виявлення ефективності системи у забезпеченні користувача повним набором релевантних результатів.

F1-міра дозволяє збалансувати Precision і Recall, створюючи єдиний показник для загальної оцінки продуктивності. Mean Average Precision (MAP) розглядає середню точність на різних рівнях релевантності, враховуючи також ранжування результатів. Ця метрика є ключовою в контексті рекомендаційних систем, де порядок результатів відіграє важливу роль для користувача.

Таблиця 4.1 – Результати тестування алгоритмів

Алгоритм	**Precision**	**Recall**	**F1-міра**	**MAP**
Контентна фільтрація	0.78	0.65	0.71	0.74
Колаборативна фільтрація	0.80	0.68	0.73	0.77

Продовження таблиці 4.1

Гібридна модель	0.82	0.75	0.78	0.81

Гібридна модель продемонструвала найкращі результати за всіма метриками, що підтверджує її переваги у поєднанні підходів контентної та колаборативної фільтрації. Поєднання цих двох методів дозволяє враховувати як характеристичні атрибути об'єктів, так і поведінкові особливості користувачів, що забезпечує високу точність рекомендацій навіть у складних сценаріях.

Наприклад, якщо користувач переглядав об'єкти у певному ціновому діапазоні та розташуванні, але не залишав активних взаємодій (кліків або додавання до обраного), модель здатна врахувати ці аспекти через контентну складову. Одночасно колаборативна частина моделі додає соціальний контекст, аналізуючи схожих користувачів і їхні дії.

Крім того, гібридна модель виявилася надзвичайно корисною для вирішення проблеми "холодного старту". Завдяки інтеграції контентної фільтрації, система змогла надавати релевантні рекомендації навіть для нових користувачів без історії взаємодій. Це особливо важливо для масштабованих платформ, де велика частина аудиторії є новими користувачами.

З іншого боку, для досвідчених користувачів із багатою історією дій колаборативна складова дозволяє збагачувати рекомендації за рахунок аналізу спільної поведінки з іншими користувачами, що робить результати більш точними та персоналізованими.

Гібридна модель також продемонструвала стійкість до розрідженості даних, яка є типовою та проблемою для колаборативної фільтрації. Наприклад, у випадках, коли лише невелика частина користувачів взаємодіяла з певними об'єктами, модель все одно могла генерувати рекомендації, використовуючи дані про характеристики об'єктів. Це забезпечує ширше покриття рекомендацій і зменшує ризик втрати релевантних результатів через нестачу даних про взаємодії. Більше того, завдяки комбінуванню двох підходів, модель змогла

враховувати як індивідуальні вподобання користувачів, так і загальні тенденції серед усієї аудиторії, підвищуючи точність рекомендацій.

Аналіз продуктивності

Крім точності рекомендацій, оцінювалася продуктивність системи з точки зору часу відповіді для кожного алгоритму. Результати наведені нижче: було проведено серію тестів на різних наборах даних, які імітують реальні умови використання платформи.

Таблиця 4.2 – Результати продуктивності алгоритмів фільтрації

Алгоритм	**Час відповіді (мс)**
Контентна фільтрація	100
Колаборативна фільтрація	150
Гібридна модель	200

Контентна фільтрація забезпечує найшвидший час відповіді завдяки простоті розрахунків, оскільки основна робота алгоритму полягає в аналізі характеристик об'єктів. Наприклад, обчислення схожості між об'єктами на основі таких атрибутів, як ціна, площа чи кількість кімнат, виконується швидко, навіть для великих наборів даних. Однак цей підхід демонструє обмеження в ситуаціях, коли користувачі не залишають достатньої кількості даних про свої вподобання або коли нові об'єкти не мають подібних характеристик до вже існуючих. Це може призводити до менш точних рекомендацій для користувачів із малою активністю або унікальними вподобаннями.

Колаборативна фільтрація, хоча й вимагає більше ресурсів для обчислення, демонструє більшу гнучкість у генерації рекомендацій. Вона враховує поведінкові взаємодії користувачів, такі як перегляди, кліки та додавання до обраного. Це дозволяє створювати рекомендації на основі схожості між користувачами чи об'єктами. Наприклад, якщо кілька користувачів із подібними уподобаннями переглядали однакові об'єкти, система може запропонувати ці об'єкти новому користувачеві з аналогічною поведінкою. Проте, цей підхід вразливий до розрідженості даних, коли велика частина об'єктів або взаємодій є

малорепрезентативними, що може ускладнювати формування якісних рекомендацій.

