Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

за спеціальності 121 – Інженерія програмного забезпечення

На тему: Розробка та реалізація веб-додатку для аналізу та оцінки фільмів з використанням технологій машинного навчання для системи рекомендацій

Засвідчую, що в цій кваліфікаційній роботі немає
запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-20-2
_____ / М.Г. Якуба /

Керівник
кваліфікаційної роботи _____ / А.В. Козиков /

Завідувач кафедри _____ / А.М. Стрюк /

Кривий Ріг
2024

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: бакалавр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри

_____ А.М. Стрюк

«__» _____ 20__р.

ЗАВДАННЯ

на кваліфікаційну роботу

студенту групи ІПЗ-20-2 Якубі Микиті Григоровичу

1. Тема: Розробка та реалізація веб-додатку для аналізу та оцінки фільмів з використанням технологій машинного навчання для системи рекомендацій затверджено наказом по КНУ №__ від «__» _____ 2024р.
2. Термін подання студентом закінченої роботи : «__» _____ 2024р.
3. Вихідні дані по роботі: розроблювана система повинна виконувати рекомендацію фільмів користувачу на основі вподобань користувача.
4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): аналіз потреб користувачів у виборі фільмів, стратегії збору та обробки даних, вибір відповідних алгоритмів рекомендацій, а також плани щодо впровадження та тестування системи.
5. Перелік ілюстративного матеріалу: блок-схеми, скріншоти дизайну інтерфейсу, скріншоти програмного коду.

Календарний план**:**

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Збір вхідних даних	23.01.2024 – 28.01.2024
2	Аналіз літератури та існуючих аналогів	02.02.2024 – 06.02.2024
3	Підготовка матеріалів першого розділу роботи	13.02.2024 – 18.02.2024
4	Підготовка матеріалів другого розділу	19.02.2024 – 26.02.2024
5	Розробка програмного забезпечення для реалізації системи	27.02.2024 – 10.03.2024
6	Розробка інтерфейсу	11.03.2024 – 21.03.2024
7	Розробка та реалізація основних функцій системи	22.03.2024 – 26.03.2024
8	Підготовка матеріалів третього розділу	26.03.2024 – 19.04.2024
9	Оформлення пояснювальної записки	22.04.2024 – 26.05.2024

Дата видачі завдання: «23» січня 2024р.

Студент _____/ М.Г. Якуба

Керівник роботи _____/ А.В. Козиков

РЕФЕРАТ

РЕКОМЕНДАЦІЙНА СИСТЕМА, ФІЛЬМИ, РЕКОМЕНДАЦІЇ

Пояснювальна записка: 58 с., 6 табл., 19 рис., 1 дод., 10 джерел.

У сучасному цифровому світі, коли вибір фільмів став насиченим та різноманітним завдяки безлічі онлайн-платформ для стрімінгу та кінематографічних послуг, проблема вибору відповідного контенту для перегляду стає актуальнішою, ніж коли-небудь раніше. Від кількості часу, який ми готові витратити на пошук наступного фільму, до особистих вподобань та настроїв, які можуть змінюватися з часом, ефективна система рекомендацій стає важливим інструментом для задоволення потреб глядачів.

Ця робота спрямована на розробку системи рекомендацій фільмів, яка базується на аналізі великого обсягу даних про фільми, враховуючи різноманітні аспекти, такі як жанр, рейтинг користувачів, рецензії та інші параметри.

Мета полягає в створенні інтелектуальної системи, яка здатна пропонувати персоналізовані рекомендації, враховуючи унікальні вподобання кожного користувача та забезпечуючи їм неперевершені кінематографічні враження.

ABSTRACT

RECOMMENDATION SYSTEM, MOVIES, RECOMMENDATIONS

Thesis: 58 p., 6 table, 19 figure, 1 appendix, 10 sources.

In today's digital world, where the choice of movies has become rich and diverse thanks to a plethora of online streaming platforms and cinema services, the challenge of choosing the right content to watch is more pressing than ever before. From the amount of time we are willing to spend searching for the next movie to personal preferences and moods that can change over time, an effective recommendation system becomes an important tool to meet the needs of viewers.

This work aims to develop a movie recommendation system based on the analysis of a large amount of movie data, taking into account various aspects such as genre, user rating, reviews, and other parameters.

The goal is to create an intelligent system that is able to offer personalized recommendations, taking into account the unique preferences of each user and providing them with an unrivaled cinematic experience. Translated with DeepL.com (free version)

ЗМІСТ

ВСТУП

У світі переповненому величезною кількістю фільмів і серіалів, вибір ідеального контенту може стати справжнім викликом. Проте, з розвитком технологій та алгоритмів рекомендацій, завдяки відкритим даним та машинному навчанню, з'являються інструменти, які допомагають кожному знайти те, що йому подобається. Ця програма з рекомендації фільмів ставить перед собою завдання полегшити процес вибору та забезпечити задоволення від перегляду.

В даному дослідженні ми докладемо зусиль для аналізу та порівняння різних методів рекомендаційних систем, визначимо їхні переваги та недоліки. Ми розглянемо різноманітні підходи до врахування індивідуальних вподобань глядачів, а також використовувані алгоритми та їх ефективність у забезпеченні персоналізованих рекомендацій.

Цей проект прагне не лише допомогти користувачам знайти відповідний контент для перегляду, але й пролити світло на методологію застосування рекомендаційних систем у сучасному світі розваг.

1 СУЧАСНИЙ СТАН І АКТУАЛЬНІСТЬ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1.1 Актуальність теми кваліфікаційної роботи

Сучасний швидкозмінюваний медіапейзаж та постійне поповнення асортименту кіно- та серіальних продуктів ставлять перед глядачами завдання не лише знаходити цікавий контент, але й ефективно відбирати його серед безлічі варіантів. Однак, досить часто користувачі стикаються з проблемою перенасиченості інформацією та вибору, який може бути навіть паралізуючим.

Тому актуальність цієї кваліфікаційної роботи полягає у тому, щоб вивчити та оцінити ефективність різноманітних методів рекомендаційних систем у кінопоказі.

Розробка та вдосконалення алгоритмів рекомендацій, які враховують індивідуальні вподобання користувачів, стає критично важливою для забезпечення їхнього задоволення від перегляду та збільшення часу, проведеного усвідомлено перед екраном. Такий аналіз не лише визначить оптимальні стратегії рекомендацій у сучасному кінематографічному середовищі, але й надасть важливі висновки для подальшого розвитку і вдосконалення рекомендаційних систем в медіаіндустрії.

Ще однією важливою складовою актуальності є стрімкий розвиток технологій та поява нових платформ для перегляду, що потребує постійного оновлення та адаптації рекомендаційних алгоритмів до змін у медіаландшафті. Зростання конкуренції на ринку стрімінгових послуг та вимоги споживачів до персоналізованого контенту підкреслюють актуальність пошуку ефективних рішень у сфері рекомендаційних систем для кінопоказу.

1.2 Цілі та завдання кваліфікаційної робот

Головною метою цієї кваліфікаційної роботи є дослідження різних методів та алгоритмів рекомендаційних систем у кінопоказі з метою підвищення якості рекомендаційного процесу для кіноглядачів. Завдання цієї роботи включають:

1) проведення огляду літератури з питань рекомендаційних систем у кінематографії для визначення основних підходів та методологій, що використовуються в цій галузі;

2) аналіз і порівняння різних методів рекомендаційних систем, зокрема коллаборативного та контентного підходів, для визначення їхньої ефективності та придатності до застосування в кінопоказі;

3) розробка прототипу рекомендаційної системи для кінопоказу на основі обраних алгоритмів та методів, що враховують індивідуальні уподобання та інтереси користувачів;

4) проведення експериментального дослідження для оцінки ефективності та точності рекомендаційної системи на основі показників, таких як точність, покриття та різноманітність рекомендацій;

5) висновки та рекомендації щодо подальшого вдосконалення та розвитку рекомендаційних систем для кінопоказу з метою задоволення потреб та очікувань сучасних глядачів.

1.3 Аналіз вимог

Перед розробкою рекомендаційної системи для кінопоказу важливо провести глибокий аналіз вимог, щоб гарантувати, що продукт відповідає вимогам сучасних кіноглядачів та враховує найновіші тенденції в цій сфері. Перш за все, необхідно врахувати концепцію персоналізації, яка є важливим аспектом у сучасних рекомендаційних системах. Кожен користувач має унікальні вподобання та інтереси, тому система повинна мати здатність адаптуватися до їхніх індивідуальних потреб.

Другим важливим аспектом є різноманітність контенту. Глядачі мають різноманітні смаки та уподобання, тому система повинна забезпечити

широкий вибір фільмів та серіалів різних жанрів, щоб задовольнити потреби різних аудиторій.

Крім того, важливо врахувати адаптивність системи. Кінопоказ та вподобання глядачів постійно змінюються, тому система повинна бути готова адаптуватися до цих змін та надавати актуальні рекомендації з часом.

Ефективність також є ключовим аспектом у розробці рекомендаційних систем. Швидкість та точність надання рекомендацій впливають на задоволення користувачів та їхню лояльність до платформи.

Не останнє місце займає доступність системи. Глядачі мають можливість перегляду на різних пристроях та платформах, тому система повинна бути доступною та зручною для використання на будь-якому пристрої.

Цей аналіз вимог допоможе зрозуміти потреби та очікування користувачів та визначити ключові характеристики, які повинні бути враховані під час розробки рекомендаційної системи для кінопоказу.

Основні аспекти цього аналізу включають:

1) персоналізація: система повинна здати враховувати унікальні вподобання та інтереси кожного користувача, щоб надавати персоналізовані рекомендації, які відповідають їхнім потребам та смакам;

2) різноманітність контенту: враховуючи різноманітність смаків і уподобань глядачів, система має забезпечувати рекомендації на основі різних жанрів та типів контенту;

3) адаптивність: важливо, щоб система була здатна адаптуватися до змін у вподобаннях та поведінці користувачів, надаючи актуальні та змістовні рекомендації з часом;

4) ефективність: при проектуванні системи необхідно враховувати її ефективність з точки зору швидкості обробки та надання рекомендацій, щоб забезпечити зручний та безперервний процес перегляду;

5) доступність: система повинна бути доступною для використання на різних платформах та пристроях, забезпечуючи гнучкість і зручність користування незалежно від обраного пристрою.

2 РОЗРОБКА ФУНКЦІОНАЛЬНОЇ СХЕМИ І АЛГОРИТМУ

2.1 Розробка та докладний опис функціональної схеми програми

Почнемо з основної, функціональної схеми програми. Функціональна блок-схема в інженерії програмного забезпечення - це блок-схема, яка описує функції та взаємозв'язки системи.

Функціональні блок-схеми описують взаємозв'язки системи разом з її основними функціями. Це основний, канонічний документ для будь-якої системи або програми перед складанням будь-яких інших більш складних планів технічного проектування.

Функціональна схема головної сторінки проектованої рекомендаційної системи для фільмів є такою:

1) назва та логотип. На верхній частині сторінки розташована назва сервісу разом з логотипом;

2) меню навігації. Поруч з назвою розташоване меню навігації з розділом «Home»;

3) головна частина сторінки містить список рекомендованих фільмів. Кожен фільм представлений своїм постером та зображенням, разом з назвою та коротким описом. Користувачі можуть переглядати цей список, шукати конкретні фільми за жанром або іншими параметрами, інтересами, або відвідувати сторінки фільмів, щоб дізнатися більше;

4) функціональні кнопки фільмів. Кожний фільм у списку має функціональні кнопки та іконки, які дозволяють користувачам взаємодіяти з ним. Наприклад:

   1) детальний опис. Кнопка або посилання, яке відкриває сторінку з більш детальним описом фільму;
   2) оцінка фільму. Можливість оцінювати фільм шляхом кліку на відповідну іконку або вибору рейтингу;

5) функціональні елементи авторизації. У верхній частині сторінки розташовані елементи для авторизації користувачів. Це кнопка "Увійти", яка

відкриває форму для введення облікових даних, або іконка облікового запису яка відображає профіль користувача якщо він вже увійшов в систему.

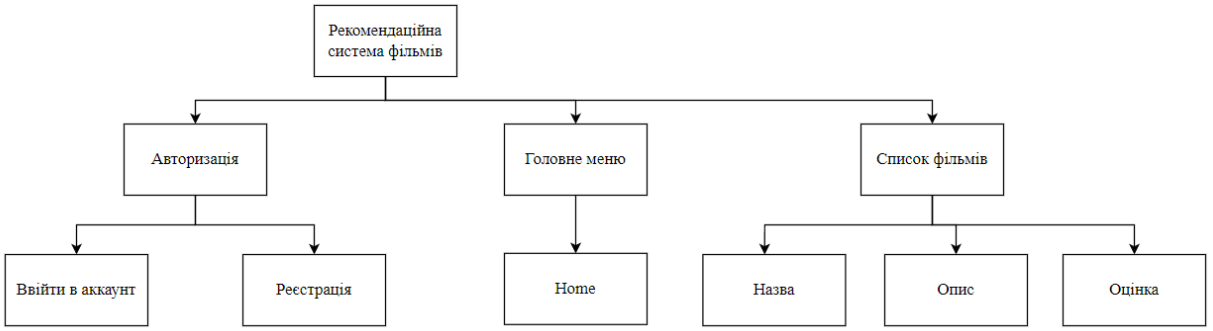

Рисунок 2.1 - Функціональна схема

Також функціональна схему можна деталізувати:

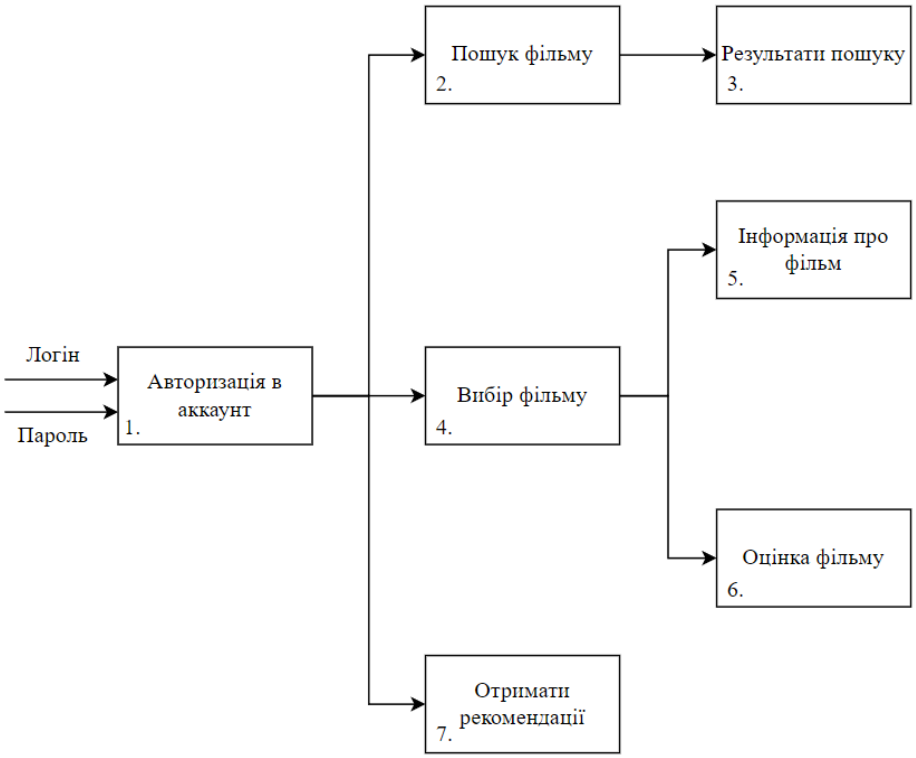

Рисунок 2.2 - Деталізована функціональна схема

Для більш детального розуміння роботи системи також необхідно реалізувати діаграму класів.

У програмній інженерії, діаграма класів на уніфікованій мові моделювання (UML) - це тип статичної структурної діаграми, яка описує структуру системи, показуючи класи системи, їхні атрибути, операції (або методи) та взаємозв'язки між об'єктами.

Таблиця 2.1 – Класи в системі

Клас	Опис
Користувач	Представляє користувача системи. Зберігає дані про ім'я, електронну пошту, пароль.
Фільм	Представляє фільм в системі. Містить дані, такі як назва, рік випуску, жанр, опис, рейтинг.
РейтингФільму	Зберігає оцінку користувача для конкретного фільму. Може містити ідентифікатор користувача, ідентифікатор фільму та оцінку.
УлюбленіФільми	Зберігає список улюблених фільмів для кожного користувача. Може містити ідентифікатор користувача та ідентифікатори улюблених фільмів.
ІсторіяПерегляду	Зберігає історію перегляду фільмів для кожного користувача. Містить ідентифікатор користувача та ідентифікатори переглянутих фільмів разом з датами перегляду.
Рекомендації	Генерує рекомендації фільмів для користувача на основі їхніх попередніх переглядів та оцінок. Може містити методи для визначення схожості фільмів та розрахунку рейтингів рекомендацій.
СистемаРекомендацій	Координує процес генерації рекомендацій для користувачів. Включає методи для аналізу даних користувачів та фільмів та генерації персоналізованих рекомендацій.
Аутентифікація	Відповідає за процес аутентифікації користувачів, включаючи вхід в систему та вихід з системи.
Фільтрація	Забезпечує функціональність пошуку та фільтрації фільмів за різними критеріями, такими як жанр, рік випуску і т.д.
Інтерфейс	Представляє інтерфейс користувача системи. Містить в собі методи для відображення сторінок, обробки запитів користувача та взаємодії з базою даних.

Загальна діаграма класів складається з 3 основних полів та має наступний вигляд:

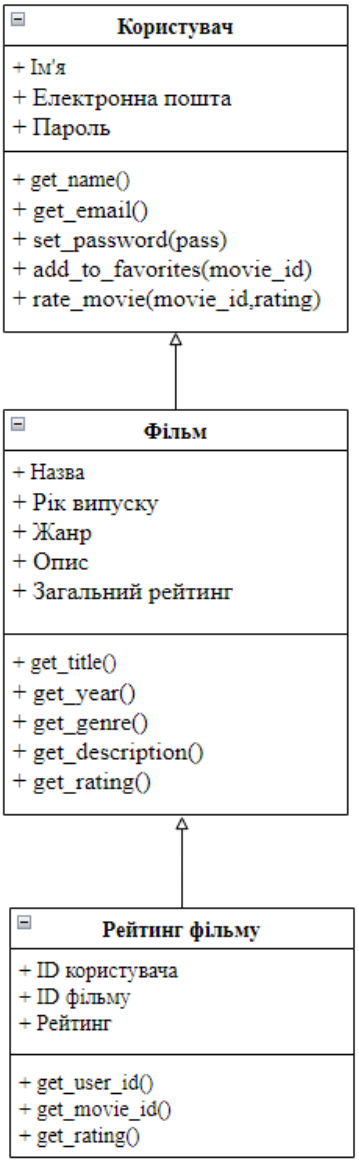

Рисунок 2.3 - Діаграма класів

1. Клас: Користувач (User)
    a. Поля:
        i. Ім'я (name)
        ii. Електронна пошта (email)
        iii. Пароль (password)
    b. Методи:
        i. get_name(): повертає ім'я користувача

ii. get_email(): повертає електронну пошту користувача
iii. set_password(new_password): змінює пароль користувача на новий
iv. add_to_favorites(movie_id): додає фільм до списку улюблених користувача
v. rate_movie(movie_id, rating): оцінює фільм користувачем

2. Клас: Фільм (Movie)
    a. Поля:
3. Назва (title)
        i. Рік випуску (release_year)
        ii. Жанр (genre)
        iii. Опис (description)
        iv. Рейтинг (rating)
    b. Методи:
        i. get_title(): повертає назву фільму
        ii. get_release_year(): повертає рік випуску фільму
        iii. get_genre(): повертає жанр фільму
        iv. get_description(): повертає опис фільму
        v. get_rating(): повертає рейтинг фільму
4. Клас: РейтингФільму (MovieRating)
    a. Поля:
        i. Ідентифікатор користувача (user_id)
        ii. Ідентифікатор фільму (movie_id)
        iii. Оцінка (rating)
5. Методи:
        i. get_user_id(): повертає ідентифікатор користувача, який залишив оцінку
        ii. get_movie_id(): повертає ідентифікатор фільму, який було оцінено
        iii. get_rating(): повертає оцінку фільму

2.2 Алгоритм рекомендацій фільмів

Для реалізації самого алгоритму був вибраний метод спільної фільтрації з використанням методу мінімізації функції витрат (Collaborative Filtering with Optimization Function).

Система рекомендацій може бути персоналізованою або неперсоналізованою. Неперсоналізовані системи можуть бути простішими, але персоналізовані системи працюють краще, оскільки вони задовольняють потреби кожного окремого користувача.

Спільна фільтрація - це поширений метод персоналізованої системи рекомендацій, який фільтрує інформацію, наприклад, дані про взаємодію з іншими подібними користувачами. Оскільки він працює на основі прогнозування оцінок користувачів, вважається, що він виконує завдання регресії. Існує два основних типи колаборативної фільтрації, від користувача до користувача (який в реалізованій системі і використовується), і від елемента до елемента.

Спільна фільтрація між користувачами в основному працює на основі припущення, що користувачі, які поставили однакові оцінки певному елементу, ймовірно, мають такі ж уподобання і щодо інших елементів. Тому цей метод в основному покладається на пошук схожості між користувачами.