Гібридна модель, яка об'єднує обидва підходи, хоч і має найвищий час відповіді (~200 мс), демонструє значну перевагу в точності та універсальності. Вона враховує як контентні характеристики об'єктів, так і поведінкові особливості користувачів, що дозволяє компенсувати обмеження кожного окремого підходу. Наприклад, у випадках, коли нові користувачі ще не мають історії взаємодій, модель може використовувати контентну фільтрацію для створення базових рекомендацій, доповнюючи їх результатами колаборативного аналізу, щойно накопичиться достатня кількість поведінкових даних. Це забезпечує збалансованість між швидкістю обчислення та якістю рекомендацій навіть у складних сценаріях.

Гібридна модель, хоча і демонструє найкращі результати за метриками точності та повноти, потребує додаткових ресурсів для обчислення, що є важливим аспектом при масштабуванні системи. Висока обчислювальна складність може створювати навантаження на сервери під час пікових періодів, особливо коли кількість користувачів платформи значно зростає. У таких випадках необхідно впроваджувати стратегії оптимізації, наприклад, кешування рекомендацій для активних користувачів або використання розподілених обчислень.

Крім того, під час тестування було виявлено, що контентна фільтрація є більш ефективною для нових об'єктів нерухомості, оскільки вона використовує виключно атрибути об'єктів, а не залежить від історичних взаємодій користувачів. Це дозволяє швидко додавати нові об'єкти до системи без втрати їхньої видимості для користувачів. Однак її обмеження стають очевидними, коли дані про об'єкти є недостатньо деталізованими або неповними, що може призводити до зниження релевантності рекомендацій.

Ще однією важливою особливістю є здатність колаборативної фільтрації враховувати зміни у вподобаннях користувачів. Наприклад, якщо користувач починає взаємодіяти з новою категорією об'єктів, система автоматично

адаптується до цих змін, ґрунтуючись на спільній поведінці інших користувачів. Це робить колаборативну фільтрацію ефективною для динамічних платформ, де уподобання користувачів часто змінюються.

Водночас розрідженість даних залишається проблемою для колаборативної фільтрації. У великих системах з великими базами даних об'єктів і користувачів, де багато записів не мають перехресних взаємодій, алгоритм може виявитися неефективним. Гібридна модель допомагає частково вирішити цю проблему, об'єднуючи атрибути об'єктів і поведінкові дані, що дозволяє створювати рекомендації навіть для рідко взаємодіючих об'єктів.

Ще одним важливим спостереженням стало те, що час відповіді системи значно залежить від обсягу оброблюваних даних. Наприклад, для активних користувачів із великою історією взаємодій генерація рекомендацій може потребувати більше часу, ніж для нових користувачів. Це підтверджує необхідність оптимізації процесу обчислень, особливо для масштабованих платформ, які обслуговують тисячі або навіть мільйони користувачів.

Експериментальні дослідження показали, що запропонована система рекомендацій ефективно персоналізує контент і адаптується до різних сценаріїв використання. Гібридна модель є оптимальним рішенням для досягнення високої точності рекомендацій при прийнятному рівні продуктивності. Подальші дослідження можуть бути спрямовані на оптимізацію часу відповіді гібридної моделі та розробку методів для вирішення проблеми холодного старту.

ВИСНОВКИ

У ході виконання дипломної роботи було розроблено ефективну систему рекомендацій для персоналізації контенту на веб-сайті оренди нерухомості. Основною метою роботи було створення адаптивної платформи, здатної враховувати індивідуальні вподобання користувачів, забезпечувати релевантність рекомендацій та покращувати загальний користувацький досвід. Досягнення цієї мети стало можливим завдяки комплексному підходу до проєктування, реалізації та тестування системи.

Проведений аналіз предметної області та існуючих рішень дозволив визначити ключові переваги та недоліки сучасних систем рекомендацій. Зокрема, було детально розглянуто методи контентної, колаборативної фільтрації та їх гібридних моделей. Ці знання стали основою для вибору оптимальної стратегії реалізації, яка забезпечила високу точність рекомендацій.

Розробка архітектури системи базувалася на використанні Model-View-Controller (MVC), що дозволило чітко розмежувати бізнес-логіку, обробку запитів і відображення даних. Такий підхід сприяв зручності підтримки та масштабованості платформи. Усі компоненти системи були інтегровані для забезпечення стабільної та ефективної роботи.

Важливою частиною роботи стала реалізація алгоритмів рекомендацій. Гібридна модель, яка поєднує елементи контентної і колаборативної фільтрації, дозволила врахувати як характеристики об'єктів нерухомості, так і поведінкові особливості користувачів. Це забезпечило релевантність рекомендацій навіть у складних сценаріях, таких як проблема "холодного старту".