Алгоритм роботи спільної фільтрації між користувачами має наступний вигляд:

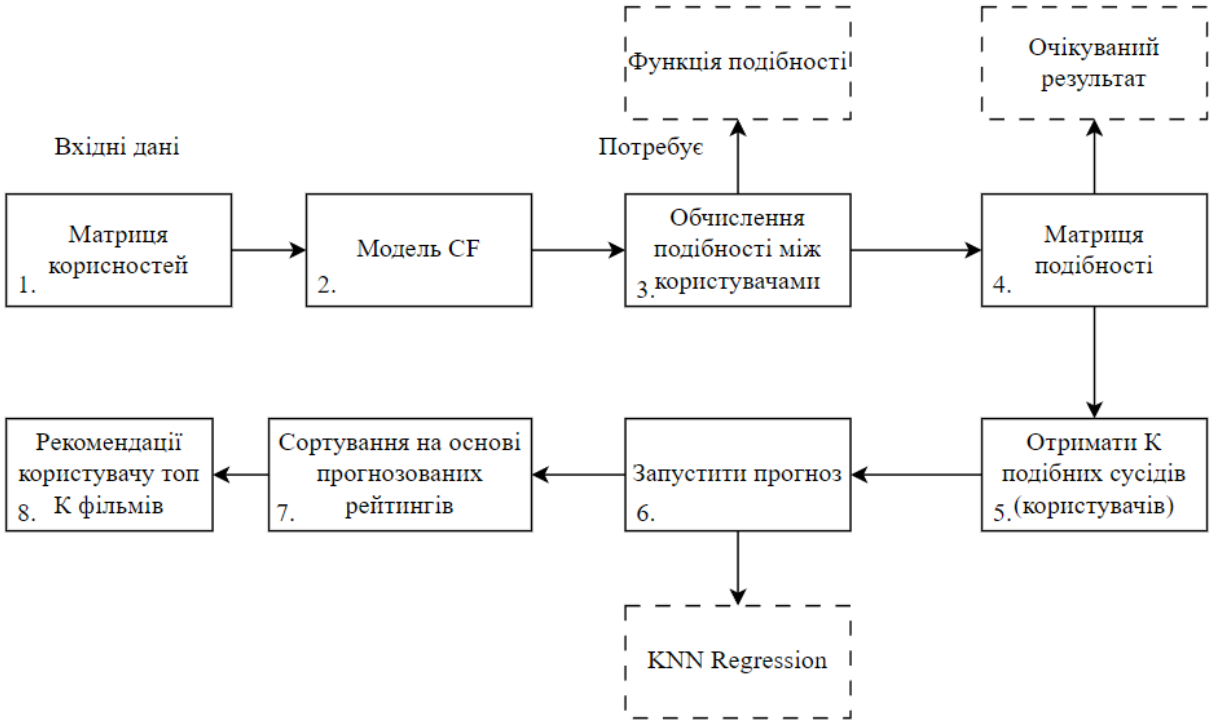

Рисунок 2.4 – Алгоритм роботи спільної фільтрації на основі користувачів

Процес починається з перетворення даних рейтингу в матрицю корисності, де список користувачів є рядками, а список фільмів - стовпчиками.

Наступним кроком є модель колаборативної фільтрації «Сусіди», де використовуємо функцію схожості для обчислення схожості між користувачами, а на виході отримуємо матрицю схожості.

Береться певна кількість (K) схожих користувачів (сусідів), і прогноз рейтингу буде отримано шляхом регресії на основі даних рейтингу цих сусідів.

Потім елементи будуть відсортовані на основі найвищого рейтингу, і найкращі елементи будуть рекомендовані користувачеві.

В випадку конкретно нашої рекомендаційної системи також реалізується нормалізація рейтингів та оптимізація функції втрат для кращої,

більш точної та швидкої роботи алгоритму. Реалізується це за допомогою методу мінімізації функції витрат.

Основні компоненти алгоритму:

1) нормалізація рейтингів: під час підготовки даних рейтинги нормалізуються, щоб врахувати різницю у рівні оцінювання між користувачами. Це допомагає уникнути перекосів у прогнозах;

2) оптимізація функції витрат: застосовується метод мінімізації функції витрат для знаходження оптимальних значень параметрів моделі (матриць X та Theta), які найкраще відображають залежності між користувачами та фільмами;

3) побудова прогнозних оцінок: на основі оптимальних значень параметрів моделі розраховуються прогнозні оцінки рейтингів для усіх фільмів. Ці прогнози використовуються для рекомендацій нових фільмів користувачам;

4) повернення результату: повертається матриця прогнозів оцінок (prediction_matrix) та середні оцінки користувачів (Ymean), які можуть бути використані для рекомендацій.

В цілому, можна зробити висновок що метод спільної фільтрації на основі користувачів добре підходить для реалізації системи рекомендацій фільмів. Але також звичайно такий метод має свої мінуси.

Таблиця 2.2 – Плюси та мінуси методу спільної фільтрації на основі користувачів

Плюси	Мінуси
Забезпечує персоналізовані рекомендації, орієнтовані на індивідуальні вподобання користувача	Важкість забезпечення рекомендацій новим користувачам або новим об'єктам (проблема холодного старту)
Не потрібні додаткові апріорні знання про користувачів чи об'єкти	Проблема розрідженості даних, особливо у великих системах з багатьма користувачами та об'єктами
Ефективний для надання рекомендацій в умовах обмеженої інформації про користувачів та об'єкти	Складність у знаходженні достатньо схожих користувачів або об'єктів для здійснення ефективних рекомендацій
Дозволяє враховувати динаміку змін в уподобаннях користувачів та нових об'єктів	Обчислювальна складність може зростати зі збільшенням кількості користувачів та об'єктів
Простота в реалізації та використанні	Можливість виникнення проблеми "паразитних ефектів" (вплив оцінок менш активних користувачів на рекомендації для активних користувачів)

2.3 Розробка та опис структури бази даних

База даних буде скаладиться з 7 різних таблиць:

1. auth_permission – містить в собі інформацію щодо функціоналу який буде доступний для користувача.

```
SELECT * FROM 'auth_permission' LIMIT 0,30
```

Execute

id	content_type_id	codename	name
1	1	add_movie	Can add movie
2	1	change_movie	Can change movie
3	1	delete_movie	Can delete movie
4	2	add_myrating	Can add myrating
5	2	change_myrating	Can change myrating
6	2	delete_myrating	Can delete myrating
7	3	add_logentry	Can add log entry
8	3	change_logentry	Can change log entry
9	3	delete_logentry	Can delete log entry
10	4	add_permission	Can add permission

Рисунок 2.5 – Перші 10 записів таблиці БД auth_permission

2. web_movie – містить в собі інформацію щодо фільмів.

```
SELECT * FROM 'web_movie' LIMIT 0,30
```

Execute

id	title	genre	movie_logo
1	Toy Story (1995)	Animation|Adventure|Comedy	Toy Story (1995).jpg
2	Jumanji (1995)	Action|Adventure|Family	Jumanji (1995).jpg
3	Grumpier Old Men (1995)	Comedy|Romance	Grumpier Old Men (1995).jpg
4	Waiting to Exhale (1995)	Comedy|Drama|Romance	Waiting to Exhale (1995).jpg
5	Father of the Bride Part II (1995)	Comedy|Family|Romance	Father of the Bride Part II (1995).jpg
6	Heat (1995)	Action|Crime|Drama	Heat (1995).jpg
7	Sabrina (1995)	Comedy|Drama	Sabrina (1995).jpg
8	Sudden Death (1995)	Action|Crime|Thriller	Sudden Death (1995).jpg
9	GoldenEye (1995)	Action|Adventure|Thriller	GoldenEye (1995).jpg
10	The American President (1995)	Comedy|Drama|Romance	The American President (1995).jpg

Рисунок 2.6 – Перші 10 записів таблиці БД web_movie

3. web_myrating – містить в собі інформацію щодо оцінки рейтингу фільмів поставлених користувачами.

```
SELECT * FROM 'web_myrating' LIMIT 0,30
```

Execute

id	rating	movie_id	user_id
1	4	1	1
2	5	2	1
3	3	3	1
4	2	4	1
5	1	5	1
6	5	6	1
7	4	7	1
8	3	8	1
9	4	9	1
10	5	10	1

Рисунок 2.7 – Перші 10 записів таблиці БД web_myrating

4.auth_user – усі зареєстровані користувачі.

```
SELECT * FROM 'auth_user' LIMIT 0,30
```

Execute

id	password	last_login	is_superuser	username	first_name	email	is_staff	is_active	date_joined	last_name
1	pbkdf2_sha256$100000$...		1	asif			1	1		
2	pbkdf2_sha256$100000$...		0	dibyo		dibyo123@gmail.com	0	1		
3	pbkdf2_sha256$100000$...		0	vivek		vivek123@gmail.com	0	1		
4	pbkdf2_sha256$100000$...		0	ayush		ayush123@gmail.com	0	1		
5	pbkdf2_sha256$100000$...		0	dfkjm		dfmkm@sdfm.com	0	1		
6	pbkdf2_sha256$100000$...		0	dasjk		jsdj@gmail.com	0	1		
7	pbkdf2_sha256$100000$...		0	dfjfdkm		dfmsn@mgf.com	0	1		
8	pbkdf2_sha256$100000$...		0	dsjfhji		jsdkanmj@jdnm.com	0	1		
9	pbkdf2_sha256$100000$...		0	tyhiok		kfg@jmgf.com	0	1		
10	pbkdf2_sha256$100000$...		0	fkfik		fdj@ijdj.co	0	1		

Рисунок 2.8 – Перші 10 записів таблиці БД auth_user

5. django_content_type – типи контенту присутні на сторінці.

```
SELECT * FROM 'django_content_type' LIMIT 0,30
```

Execute

id	app_label	model
1	web	movie
2	web	myrating
3	admin	logentry
4	auth	permission
5	auth	group
6	auth	user
7	contenttypes	contenttype
8	sessions	session

Рисунок 2.9 – Перші 8 записів таблиці БД django_content_type

6. sqlite_sequence – містить в собі інформацію про кількість записів у інших таблицях.

```
SELECT * FROM 'sqlite_sequence' LIMIT 0,30
```

Execute

name	seq
django_migrations	15
django_admin_log	0
django_content_type	8
auth_permission	24
auth_user	28
web_movie	148
web_myrating	502

Рисунок 2.10 – Таблиця sqlite_sequence

2.4 Розробка інтерфейсу програми

Хороший інтерфейс для проектів рекомендаційних систем є ключовим елементом для забезпечення задоволення користувачів, покращення їхнього досвіду взаємодії з системою та збільшення шансів на успіх проекту. Інтерфейс повинен:

1) бути легким у використанні та інтуїтивно зрозумілим для користувачів. Простота навігації та доступ до основних функцій, таких як пошук фільмів, перегляд рекомендацій та оцінка контенту, є важливими для забезпечення комфортного досвіду користувача;

2) підтримувати персоналізовані рекомендації та налаштування профілю користувача. Це дозволяє кожному користувачеві отримувати рекомендації, які відповідають його унікальним інтересам та вподобанням;

3) бути досить естетичним, простим та водночас привабливим.