Розробка функціональних компонентів сайту включала створення ключових сторінок, таких як Home, Favorites, Recommended і Profile. Було реалізовано модулі для реєстрації та входу користувачів, а також інтерактивний інтерфейс для вибору фільтрів. Особливу увагу приділено забезпеченню зручності та інтуїтивності інтерфейсу, що значно покращило користувацький досвід.

На етапі тестування проведено функціональне, адаптивне та навантажувальне тестування системи. Результати показали, що сайт стабільно працює під великим навантаженням, а алгоритми рекомендацій забезпечують високу точність і продуктивність. Ці аспекти є важливими для подальшого впровадження системи у реальні умови.

Розроблена система рекомендацій має як теоретичну, так і практичну цінність. З наукової точки зору, робота продемонструвала ефективність використання гібридних підходів для персоналізації контенту в контексті платформи оренди нерухомості. Практична цінність полягає у створенні масштабованої та легко адаптованої системи, яка може бути впроваджена в реальні бізнес-процеси.

Результати роботи відкривають перспективи для подальшого вдосконалення системи. Напрямки розвитку включають оптимізацію продуктивності алгоритмів для роботи з великими обсягами даних, розширення функціоналу системи шляхом інтеграції з картографічними сервісами, використання додаткових джерел даних, таких як соціальні мережі, та впровадження складніших моделей машинного навчання для передбачення потреб користувачів.

Таким чином, дипломна робота довела доцільність і ефективність використання сучасних технологій для створення персоналізованих систем рекомендацій. Запропоновані методи та реалізація дозволили досягти високої точності рекомендацій, зручності інтерфейсу та адаптивності системи до потреб користувачів. Це забезпечує платформі конкурентні переваги та підвищує її потенціал для подальшого розвитку та впровадження у реальні умови.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

https://www.sciencedirect.com/science/article/pii/S0306437924000255 (дата звернення: 10.01.2025).

Commerce. – 2000. – С. 158–167.

ollaborative Filtering in Recommender System: An Overview. URL: https://medium.com/@evelyn.eve.9512/collaborative-filtering-in-recommender-system-an-overview-38dfa8462b61 (дата звернення: 10.01.2025).

up with consumers. URL: https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers (дата звернення: 10.01.2025).

itpanda973/how-to-create-and-filter-records-with-php-and-mysql-from-multiple-table-31dc04ea9da9 (дата звернення: 10.01.2025).

feaecd92608 (дата звернення: 10.01.2025).

. Martinez, M. J. Barranco, L. G. Perez, and M. Espinilla. A knowledge based recommender system with multigranular linguistic information. // International Journal of Computational Intelligence Systems. – 2008. – Т. 1, № 3. – С. 225–236.

. Kaminskas and F. Ricci. Location-adapted music recommendation using tags. // International Conference on User Modeling, Adaptation, and Personalization. – Springer, 2011. – С. 183–194.

ion Engines. URL: https://insights.daffodilsw.com/blog/machine-learning-algorithms-for-recommendation-engines (дата звернення: 10.01.2025).

al-Intelligence (дата звернення: 10.01.2025).

ypes of Hybrid Recommendation System. URL: https://medium.com/analytics-vidhya/7-types-of-hybrid-recommendation-system-3e4f78266ad8 (дата звернення:

ization Techniques for Recommender Systems. // IEEE Computer. – 2009.

ггарвал К. Системи рекомендацій: підручник. – Лондон: Springer, 2016. – 494 с.

жаннач Д., Занкер М., Фельферніг А. Вступ до систем рекомендацій. – Кембридж: 2011. – 354 с.

іккі Ф., Рокач Л., Шапіра Б. Системи рекомендацій: довідник. – Нью-Йорк: Springer, 2015. – 845 с.

ДОДАТОК А
ВИХІДНИЙ КОД ОСНОВНИХ КОМПОНЕНТІВ

1. Вхідний компонент index

```
<?php
session_start();
use Dotenv\Dotenv;

require_once dirname(__DIR__) . "/incs/config.php";
require_once INCS . "/Router.php";
//require_once INCS . "/data.php";
require_once INCS . "/funcs.php";
require_once INCS . "/DB.php";
require_once ROOT . "/vendor/autoload.php";


$dotenv = Dotenv::createImmutable(ROOT);
$dotenv->load();

$dsn = "mysql:host=localhost;dbname={$_ENV['DB_NAME']};charset=utf8mb4";
$db = DB::create($dsn, $_ENV['DB_USER'], $_ENV['DB_PASSWORD']);


$router = Router::create(CONTROLLERS);
try {
	$router->get("/", "index.php");
	$router->post("/", "indexPost.php");
	$router->get("/login", "login.php");
	$router->get("/register", "register.php");
	$router->get("error_page", "error.php");
	$router->post("/register", "registerPost.php");
	$router->post("/login", "loginPost.php");
	$router->get("/logout", "logout.php");
	$router->get("/profile", "profile.php");
	$router->post("/profile", "profilePost.php");
	$router->get("/apart/*", "apart.php");
	$router->get("/favorites", "favorites.php");
	$router->post("/favorites", "favoritesPost.php");
	$router->get("/recommended", "recommended.php");
} catch (Exception $e) {
	die("can not register new router method: \"{$e->getMessage()}\"");
}

$router->match();
```

2. Класс DB

```
<?php


class DB
{
	protected static DB $instance;
	protected int $total_ad;
	protected string $main_query;
	public PDO $db;
```

```
	public function __construct()
	{
		$this->main_query = "
		 SELECT
		     a.id AS apart_id,
		     a.title,
		     a.city,
		     a.type,
		     a.price,
		     a.description,
		     a.rooms,
		     a.area,
		     a.created_at,
		     GROUP_CONCAT(ai.image_path ORDER BY ai.is_main DESC SEPARATOR ',') AS
images
		 FROM apart a
		 LEFT JOIN apart_images ai ON a.id = ai.apart_id
	    ";
	}
	public static function create(string $dsn, string $username, string $password):
self
	{
		if (!isset(self::$instance)) {
			self::$instance = new self();
		}
		self::$instance->db = new PDO($dsn, $username, $password,
[PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]);
		return self::$instance;
	}
	public function query(string $query_str): array
	{
		return $this->db->query($query_str)->fetchAll(PDO::FETCH_ASSOC);
	}
	public function register(array $data): void
	{
		try {
			$stmt = $this->db->prepare("insert into users (username, email,
password, profile_pic) values (:username, :email, :password, :avatar)");
			$stmt->execute($this->prepare_register_data($data));
		} catch (PDOException $e) {
			throw $e;
		}
	}
	public function login(array $data): array | false
	{
		try {
			$stmt = $this->db->prepare("select id, username, email, password,
profile_pic from users where email = ?");
			$stmt->execute([$data['email']]);
		} catch (PDOException $e) {
			throw $e;
		}
		return $stmt->fetch(PDO::FETCH_ASSOC);
	}
	public function delete_user(string $email): bool
	{
		$stmt = $this->db->prepare("delete from users where email = ?");
		return $stmt->execute([$email]);
	}
	public function get_password(string $email): string
	{
```

```
            $stmt = $this->db->prepare("select password from users where email = ?");
            $stmt->execute([$email]);
            return $stmt->fetchColumn();
      }
      public function set_new_password(string $email, string $password): bool
      {
            $stmt = $this->db->prepare("update users set password = :password where
email = :email");
            return $stmt->execute(['email' => $email, 'password' => $password]);
      }
      public function update_profile_pic(string $email, string $pic_path): bool
      {
            $stmt = $this->db->prepare("update users set profile_pic = ? where email =
?");
            return $stmt->execute([$pic_path, $email]);
      }
      protected function prepare_register_data(array $data): array
      {
            $result['username'] = h($data['username']);
            $result['email'] = h($data['email']);
            $result['password'] = password_hash($data['password'], PASSWORD_DEFAULT);
            $result['avatar'] = $data['avatar'];
            return $result;
      }



      public function get_ads(int $page = 1, int $limit = 10): array
      {
            // Compute the offset for pagination
            $offset = ($page - 1) * $limit;

            // Start building the base SQL query
            $query = "
       SELECT
           a.id AS apart_id,
           a.title,
           a.city,
           a.type,
           a.price,
           a.description,
           a.rooms,
           a.area,
           a.created_at,
           GROUP_CONCAT(ai.image_path ORDER BY ai.is_main DESC SEPARATOR ',') AS
images
       FROM apart a
       LEFT JOIN apart_images ai ON a.id = ai.apart_id
   ";

            // Prepare the WHERE conditions dynamically
            $conditions = [];

            if (!empty($_SESSION['applied_filter'])) {
                  foreach ($_SESSION['applied_filter'] as $key => $values) {
                        $conditions[$key] = $values;
                  }
            }

            // Add the WHERE clause if conditions exist
            if (!empty($conditions)) {
```

```
                $conditions_query = 'WHERE';
                foreach($conditions as $key => $value) {
                        $conditions_query .= '(';
                        foreach ($value as $id => $item) {

                                if ($key == 'area-min') {
                                        $conditions_query .= " $key >= :$key$id";
                                } else if ($key == 'price') {
                                        $conditions_query .= " $key <= :$key$id";
                                } else {
                                        $conditions_query .= " $key = :$key$id";
                                }

                                if ($id < count($value) - 1) {
                                        $conditions_query .= " OR";
                                }
                        }
                        $conditions_query .= ") AND";
                }
                $query .= trim($conditions_query, ' AND');
        }

//          error_log("SQL: $query", 3, './../errors.log');

          // Add grouping, ordering, and pagination
          $query .= "
        GROUP BY a.id
        ORDER BY a.id
    ";


          $stmt_total = $this->db->prepare($query);

          // Prepare the statement
          $query .= "LIMIT :limit OFFSET :offset";
          $stmt = $this->db->prepare($query);

          // Bind the dynamic parameters
          if ( !empty($conditions) ) {
                foreach ($conditions as $key => $value) {
                        foreach ($value as $id => $item) {
                                $stmt->bindValue(":$key$id", $item);
                                $stmt_total->bindValue(":$key$id", $item);
                        }
                }
          }


          // Calc total
          $stmt_total->execute();
          $this->total_ad = count($stmt_total->fetchAll());




          // Bind pagination parameters
          $stmt->bindValue(':limit', $limit, PDO::PARAM_INT);
          $stmt->bindValue(':offset', $offset, PDO::PARAM_INT);

          // Execute the query
```

```
		$stmt->execute();

		// Fetch the results
		$apartments = [];
		while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
			$images = $row['images'] ? explode(',', $row['images']) : [];
			$apartments[] = [
				'id' => $row['apart_id'],
				'title' => $row['title'],
				'city' => $row['city'],
				'type' => $row['type'],
				'price' => $row['price'],
				'description' => $row['description'],
				'rooms' => $row['rooms'],
				'area' => $row['area'],
				'created_at' => $row['created_at'],
				'images' => $images,
			];
		}

		return $apartments;
	}

	protected function format_ads(PDOStatement $stmt): array
	{
		$apartments = [];
		while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
			$images = $row['images'] ? explode(',', $row['images']) : [];
			$apartments[] = [
				'id' => $row['apart_id'],
				'title' => $row['title'],
				'city' => $row['city'],
				'type' => $row['type'],
				'price' => $row['price'],
				'description' => $row['description'],
				'rooms' => $row['rooms'],
				'area' => $row['area'],
				'created_at' => $row['created_at'],
				'images' => $images,
			];
		}

		return $apartments;
	}

	public function get_certain_ad(int $id, int $user_id = -1): array
	{
		$query = $this->main_query . " WHERE a.id = :id" . " GROUP BY a.id ORDER
BY a.id ";
		$stmt = $this->db->prepare($query);
		if ($stmt === false) {
			throw new Exception("could not prepare");
		}

		$stmt->execute([':id' => $id]);

		// interactions
		if ($user_id != -1) {
			$interaction_stmt = $this->db->prepare("
				INSERT INTO interactions (user_id, apart_id, action)
				VALUES (:user_id, :apart_id, 'click')
```

```
                    ");
                    $res = $interaction_stmt->execute([':user_id' => $user_id,
':apart_id' => $id]);
                    if (!$res) {
                          throw new Exception("Failed to log click interaction.");
                    }
             }

             return $this->format_ads($stmt)[0];
      }

      public function set_fav_ad(int $user_id, int $apart_id): bool
      {
             $stmt = $this->db->prepare("insert into favorites (user_id, apart_id)
values (?, ?)");

             // add interaction
             $interaction_stmt = $this->db->prepare("