Приклад схеми реалізації інтерфейсу наступний:

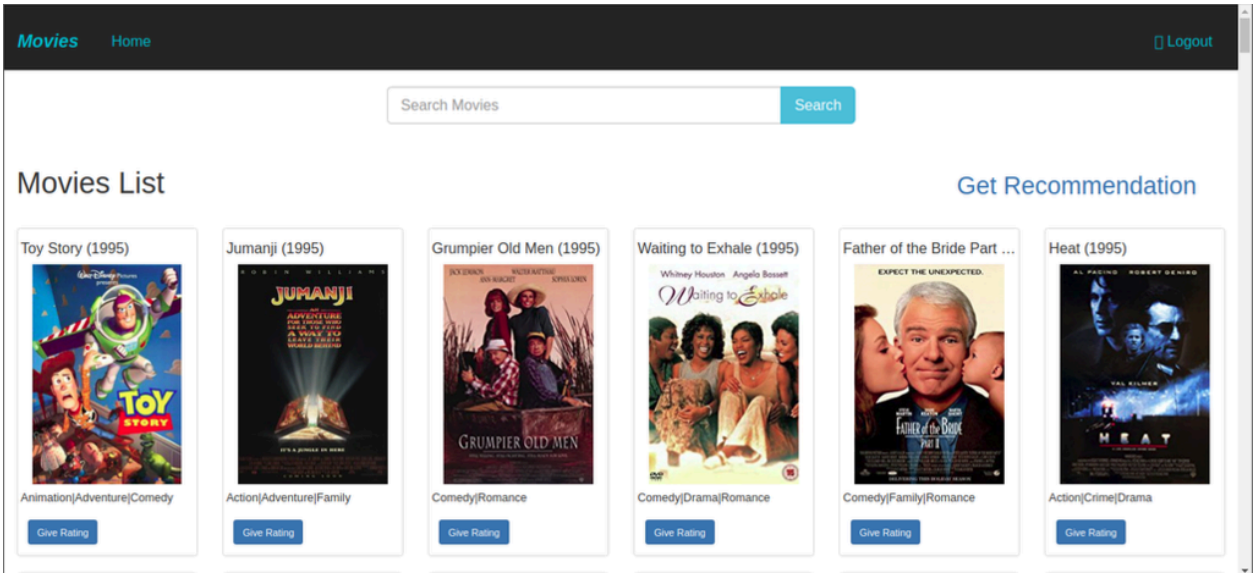

Рисунок 2.11 - Схема реалізації інтерфейсу

Користувач на головній сторінці може ввійти або вийти з акаунту, здійснити пошук по фільмах, оцінити окремий фільм, або отримати рекомендації натиснувши на кнопку.

Після того як користувач натиснув на кнопку «Get Recommendation» він отримує список рекомендацій.

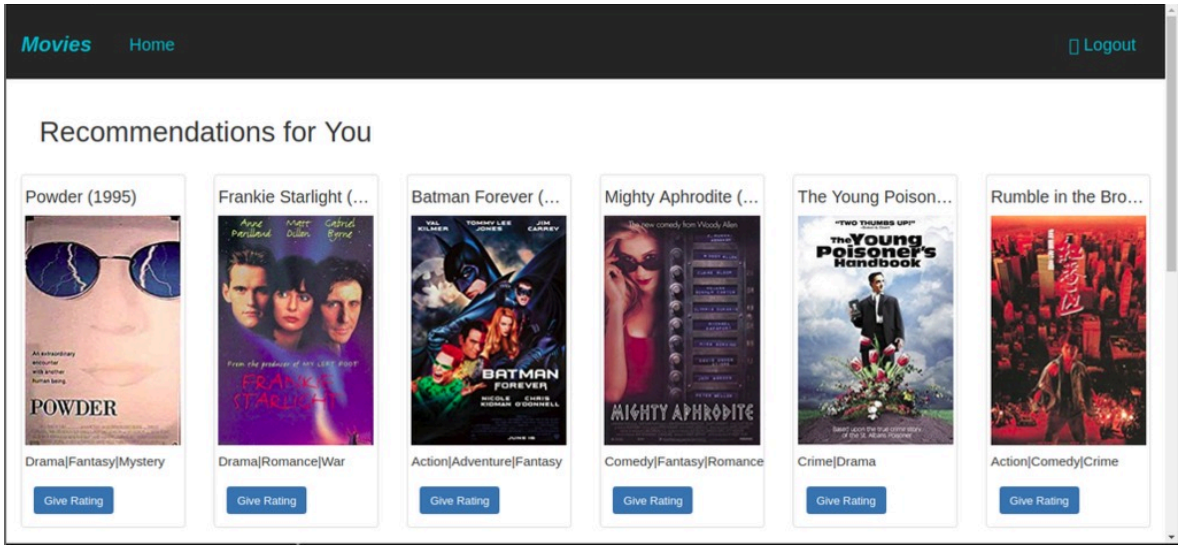

Рисунок 2.12 - Список рекомендацій які отримує користувач

Також користувач може дізнатися більше про фільм натиснувши на нього.

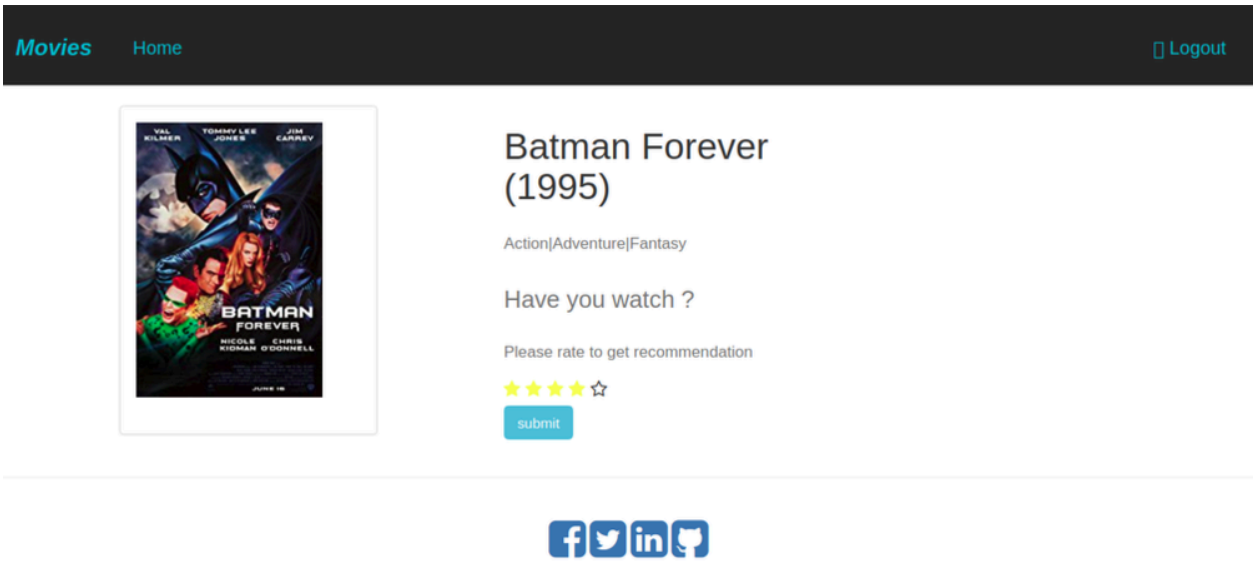

Рисунок 2.13 – Більш детальна інформація про фільм

2.5 Опис функціональних та нефункціональних вимог

Рекомендаційна система фільмів має забезпечувати користувачам можливість реєстрації та входу в систему, перегляду списку фільмів, оцінювання фільмів, зберігання улюблених фільмів та перегляд історії перегляду. Кожен фільм повинен мати детальну сторінку з інформацією про нього, таку як назва, рік випуску, жанр та опис. Система повинна надавати

персоналізовані рекомендації користувачам на основі їхніх попередніх дій та взаємодії з системою.

Таблиця 2.2 – Функціональні вимоги

Функція	Опис
Реєстрація користувачів	Можливість створення нових облікових записів користувачів
Вхід в систему	Можливість входу в систему для зареєстрованих користувачів
Перегляд списку фільмів	Відображення списку фільмів, доступних для перегляду
Оцінювання фільмів	Можливість користувачів оцінювати фільми
Зберігання улюблених фільмів	Функціональність для збереження улюблених фільмів користувачів
Перегляд історії перегляду	Відображення історії перегляду фільмів користувачем
Детальна сторінка фільму	Сторінка з докладною інформацією про кожен фільм
Персоналізовані рекомендації	Надання персоналізованих рекомендацій користувачам

Нефункціональні вимоги визначають атрибути системи, такі як продуктивність, безпека та надійність, які не відображаються безпосередньо у функціональності системи, але впливають на її ефективність та користувальницький досвід.

Таблиця 2.3 – Нефункціональні вимоги до системи

Нефункціональні вимоги	Опис
Продуктивність	Система повинна завантажувати сторінки швидко та ефективно, навіть при великому обсязі даних та одночасному доступі багатьох користувачів.
Масштабованість	Система повинна бути здатною масштабуватися для впорядкування зростаючого обсягу даних та забезпечення стабільної роботи при збільшенні користувачів.
Безпека	Забезпечення конфіденційності, цілісності та доступності даних користувачів через використання надійних методів аутентифікації та шифрування.
Надійність	Система повинна бути стійкою до відмов та забезпечувати високий рівень доступності, а також мінімізувати ризик втрати даних.
Ергономіка	Інтерфейс користувача повинен бути зрозумілим, зручним у використанні та добре структурованим для покращення користувальницького досвіду.

2.6 Опис використаних технологій під час розробки системи

Були використані наступні веб технології:

1) HTML (HyperText Markup Language). HTML є основною мовою розмітки для створення структури веб-сторінок. Використання HTML дозволяє створювати різноманітні елементи сторінки, такі як текст, заголовки, списки, зображення та посилання;

2) CSS (Cascading Style Sheets). CSS використовується для визначення зовнішнього вигляду елементів веб-сторінки, таких як кольори,

шрифти, розміри, відступи та макети. Використання CSS дозволяє створювати стильні та привабливі веб-сайти зі стандартною розміткою HTML;

3) JavaScript. JavaScript є мовою програмування, яка використовується для створення динамічних та інтерактивних елементів на веб-сторінках. В нашому випадку використання JavaScript необіхдне для реалізації функцій, анімації, перевірки форм, взаємодії з користувачем та багато іншого;

4) Bootstrap. Bootstrap є потужним фреймворком CSS, який надає набір готових компонентів та стилів для швидкої розробки адаптивних та стильних веб-сторінок. Використання Bootstrap дозволяє швидко створювати професійний вигляд веб-сайту, забезпечуючи при цьому адаптивність для різних пристроїв;

5) Django. Django є потужним фреймворком для розробки веб-додатків на мові програмування Python. Використання Django дозволяє швидко створювати веб-сайти зі складними функціональними можливостями, такими як автентифікація користувачів, робота з базами даних, маршрутизація URL.