             INSERT INTO interactions (user_id, apart_id, action, timestamp)
             VALUES (:user_id, :apart_id, 'favorite', NOW())
             ON DUPLICATE KEY UPDATE timestamp = NOW()
             ");

             $interaction_stmt->execute([':user_id' => $user_id, ':apart_id' =>
$apart_id]);

             return $stmt->execute([$user_id, $apart_id]);
      }

      public function delete_fav_ad(int $user_id, int $apart_id): bool
      {
             $stmt = $this->db->prepare("delete from favorites where user_id = ? and
apart_id = ?");

             //delete interaction
             $interaction_stmt = $this->db->prepare("
             INSERT INTO interactions (user_id, apart_id, action)
        VALUES (:user_id, :apart_id, 'unfavorite')
        ");

             $interaction_stmt->execute([':user_id' => $user_id, ':apart_id' =>
$apart_id]);

             return $stmt->execute([$user_id, $apart_id]);
      }

      public function get_fav_ad($user_id): array
      {
             $stmt = $this->db->prepare("select * from favorites where user_id = ?");
             $stmt->execute([$user_id]);
             return $stmt->fetchAll(PDO::FETCH_ASSOC);
      }

      public function getFavorites(int $userId): array
      {
             $query = "
        SELECT
            a.id AS apart_id,
            a.title,
            a.city,
            a.type,
```

```
            a.price,
            a.description,
            a.rooms,
            a.area,
            a.created_at,
            GROUP_CONCAT(ai.image_path ORDER BY ai.is_main DESC SEPARATOR ',') AS
images
        FROM (
            SELECT
                f.apart_id,
                MAX(f.added_at) AS added_at
            FROM favorites f
            WHERE f.user_id = :user_id
            GROUP BY f.apart_id
        ) AS filtered_favorites
        INNER JOIN apart a ON filtered_favorites.apart_id = a.id
        LEFT JOIN apart_images ai ON a.id = ai.apart_id
        GROUP BY a.id
        ORDER BY filtered_favorites.added_at DESC
    ";

            $stmt = $this->db->prepare($query);
            $stmt->bindValue(':user_id', $userId, PDO::PARAM_INT);
            $stmt->execute();

            $favorites = [];

            while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
                $images = $row['images'] ? explode(',', $row['images']) : [];

                $favorites[] = [
                    'id' => $row['apart_id'],
                    'title' => $row['title'],
                    'city' => $row['city'],
                    'type' => $row['type'],
                    'price' => $row['price'],
                    'description' => $row['description'],
                    'rooms' => $row['rooms'],
                    'area' => $row['area'],
                    'created_at' => $row['created_at'],
                    'images' => $images
                ];
            }

            return $favorites;
        }

        public function get_all_locations(): array
        {
            $stmt = $this->db->prepare("select distinct city from apart");
            $stmt->execute();
            return $stmt->fetchAll(PDO::FETCH_COLUMN);
        }

        public function get_ads_num(): int
        {
            return $this->total_ad;
        }

        public function set_interactions_view(array $ads_list, int $user_id): void
        {
```

```
            if (empty($ads_list)) {
                  return;
            }

            $placeholders = [];
            $params = [];
            foreach ($ads_list as $index => $ad_id) {
                  $placeholders[] = "(:user_id, :apart_id_{$index}, 'view')";
                  $params[":apart_id_{$index}"] = $ad_id;
            }

            $query = "
            INSERT INTO interactions (user_id, apart_id, action)
            VALUES " . implode(", ", $placeholders) . "
           ON DUPLICATE KEY UPDATE timestamp = NOW()
        ";

            $params[':user_id'] = $user_id;
            $stmt = $this->db->prepare($query);
            $stmt->execute($params);

      }


      public function getRecommendedAds(int $userId, int $limit = 10): array
      {
            $query = "
        SELECT
            a.*,
            GROUP_CONCAT(DISTINCT ai.image_path ORDER BY ai.is_main DESC SEPARATOR ',')
AS images,
            SUM(CASE
                WHEN i.action = 'favorite' THEN 3
                WHEN i.action = 'click' THEN 2
                WHEN i.action = 'view' THEN 1
                ELSE 0 END) AS score
        FROM apart a
        LEFT JOIN apart_images ai ON a.id = ai.apart_id
        LEFT JOIN interactions i ON a.id = i.apart_id
        WHERE a.city IN (
            SELECT DISTINCT a1.city
            FROM apart a1
            INNER JOIN interactions i1 ON a1.id = i1.apart_id
            WHERE i1.user_id = :user_id AND i1.action IN ('favorite', 'click')
        )
        AND a.id NOT IN (
            SELECT apart_id
            FROM interactions
            WHERE user_id = :user_id AND action IN ('favorite', 'unfavorite')
        )
        GROUP BY a.id
        ORDER BY score DESC, a.created_at DESC
        LIMIT :limit;
    ";

            $stmt = $this->db->prepare($query);
            $stmt->bindValue(':user_id', $userId, PDO::PARAM_INT);
            $stmt->bindValue(':limit', $limit, PDO::PARAM_INT);
            $stmt->execute();

            $recommendedAds = [];
```

```
            while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
                  $images = $row['images'] ? explode(',', $row['images']) : [];
                  $recommendedAds[] = [
                        'id' => $row['id'],
                        'title' => $row['title'],
                        'city' => $row['city'],
                        'type' => $row['type'],
                        'price' => $row['price'],
                        'description' => $row['description'],
                        'rooms' => $row['rooms'],
                        'area' => $row['area'],
                        'created_at' => $row['created_at'],
                        'score' => $row['score'],
                        'images' => $images,
                  ];
            }

            return $recommendedAds;
      }



      protected function getDefaultRecommendations(int $limit = 10): array
      {
            $query = "
        SELECT a.*,
               GROUP_CONCAT(DISTINCT ai.image_path ORDER BY ai.is_main DESC SEPARATOR
','') AS images,
               COUNT(i.id) AS interaction_count
        FROM apart a
        LEFT JOIN apart_images ai ON a.id = ai.apart_id
        LEFT JOIN interactions i ON a.id = i.apart_id AND i.action IN ('favorite',
'view', 'click')
        GROUP BY a.id
        ORDER BY interaction_count DESC, a.created_at DESC
        LIMIT :limit;
    ";

            $stmt = $this->db->prepare($query);
            $stmt->bindValue(':limit', $limit, PDO::PARAM_INT);
            $stmt->execute();

            $defaultAds = [];

            while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
                  $images = $row['images'] ? explode(',', $row['images']) : [];
                  $defaultAds[] = [
                        'id' => $row['id'],
                        'title' => $row['title'],
                        'city' => $row['city'],
                        'type' => $row['type'],
                        'price' => $row['price'],
                        'description' => $row['description'],
                        'rooms' => $row['rooms'],
                        'area' => $row['area'],
                        'created_at' => $row['created_at'],
                        'images' => $images,
                  ];
            }
```

```
            return $defaultAds;
        }

        public function get_recommendations(int $user_id, int $limit = 10): array
        {
            $stmt = $this->db->prepare("SELECT COUNT(*) FROM interactions WHERE
            user_id = ?");
            $stmt->execute([$user_id]);
            $user_interactions_counter = $stmt->fetchColumn();


            if ((int)$user_interactions_counter === 0) {
                return $this->getDefaultRecommendations($limit);
            }


            return $this->getRecommendedAds($user_id, $limit);
        }


}
```