У Python3 для машинного навчання використовувалися такі бібліотеки:

1) NumPy. NumPy - це основна бібліотека для наукових обчислень у Python. Вона надає підтримку для масивів та матриць, а також високорівневі математичні функції, що робить її ідеальним інструментом для обробки даних та матричних обчислень у машинному навчанні;

2) Pandas. Pandas - це потужна бібліотека для аналізу та обробки даних у Python. Вона надає структури даних, такі як DataFrame, які дозволяють легко і ефективно виконувати операції над даними, такі як фільтрування, сортування, групування та зведення даних. Pandas дуже корисний для підготовки та очищення даних перед їх використанням у моделях машинного навчання. SciPy. SciPy - це бібліотека для наукових

обчислень, яка розширює функціональність NumPy, надаючи додаткові інструменти для оптимізації, інтерполяції, інтегрування, а також для роботи з різними видами даних. SciPy часто використовується для реалізації алгоритмів машинного навчання та статистичного аналізу;

Як база даних було використано SQLite. SQLite - це вбудована, серверна база даних, яка працює без окремого серверу баз даних. Вона зберігає базу даних у єдиному файлі на диску, що дозволяє легко і просто використовувати її в різних застосунках та проектах. SQLite проста у встановленні та налаштуванні, надійна, кросплатформенна, ефективна та розширювана. Ця база даних є ідеальним вибором для проектів, де потрібно зберігати та управляти даними, включаючи застосунки з машинним навчанням.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА БАЗИ ДАНИХ

3.1 Аналіз обраної среди

У аналізі обраної системи Python для системи рекомендацій фільмів можна виділити кілька ключових аспектів.

Перш за все, Python надає широкий вибір бібліотек для розробки машинного навчання та аналізу даних. Це включає в себе такі популярні інструменти, як Pandas, NumPy, SciPy, Scikit-learn, TensorFlow та PyTorch. Завдяки цьому, розробники можуть легко реалізувати та налаштувати алгоритми рекомендацій, а також проводити експерименти з різними моделями.

Крім того, Python відомий своєю простотою та читабельністю коду, що полегшує розробку та супровід системи рекомендацій фільмів. Це особливо корисно для команд, що працюють над проектом, оскільки спрощує спілкування та співпрацю між учасниками.

Тому ми маємо велику та активну спільноту розробників, яка надає підтримку та розвиває різноманітні інструменти для розробки програм. Це створює сприятливі умови для розвитку та вдосконалення системи рекомендацій фільмів з плином часу.

Також було обрано PyCharm - це інтегроване середовище розробки (IDE) для мови програмування Python, яке надає широкий спектр інструментів для зручного написання, тестування та налагодження програм. Воно відзначається високою продуктивністю та можливістю розширення завдяки ряду корисних функцій, таких як автодоповнення коду, підказки по API, інтегрований тестувальний фреймворк, система контролю версій та інші.

В цілому ми маємо інтуїтивно зрозумілий інтерфейс, що дозволяє розробникам ефективно працювати з кодом та зосередитися на процесі розробки, а не на пошуку інструментів чи налаштування середовища. Його потужність полягає в можливості розробки як простих скриптів, так і великих проектів, забезпечуючи при цьому стабільну та швидку роботу.

PyCharm - це інтегроване середовище розробки (IDE) для мови програмування Python, розроблене компанією JetBrains. Воно надає широкі можливості для зручного програмування на Python, включаючи підтримку роботи з проектами, інтеграцію з версійним контролем, автодоповнення коду, рефакторинг, відлагодження програм, підтримку віртуальних середовищ та багато іншого.

PyCharm має дружній інтерфейс та ряд інструментів, спрямованих на полегшення роботи програмістів на Python. Від відкриття нового проекту до виведення готового продукту, PyCharm пропонує різноманітні функції, що сприяють ефективному програмуванню та підвищують продуктивність розробника.

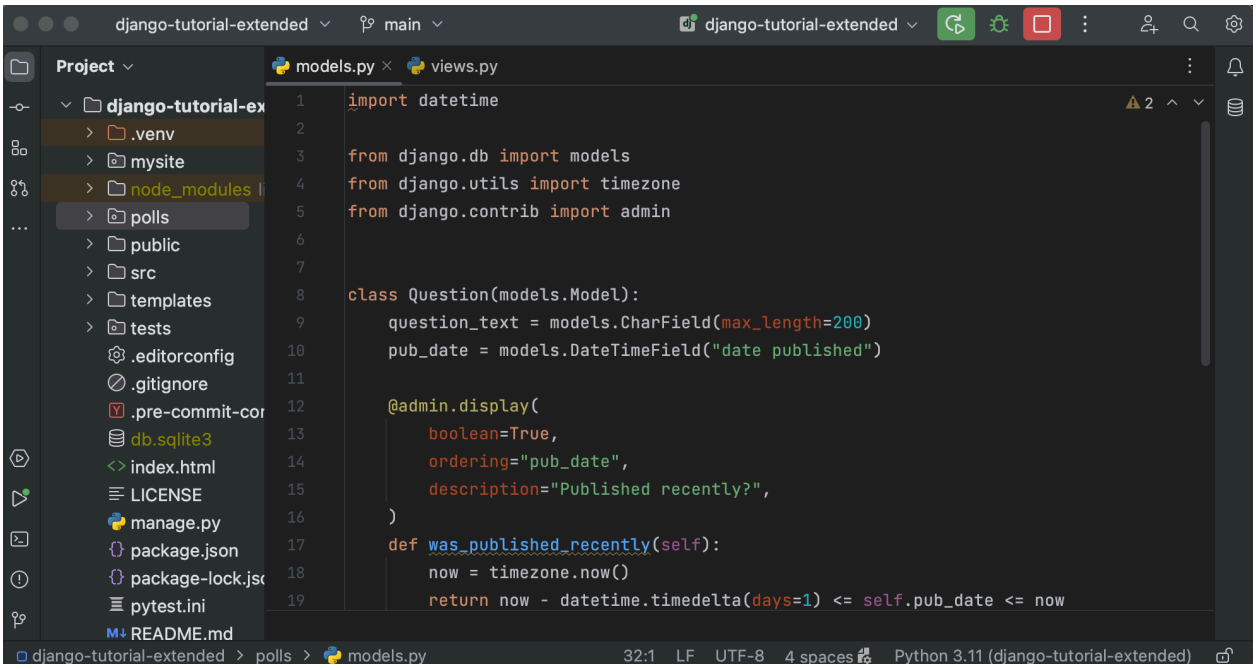

Рисунок 3.1 Інтерфейс програми PyCharm

Таблиця 3.1 – Плюси та мінуси середи розробки PyCharm

Плюси	Мінуси
Потужний інструмент для розробки	Висока вартість платної версії
Велика кількість функцій та інструментів	Великий обсяг ресурсів, потрібних для запуску
Підтримка різних фреймворків і бібліотек Python	Важкий для початківців
Інтеграція зі засобами версійного контролю	Потребує певного часу для освоєння всіх можливостей
Відмінна підтримка віртуальних середовищ Python	Великий обсяг функцій може здаватися зайвим для простих проектів
Велика спільнота користувачів та підтримка розробників	

3.2 Вибір датасету

Датасет фільмів містить різноманітні поля, які дозволяють ідентифікувати та описати кожну стрічку. Серед основних атрибутів можна виділити назву фільму, жанр, тривалість, рейтинг та посилання на постери. Ці дані допомагають користувачам швидко знаходити та вибирати фільми за їх ключовими характеристиками.

Крім основних полів, датасет може містити додаткові атрибути, такі як опис сюжету, дата виходу, режисер, акторський склад та інші деталі. Ці додаткові дані розширюють можливості аналізу та рекомендацій, допомагаючи користувачам знайти фільми, які відповідають їхнім вподобанням та інтересам.

Загальна структура датасету дозволяє ефективно використовувати дані для створення різноманітних алгоритмів рекомендацій та аналізу популярності фільмів. Кожен атрибут грає важливу роль у формуванні повної

та зрозумілої інформації про кожен фільм, що робить датасет корисним інструментом для досліджень у галузі кінематографу.

Таблиця 3.2 – Поля та їх опис.

Поле	Опис
Назва	Назва фільму
Жанр	Жанр фільму
Тривалість	Тривалість фільму у хвилинах
Рейтинг	Рейтинг фільму
Постер	Посилання на зображення постера
Опис	Опис сюжету фільму
Режисер	Режисер фільму
Актори	Список акторів у фільмі
Рік	Рік виходу фільму
Оцінки	Рейтинги фільму на різних платформах

3.3 Програмна реалізація основних функцій

3.3.1 Програмна реалізація логіки на Python

Основою будь-якої програмної реалізації системи рекомендацій фільмів є функція main, призначена для запуску основних алгоритмів та функціональностей системи. Проте, важливо також врахувати різноманітні аспекти обробки даних, такі як завантаження даних про фільми, видалення зайвої інформації, а також ефективну роботу з ними для забезпечення точності та ефективності рекомендацій.

Нижче наведено приклади фрагментів коду, які демонструють реалізацію цих функцій для подальшого використання їх в інших частинах програми. Код надає можливість завантажувати дані про фільми з вхідного

джерела, виконувати обробку цих даних та здійснювати рекомендації на основі заданих критеріїв.

```
def recommend(request):
     if not request.user.is_authenticated:
          return redirect("login")
     if not request.user.is_active:
          raise Http404
     df=pd.DataFrame(list(Myrating.objects.all().values()))
     nu=df.user_id.unique().shape[0]
     current_user_id= request.user.id
     # if new user not rated any movie
     if current_user_id>nu:
          movie=Movie.objects.get(id=15)
          q=Myrating(user=request.user,movie=movie,rating=0)
          q.save()

     print("Current user id: ",current_user_id)
     prediction_matrix,Ymean = Myrecommend()
     my_predictions =
prediction_matrix[:,current_user_id-1]+Ymean.flatten()
     pred_idxs_sorted = np.argsort(my_predictions)
     pred_idxs_sorted[:] = pred_idxs_sorted[::-1]
     pred_idxs_sorted=pred_idxs_sorted+1
     print(pred_idxs_sorted)
     preserved = Case(*[When(pk=pk, then=pos) for pos, pk in
enumerate(pred_idxs_sorted)])
     movie_list=list(Movie.objects.filter(id__in =
pred_idxs_sorted,).order_by(preserved)[:10])
     return
render(request,'web/recommend.html',{'movie_list':movie_list})
```

Цей код працює наступним чином:

1) перевірка, чи користувач аутентифікований. Якщо користувач не увійшов у систему, відбувається перенаправлення на сторінку входу;

2) перевірка, чи користувач активний. Якщо обліковий запис користувача неактивний, генерується помилка HTTP 404;

3) створення DataFrame з усіх записів об'єктів Myrating;

4) обчислення кількості унікальних користувачів, які оцінили фільми;

5) отримання ідентифікатора поточного користувача;

6) якщо поточний користувач новий і не оцінив жодного фільму, додається запис з оцінкою 0 для фільму з ID 15;

7) виклик функції Myrecommend для отримання прогнозів рекомендацій. Розрахунок особистих прогнозів рейтингів для фільмів;

8) сортування індексів фільмів за прогнозованими рейтингами у порядку спадання;

9) створення виразу для збереження порядку сортування фільмів;

10) вибірка десяти фільмів з бази даних, які найбільш відповідають рекомендаціям, і відображення їх на сторінці за допомогою шаблону 'recommend.html';

11) повернення відповіді на запит зі списком рекомендованих фільмів для користувача.