3. Клас Router

```
<?php

class Router
{
        protected static Router $instance;
        protected string $error_page_path;
        protected string $base_path;
        public string $query_string;
        protected string $uri;
        protected string $method;
        protected array $routes;
        public function __construct(string $base_path = "")
        {
            $this->uri = strtok($_SERVER['REQUEST_URI'], '?');
            $this->query_string = strtok('?');
            $this->method = $_SERVER['REQUEST_METHOD'];
            $this->base_path = $base_path;
        }
        public static function create(string $base_path = ""): self
        {
            if (!isset(self::$instance)) {
                self::$instance = new Router($base_path);
            }
            return self::$instance;
        }
        public function get(string $uri, string $path, bool $use_base_path = true): void
        {
            ['error' => $error, 'path' => $path] = $this->validate_path($path,
$use_base_path);
            if ($error) {
                throw new Exception("File in path: \"$path\" does not exist");
            }
            if ($uri == 'error_page') {
                $this->error_page_path = $path;
            }
```

```
            $this->routes['GET'][$uri] = "$path";
      }
      public function post(string $uri, string $path, bool $use_base_path = true): void
      {
            ['error' => $error, 'path' => $path] = $this->validate_path($path,
$use_base_path);
            if ($error) {
                  throw new Exception("File in path: \"$path\" does not exist");
            }
            $this->routes['POST'][$uri] = $path;
      }
      protected function validate_path(string $path, bool $use_base_path): array
      {
            $base_path = $use_base_path ? $this->base_path : "";
            $path = trim($path, "/");
            if (!file_exists("$base_path/$path")) {
                  return ['error' => true, 'path' => "$base_path/$path"];
            }
            return ['error' => false, 'path' => "$base_path/$path"];
      }
      public function match(): void
      {
            $uri_data = explode("/", $this->uri);
            $real_uri = "/" . $uri_data[1];
            if (count($uri_data) > 2) {
                  $real_uri.= "/*";
            }

            if (!array_key_exists($real_uri, $this->routes[$this->method])) {
                  http_response_code(404);
                  include $this->error_page_path;
                  exit;
            }
            include $this->routes[$this->method][$real_uri];
      }

}
```