Наступний код створює моделі даних для Django-додатка.

```
from django.contrib.auth.models import Permission, User
from django.core.validators import MaxValueValidator,
MinValueValidator
from django.db import models

class Movie(models.Model):
     title     = models.CharField(max_length=200)
     genre          = models.CharField(max_length=100)
     movie_logo  = models.FileField()

     def __str__(self):
          return self.title

class Myrating(models.Model):
     user      =
models.ForeignKey(User,on_delete=models.CASCADE)
     movie     =
models.ForeignKey(Movie,on_delete=models.CASCADE)
     rating    =
models.IntegerField(default=1,validators=[MaxValueValidator(5),M
inValueValidator(0)])
```

Movie (фільм) - модель, що відображає дані про фільм. Містить поля для назви фільму (title), жанру (genre) та логотипу фільму (movie_logo), який зберігається як файл.

Myrating (оцінка) - модель, що відображає оцінки користувачів для фільмів. Містить поля, що пов'язуються з користувачем (user) та фільмом (movie), а також поле rating, що представляє собою ціле число від 0 до 5 зі

стандартним значенням 1. Це поле також має валідатори для перевірки, що значення знаходиться в межах від 0 до 5.

Наступний код відповідає за авторизацію та реєстрацію користувачів.

```
# Register user
def signUp(request):
     form =UserForm(request.POST or None)
     if form.is_valid():
          user      = form.save(commit=False)
          username  =    form.cleaned_data['username']
          password  = form.cleaned_data['password']
          user.set_password(password)
          user.save()
          user =
authenticate(username=username,password=password)
          if user is not None:
               if user.is_active:
                    login(request,user)
                    return redirect("index")
     context ={
          'form':form
     }
     return render(request,'web/signUp.html',context)



# Login User
def Login(request):
     if request.method=="POST":
          username = request.POST['username']
          password = request.POST['password']
          user     =
authenticate(username=username,password=password)
          if user is not None:
               if user.is_active:
                    login(request,user)
                    return redirect("index")
               else:
                    return
render(request,'web/login.html',{'error_message':'Your account
disable'})
          else:
               return
render(request,'web/login.html',{'error_message': 'Invalid
Login'})
     return render(request,'web/login.html')

#Logout user
def Logout(request):
     logout(request)
     return redirect("login")
```

1) signUp. Ця функція обробляє запити на реєстрацію користувача. Вона перевіряє, чи є вхідні дані форми коректними, зберігає нового користувача в базу даних, а також автоматично автентифікує його в системі, якщо реєстрація пройшла успішно;

2) login. Функція обробляє запити на вхід користувача. Вона перевіряє, чи існує користувач з введеними даними, і якщо так, то автентифікує його в системі. Якщо обліковий запис користувача активний, користувач автоматично входить в систему і перенаправляється на головну сторінку. В іншому випадку виводиться повідомлення про відключений обліковий запис або про невірні дані входу$;

3) logout. Функція обробляє запити на вихід користувача. Вона видаляє дані сесії користувача, тим самим виходячи з системи, і перенаправляє користувача на сторінку входу.

Тобто під час реєстрації нового користувача введені дані перевіряються на коректність, після чого створюється запис у базі даних. Після успішної реєстрації користувач автоматично автентифікується в системі.

При вході в систему дані користувача перевіряються, і якщо вони вірні, користувач автентифікується та отримує доступ до облікового запису. При виході з системи вся інформація сесії користувача видаляється, і він перенаправляється на сторінку входу.

3.3.2 Програмна реалізація розмітки веб-сторінок

Було використано HTML для створення основної структури веб-сторінки, визначаючи різні елементи, такі як заголовки, параграфи та форми. CSS використовувався для стилізації цих елементів, задавання їхнього вигляду, кольору, шрифту, розміщення та інших властивостей, щоб створити привабливий дизайн сторінки. Також було використано сіткову систему для організації розташування елементів на сторінці та забезпечення її адаптивності до різних розмірів екранів.

Приклад HTML та CSS коду головної сторінки:

```
<!DOCTYPE html>
<html>
<head>
     <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1">
    <meta http-equiv="X-UA-Compatible"
content="IE=edge,chrome=1">
    <title>{% block title %}Movies{% endblock title %}</title>
    {% load staticfiles %}

     <link rel="stylesheet" type="text/css" href="{% static
'web/css/bootstrap.min.css'%}">
     <link rel='stylesheet'
href='https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/fon
t-awesome.min.css' >
    <link
href='http://fonts.googleapis.com/css?family=Open+Sans:400,300,7
00' rel='stylesheet' type='text/css'>
    <link rel='stylesheet' href='{% static "web/css/base.css"
%}' />

    <style type="text/css">
        .thumbnail p, .thumbnail h4 {
            white-space: nowrap;
            text-overflow: ellipsis;
            overflow: hidden;
            }
        .star-rating {
            line-height:32px;
            font-size:1.25em;
            }

.star-rating .fa-star{color: yellow;}

    </style>
</head>
<body>
     <nav class="navbar navbar-inverse">
                <div class="container-fluid">

                    <!-- Header -->
                    <div class="navbar-header">
                        <button type="button"
class="navbar-toggle" data-toggle="collapse"
data-target="#topNavBar">
                            <span class="icon-bar"></span>
                            <span class="icon-bar"></span>
                            <span class="icon-bar"></span>
                        </button>
                        <a class="navbar-brand"
href="">Movies</a>
                    </div>
```

Цей HTML-код створює загальний макет сторінки веб-сайту для відображення фільмів.

Короткий опис кожної частини:

1) заголовок документа. Встановлюються різні метатеги, такі як кодування символів, метатег перегляду та сумісність з браузерами. Заголовок сторінки встановлюється за допомогою шаблонного блоку, який може бути перезаписаний у поточному веб-застосунку;

2) підключення зовнішніх ресурсів. Підключення зовнішніх таблиць стилів, таких як Bootstrap, шрифтів та іконок;

3) внутрішні стилі. Визначення деяких додаткових стилів прямо в HTML-документі для керування виглядом певних елементів, таких як мініатюри фільмів та рейтингові зірки;

4) навігаційна панель. Створення навігаційної панелі з використанням компонентів Bootstrap для адаптивного відображення на різних пристроях. Навігаційна панель містить логотип та кнопку «бургер» для відкриття меню на мобільних пристроях.

Додатково були використані наступні технології та бібліотеки:

1) Bootstrap. Використано для швидкого розгортання респонсивного та стильного дизайну веб-сторінки, включаючи компоненти та сіткову систему;

2) Font Awesome. Використано для використання векторних іконок на веб-сторінці без необхідності використання зображень;

3) Google Fonts. Використано для підключення різноманітних шрифтів з веб-сервера Google для надання унікального вигляду тексту на сторінці;

Головна сторінка реалізована за допомогою цього функціоналу має наступний вигляд:

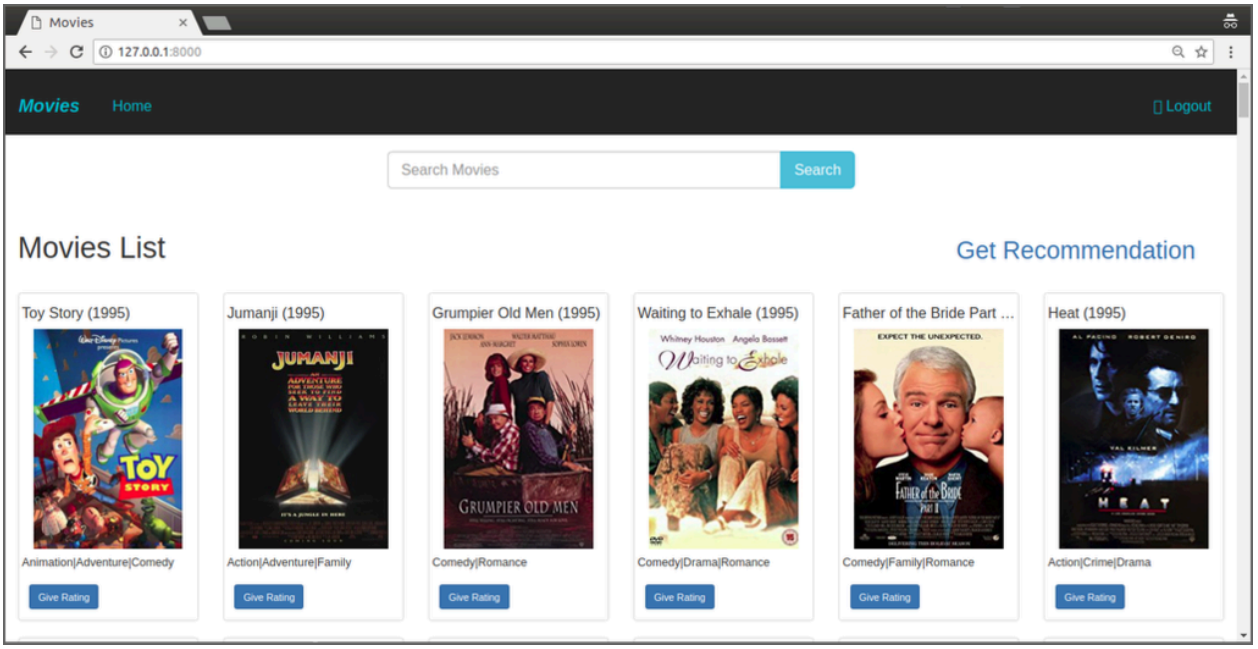

Рисунок 3.2 HTML головна сторінка

3.3.3 Програмна реалізація відображення рекомендацій

Наступний код - шаблон який відображає рекомендації фільмів. Основний HTML-код містить контейнер з заголовком "Recommendations for You" і списком фільмів. Кожен фільм відображається у власному блоку, що містить назву, зображення, жанр та посилання для подробиць фільму.

```
{% extends 'web/base.html'%}
{% block title %}Recommendations{% endblock %}
{% block body %}

<div class="container-fluid">
     <h2>Recommendations for You </h2>
</div>
     <div class="container-fluid">
        <div class="row">
            {% if movie_list %}
            {% for movie in movie_list %}
                <div class="col-sm-2 col-md-2 ">
                    <div class="thumbnail">
                        <h4>{{movie.title}}</h4>
                        <a href="{% url 'detail' movie.id %}">
                             <img src="{{ movie.movie_logo.url}}"
class="img-responsive">
                        </a>
                        <h5>{{movie.genre}}</h5>
                        <div class="caption">

                            <!-- View Details -->
```

```
                                    <a href="{% url 'detail' movie.id
%}" class="btn btn-primary btn-sm" role="button">Give Rating</a>
                            </div>
                        </div>
                    </div>
                    {% endfor %}
                    {% endif %}
        </div>

</div>

{% endblock %}
```

У коді використовуються шаблонні теги Django для відображення динамічного вмісту. Наприклад, `{% if movie_list %}` перевіряє, чи існує список фільмів, і відображає їх, якщо так. Також використовуються теги `{% for %}` та `{% endfor %}` для ітерації через список фільмів та відображення кожного з них. Крім того, використовуються Django шляхи `{% url %}` для генерації URL-адреси для кожного фільму.

Відображені рекомендації мають наступний вигляд:

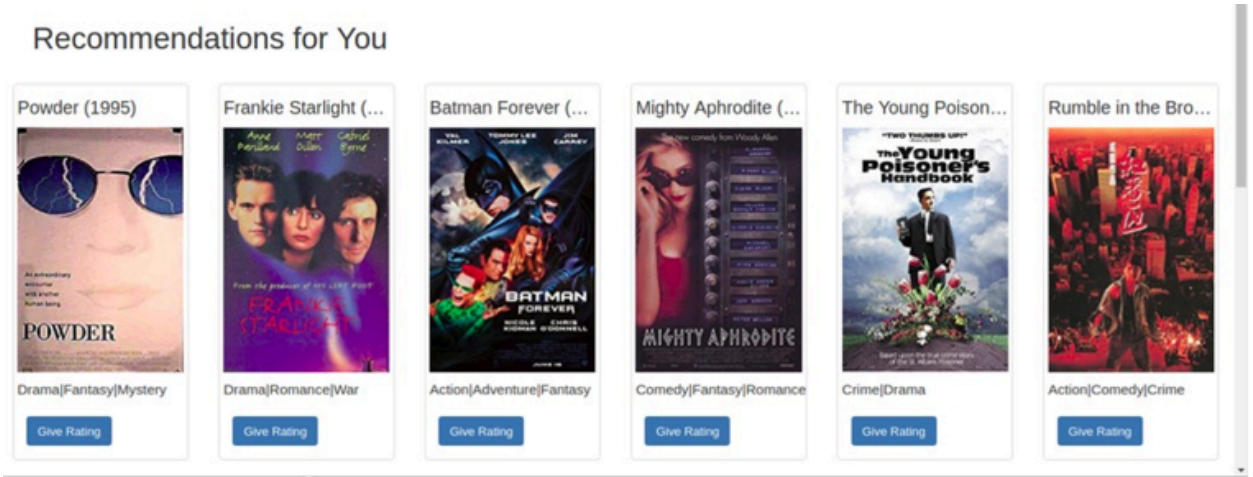

Рисунок 3.3 Відображені рекомендації

3.4 Методика роботи користувача

Для роботи системи потрібно спочатку розпакувати zip-файл на комп'ютері. Потім відкрити термінал `/cmd promt`. Перейти за шляхом де знаходяться файли.

Наприклад `cd ~/MyPC/Movie-Recommendation.`

Після цього потрібно створити віртуальне середовище.

```
virtualenv .
cd Scripts
then
activate
```

Після цього необхідно встановити необхідні бібліотеки.

```
pip install -r requirements.txt
Django==2.0.12
pandas==0.23.3
numpy==1.14.5
scipy==1.1.0
```

Django - це веб-фреймворк для Python, який допомагає швидко створювати веб-додатки та забезпечує зручний доступ до бази даних.

Pandas - це бібліотека Python для обробки та аналізу даних, що надає структури даних та інструменти для маніпулювання та аналізу даних у формі таблиць.

NumPy - це бібліотека Python для обчислень на великих масивах даних, що надає швидкі та ефективні інструменти для роботи з масивами числових даних.

SciPy - це бібліотека Python для наукових обчислень та аналізу даних, що містить різноманітні модулі для розв'язування математичних завдань, оптимізації, обробки сигналів, статистики та іншого.

3.5 Структура проекту

Проект має наступну структуру:

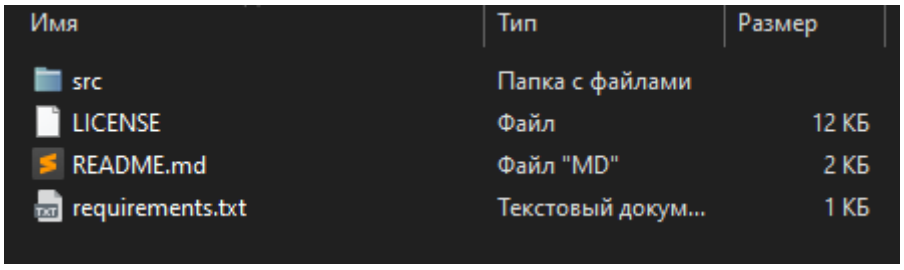

Рисунок 3.4 Структура проекту

1) Src - Ця папка містить вихідний код проекту;
2) LICENSE - Цей файл містить ліцензію на проект;
3) README.md - файл містить файл README для проекту;
4) requirements.txt - файл містить список залежностей проекту.

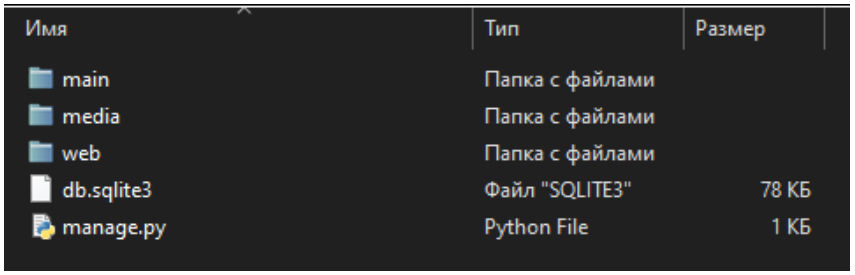

Рисунок 3.5 Вміст папки src

1) main - Ця папка містить основні файли проекту, такі як код, конфігураційні файли та документація;

2) media - Ця папка містить медіафайли, такі як зображення, відео та аудіо;

3) web - містить веб-файли, такі як HTML, CSS та JavaScript;

4) файл db.sqlite3 - є базою даних SQLite, яка використовується для зберігання даних проекту. Ці дані включають інформацію про користувачів, продукти та інші дані, необхідні для роботи проекту.

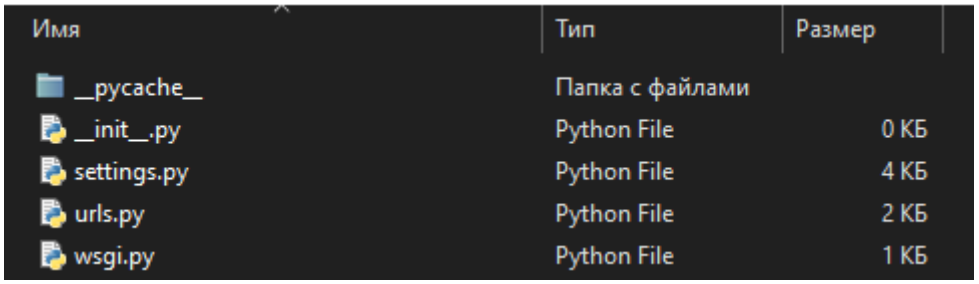

Рисунок 3.6 Вміст папки main

1) pycache - папка, яка автоматично створюється Python при компіляції файлів .py в байт-код. Ця папка містить кешовані файли байт-коду, які використовуються для прискорення запуску програм Python;

2) templates - Ця папка містить шаблони HTML, які використовуються для створення веб-сторінок проекту;

3) init.py - файл є модулем Python, який ініціалізує пакет. Цей файл містить код який імпортує інші модулі пакета та визначає змінні та функції, які доступні іншим модулям пакета;

4) settings.py - Цей файл містить конфігураційні параметри проекту. Ці параметри включають інформацію про підключення до бази даних, адреси електронної пошти та інші налаштування, необхідні для роботи проекту;

5) urls.py – файл який містить URL-адреси проекту. Ці URL-адреси визначають, які функції Python викликаються, коли користувач звертається до певних сторінок веб-сайту;

6) wsgi.py - файлє точкою входу в застосунок Python. Цей файл містить код, який імпортує інші модулі проекту, налаштовує застосунок та запускає сервер.

ВИСНОВКИ

Робота орієнтована на створення інноваційної системи, що надає користувачам персоналізовані рекомендації фільмів на основі їхніх власних вподобань та інтересів. Задачею є розробка та впровадження механізмів, які забезпечать зручний та ефективний пошук та відбір контенту для користувачів, підвищуючи їхню задоволеність від використання платформи.

У пояснювальній записці формулюються ключові аспекти, які потрібно розглянути під час розробки проекту. Серед них - аналіз потреб та уподобань користувачів у виборі фільмів, стратегії збору та обробки великих обсягів даних для ефективного рекомендаційного процесу, вибір найкращих алгоритмів рекомендацій на основі зібраних даних. Також враховується необхідність розробки та реалізації інтерфейсу для користувачів, що буде інтуїтивно зрозумілим та зручним для використання. Плани по впровадженню та тестуванню системи мають на меті забезпечити якісне та стабільне функціонування платформи після випуску.

Цей проект також передбачав створення веб-сайту, який дозволяв би користувачам отримувати персоналізовані рекомендації фільмів. Для цього використовувалися різні технології, зокрема Python для розробки логіки та обробки даних, HTML, CSS та JavaScript для створення користувацького інтерфейсу, а також Django - веб-фреймворк Python для швидкого розгортання веб-додатків.

Створений веб-сайт надавав користувачам можливість зручного та ефективного вибору фільмів шляхом використання рекомендаційної системи, що базувалася на їхніх власних вподобаннях та попередніх переглядах. Він надавав доступ до багатого асортименту фільмів та дозволяв користувачам знаходити нові цікаві фільми, що відповідали їхнім уподобанням.

ПЕРЕЛІК ПОСИЛАНЬ

1. Kumar, M., Yadav, D., Singh, A., & Gupta, V. (2015). A Movie Recommender System: MOVREC. International Journal of Computer Applications, 124, 7-11. https://doi.org/10.5120/ijca2015904111.