4. Контролер index

```
<?php
global $db;

$page = isset($_GET['page']) ? (int)$_GET['page'] : 1;
$ads_per_page = 10;


$ads_img_folder = '/assets/apartments-pics/';



if (!array_key_exists('applied_filter', $_SESSION)) {
      $_SESSION['applied_filter'] = [];
}

//unset($_SESSION['applied_filter']);


$filters_key = ['rooms', 'area', 'type', 'city', 'price'];
if ($_SERVER['QUERY_STRING']) {
```

```
	foreach ($_GET as $key => $value) {
		if ($value and in_array($key, $filters_key)) {
			if ( !array_key_exists($key, $_SESSION['applied_filter']) ) {
				$_SESSION['applied_filter'][$key] = [];
			}
			if (is_array($value)) {
				foreach ($value as $item) {
					if ( !in_array($item,
$_SESSION['applied_filter'][$key]) ) {
$_SESSION['applied_filter'][$key][] = $item;
					}
				}
				continue;
			}
			if ( !in_array($value, $_SESSION['applied_filter'][$key]) ) {
				$_SESSION['applied_filter'][$key][] = $value;
			}
		}
	}
}

$ads = $db->get_ads($page, $ads_per_page);

$ads_num = $db->get_ads_num();
$total_pages = ceil($ads_num / $ads_per_page);

$locations = $db->get_all_locations();

if (isset($_SESSION['applied_filter']) and $ads and isset($_SESSION['user'])) {
	$views_ads = array_map(function (array $ad_item) {
		return $ad_item['id'];
	}, $ads);
	$db->set_interactions_view($views_ads, $_SESSION['user']['id']);
}


include VIEWS . "/index.tpl.php";
```