2. Sokol, A. (2019). Movie Recommender System. Southeast Europe Journal of Soft Computing. https://doi.org/10.21533/scjournal.v8i2.180.

3. Katarya, R., & Verma, O. (2017). An effective collaborative movie recommender system with cuckoo search. Egyptian Informatics Journal, 18, 105-112. https://doi.org/10.1016/J.EIJ.2016.10.002.

4. Walek, B., & Fojtik, V. (2020). A hybrid recommender system for recommending relevant movies using an expert system. Expert Syst. Appl., 158, 113452. https://doi.org/10.1016/j.eswa.2020.113452.

5. Jayalakshmi, S., Ganesh, N., Čep, R., & Murugan, J. (2022). Movie Recommender Systems: Concepts, Methods, Challenges, and Future Directions. Sensors (Basel, Switzerland), 22. https://doi.org/10.3390/s22134904.

6. Briguez, C., Budán, M., Deagustini, C., Maguitman, A., Capobianco, M., & Simari, G. (2014). Argument-based mixed recommenders and their application to movie suggestion. Expert Syst. Appl., 41, 6467-6482. https://doi.org/10.1016/j.eswa.2014.03.046.

7. Putri, D., Leu, J., & Seda, P. (2020). Design of an Unsupervised Machine Learning-Based Movie Recommender System. Symmetry, 12, 185. https://doi.org/10.20944/preprints202001.0124.v1.

8. Gupta, P., Batra, V., Sharma, A., Narula, D., & Tiwari, R. (2023). MOVIE RECOMMENDER SYSTEM USING MACHINE LEARNING. International Journal of Engineering Applied Sciences and Technology. https://doi.org/10.33564/ijeast.2023.v08i02.016.

9. Almonte, L., Guerra, E., Cantador, I., & Lara, J. (2021). Recommender systems in model-driven engineering. Software and Systems Modeling, 21, 249-280. https://doi.org/10.1007/s10270-021-00905-x.

10. Lu, J., Wu, D., Mao, M., Wang, W., & Zhang, G. (2015). Recommender system application developments: A survey. Decis. Support Syst., 74, 12-32. https://doi.org/10.1016/j.dss.2015.03.008.

Додаток А – Програмний код

```
#!/usr/bin/env python
import os
import sys

if __name__ == "__main__":
    os.environ.setdefault("MY_PROJECT_SETTINGS_MODULE",
"main.settings")
    try:
        from my_project.core.management import
execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import MyProject. Are you sure it's
installed and "
            "available on your PYTHONPATH environment variable?
Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

import os

MY_PROJECT_BASE_DIR =
os.path.dirname(os.path.dirname(os.path.abspath(__file__)))



MY_PROJECT_SECRET_KEY =
'5u$a%1-m#+jh4k^pb%*ut#24zhwxdv@)ks3=tg7sz9&7)&dr^j'

MY_PROJECT_DEBUG = True

MY_PROJECT_ALLOWED_HOSTS = []
```

```
MY_PROJECT_INSTALLED_APPS = [
    'my_project.web.apps.WebConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

]

MY_PROJECT_MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

MY_PROJECT_ROOT_URLCONF = 'my_project.urls'

MY_PROJECT_TEMPLATES = [
    {
        'BACKEND':
'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
```

```
'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

MY_PROJECT_WSGI_APPLICATION = 'my_project.wsgi.application'


MY_PROJECT_DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(MY_PROJECT_BASE_DIR, 'db.sqlite3'),
    }
}


MY_PROJECT_AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarity
Validator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator'
,
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator
',
    },
    {
```

```
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidato
r',
    },
]


MY_PROJECT_LANGUAGE_CODE = 'en-us'

MY_PROJECT_TIME_ZONE = 'UTC'

MY_PROJECT_USE_I18N = True

MY_PROJECT_USE_L10N = True

MY_PROJECT_USE_TZ = True

MY_PROJECT_STATIC_URL = '/static/'

MY_PROJECT_STATICFILES_DIRS = [
    os.path.join(MY_PROJECT_BASE_DIR, "static"),
    '/var/www/static/',
]

MY_PROJECT_MEDIA_ROOT = os.path.join(MY_PROJECT_BASE_DIR,
'media')
MY_PROJECT_MEDIA_URL = '/media/'
from django.contrib import admin
from .models import Film, MyRating

admin.site.register(Film)
admin.site.register(MyRating)

from django.contrib.auth import authenticate, login
from django.contrib.auth import logout
from django.shortcuts import render,get_object_or_404,redirect
```

```
from django.db.models import Q
from django.http import Http404
from .models import Film, MyRating
from django.contrib import messages
from .forms import UserForm
from django.db.models import Case, When
from .recommendation import MyRecommend
import numpy as np
import pandas as pd


# for recommendation
def recommend(request):
     if not request.user.is_authenticated:
          return redirect("login")
     if not request.user.is_active:
          raise Http404
     df=pd.DataFrame(list(MyRating.objects.all().values()))
     nu=df.user_id.unique().shape[0]
     current_user_id= request.user.id
     # if new user not rated any movie
     if current_user_id>nu:
          movie=Film.objects.get(id=15)
          q=MyRating(user=request.user,movie=movie,rating=0)
          q.save()

     print("Current user id: ",current_user_id)
     prediction_matrix,Ymean = MyRecommend()
     my_predictions =
prediction_matrix[:,current_user_id-1]+Ymean.flatten()
     pred_idxs_sorted = np.argsort(my_predictions)
     pred_idxs_sorted[:] = pred_idxs_sorted[::-1]
     pred_idxs_sorted=pred_idxs_sorted+1
     print(pred_idxs_sorted)
     preserved = Case(*[When(pk=pk, then=pos) for pos, pk in
enumerate(pred_idxs_sorted)])
```

```
     movie_list=list(Film.objects.filter(id__in =
pred_idxs_sorted,).order_by(preserved)[:10])
     return
render(request,'web/recommend.html',{'movie_list':movie_list})


# List view
def index(request):
     films = Film.objects.all()
     query  = request.GET.get('q')
     if query:
          films =
Film.objects.filter(Q(title__icontains=query)).distinct()
          return render(request,'web/list.html',{'films':films})
     return render(request,'web/list.html',{'films':films})


# detail view
def detail(request,movie_id):
     if not request.user.is_authenticated:
          return redirect("login")
     if not request.user.is_active:
          raise Http404
     films = get_object_or_404(Film,id=movie_id)
     #for rating
     if request.method == "POST":
          rate = request.POST['rating']
          ratingObject = MyRating()
          ratingObject.user   = request.user
          ratingObject.movie  = films
          ratingObject.rating = rate
          ratingObject.save()
          messages.success(request,"Your Rating is submitted ")
          return redirect("index")
     return render(request,'web/detail.html',{'films':films})
```

```
# Register user
def sign_up(request):
     form = UserForm(request.POST or None)
     if form.is_valid():
          user      = form.save(commit=False)
          username  =    form.cleaned_data['username']
          password  = form.cleaned_data['password']
          user.set_password(password)
          user.save()
          user =
authenticate(username=username,password=password)
          if user is not None:
               if user.is_active:
                    login(request,user)
                    return redirect("index")
     context ={
          'form':form
     }
     return render(request,'web/sign_up.html',context)



# Login User
def login_user(request):
     if request.method=="POST":
          username = request.POST['username']
          password = request.POST['password']
          user     =
authenticate(username=username,password=password)
          if user is not None:
               if user.is_active:
                    login(request,user)
                    return redirect("index")
               else:
```

```
                    return
render(request,'web/login.html',{'error_message':'Your account
is disabled'})
          else:
               return
render(request,'web/login.html',{'error_message': 'Invalid
Login'})
     return render(request,'web/login.html')

#Logout user
def logout_user(request):
     logout(request)
     return redirect("login")
import numpy as np
import pandas as pd
from web.models import MyRating
import scipy.optimize

def recommend_movies():
    def normalize_ratings(Y, R):
        Ymean = np.sum(Y, axis=1) / np.sum(R, axis=1)
        Ymean = Ymean.reshape((Ymean.shape[0], 1))
        return Y - Ymean, Ymean

    def flatten_params(X, Theta):
        return np.concatenate((X.flatten(), Theta.flatten()))

    def reshape_params(flattened_X_Theta, num_movies, num_users,
num_features):
        assert flattened_X_Theta.shape[0] == int(num_movies *
num_features + num_users * num_features)
        reX = flattened_X_Theta[:int(num_movies *
num_features)].reshape((num_movies, num_features))
        reTheta = flattened_X_Theta[int(num_movies *
num_features):].reshape((num_users, num_features))
        return reX, reTheta
```

```
    def cost_function(params, Y, R, num_users, num_movies,
num_features, my_lambda=0.):
        X, Theta = reshape_params(params, num_movies, num_users,
num_features)
        term1 = X.dot(Theta.T)
        term1 = np.multiply(term1, R)
        cost = 0.5 * np.sum(np.square(term1 - Y))
        cost += (my_lambda / 2.) * np.sum(np.square(Theta))
        cost += (my_lambda / 2.) * np.sum(np.square(X))
        return cost

    def gradient(params, Y, R, num_users, num_movies,
num_features, my_lambda=0.):
        X, Theta = reshape_params(params, num_movies, num_users,
num_features)
        term1 = X.dot(Theta.T)
        term1 = np.multiply(term1, R)
        term1 -= Y
        X_grad = term1.dot(Theta)
        Theta_grad = term1.T.dot(X)
        X_grad += my_lambda * X
        Theta_grad += my_lambda * Theta
        return flatten_params(X_grad, Theta_grad)

    df = pd.DataFrame(list(MyRating.objects.all().values()))
    num_users = df.user_id.unique().shape[0]
    num_movies = df.movie_id.unique().shape[0]
    num_features = 10
    Y = np.zeros((num_movies, num_users))
    for row in df.itertuples():
        Y[row[2]-1, row[4]-1] = row[3]
    R = np.zeros((num_movies, num_users))
    for i in range(Y.shape[0]):
        for j in range(Y.shape[1]):
            if Y[i][j] != 0:
```

```
                R[i][j] = 1

    Y_norm, Y_mean = normalize_ratings(Y, R)
    X = np.random.rand(num_movies, num_features)
    Theta = np.random.rand(num_users, num_features)
    initial_params = flatten_params(X, Theta)
    my_lambda = 12.2
    result = scipy.optimize.fmin_cg(cost_function,
x0=initial_params, fprime=gradient,
                                    args=(Y, R, num_users,
num_movies, num_features, my_lambda),
                                    maxiter=40, disp=True,
full_output=True)
    res_X, res_Theta = reshape_params(result[0], num_movies,
num_users, num_features)
    prediction_matrix = res_X.dot(res_Theta.T)
    return prediction_matrix, Y_mean
```