5. Контролер index для POST запитів

```
<?php

if ( isset($_POST['clear_filters']) ) {
	unset($_SESSION['applied_filter']);
	redirect('/');
}

$data = json_decode(file_get_contents('php://input'), true);
$data_key = array_key_first($data);
$data_value = array_values($data)[0];



if (isset($_SESSION['applied_filter'][$data_key])) {
	if ( count($_SESSION['applied_filter'][$data_key]) == 1 ) {
		unset($_SESSION['applied_filter'][$data_key]);
	} else {
		$target_key = array_search($data_value,
$_SESSION['applied_filter'][$data_key]);

		if ($target_key !== false) {
```

```
//                  array_splice($_SESSION['applied_filter'][$data_key], 1,
$target_key);
                    $_SESSION['applied_filter'][$data_key] =
array_filter($_SESSION['applied_filter'][$data_key], function ($item) use ($data_value)
{
                           return $item != $data_value;
                    });

//                  error_log("DATA AFTER FILTER: " .
print_r($_SESSION['applied_filter'], true), 3, './../errors.log');
             }
//           error_log("\nDATA: " . print_r($_SESSION['applied_filter'], true), 3,
'./../errors.log');
       }
       echo json_encode(['success' => true]);
}
```

онтролер login

7. Контролер register для POST запитів

```
<?php


global $db;
[$input, $error] = validate_form();
$default_values = $input;

if ($error) {
	require VIEWS . "/register.tpl.php";
} else {
	try {
		$db->register($input);
		$_SESSION['success'] = "User " . h($input['username']) . " was
successfully register";
		redirect("/login");
	} catch (PDOException $e) {
		if ($e->getCode() == 23000) {
			$_SESSION['error'] = "This email is already used";
			require VIEWS . "/register.tpl.php";
		} else {
			http_response_code(401);
			redirect("/error");
		}
	}
}



function validate_form(): array
{
	$input = array();
	$error = array();

	$input['username'] = trim($_POST['username'] ?? '');
	if ($input['username'] == "") {
		$error['username'] = "Username is required";
	}
	$input['email'] = filter_input(INPUT_POST, "email", FILTER_VALIDATE_EMAIL);
	if ($input['email'] == null or $input['email'] === false) {
		$error['email'] = 'Email is required';
	}
	try {
		$avatar_res = avatar_validation($input);
		if (!$avatar_res) {
			$error['avatar'] = 'Profile picture is required';
		}
	} catch (Exception $exception) {
		http_response_code(500);
		require_once VIEWS . "/register.tpl.php";
		exit;
	}
	$input['password'] = trim($_POST['password'] ?? "");
	if ($input['password'] == '' or strlen($input['password']) < 5) {
		$error['password'] = "Password is required and should be at least 5
character";
```

```
	}
	$input['avatar_mode_selector'] = $_POST['avatar_mode_selector'];
	return [$input, $error];
}

function avatar_validation(array &$input): bool
{
	switch ($_POST['avatar_mode_selector']) {
		case 'default_mode':
			if (!isset($_POST['avatar'])) {
				return false;
			}
			$input['avatar'] = "/assets/profile-pics/{$_POST['avatar']}";
			return true;
		case 'custom_mode':
			if ($_FILES['avatar']['error'] === 0) {
				$new_name = uniqid() . "." .
pathinfo($_FILES['avatar']['name'], PATHINFO_EXTENSION);
				$move_to = SITE_ROOT . "/public/assets/profile-
pics/custom_mode/$new_name";
				if (!move_uploaded_file($_FILES['avatar']['tmp_name'],
$move_to)) {
					throw new Exception("can not move file
{$_FILES['avatar']['tmp_name']} to $move_to");
				}
				chmod($move_to, 0666);
				$input['avatar'] = "/assets/profile-
pics/custom_mode/$new_name";
				return true;
			} else if (isset($_POST['prev_avatar'])) {
				$input['avatar'] = $_POST['prev_avatar'];
				return true;
			} else {
				return false;
			}
		default:
			return false;
	}
}
```

онтролер Profile для POST запитів

омпонент script.js