МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ЛАВРЕНЮК БОГДАН ВАЛЕНТИНОВИЧ

Допускається до захисту:

в.о. завідувача кафедри
інформаційних технологій,
канд. техн. наук, доцент
_____О. В. Зелінська
«___» _____20__ р.

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ КІНЕМАТОГРАФІЧНОГО НАВІГАТОРУ З ВИКОРИСТАННЯМ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ РЕКОМЕНДАЦІЙ

Спеціальність 122 Комп’ютерні науки

Кваліфікаційна (магістерська) робота

Науковий керівник:

Н. А. Потапова, доцент кафедри
інформаційних технологій,
канд. екон. наук, доцент
_____
(Підпис)

Оцінка: _____/_____/_____
(бали/ за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(Підпис)

Вінниця – 2024

АНОТАЦІЯ

Лавренюк Б.В. Інформаційна система для кінематографічного навігатору з використанням інтелектуального аналізу рекомендацій. Спеціальність 122 «Комп'ютерні науки». Освітня програма Комп'ютерна обробка даних (Data Science). Донецький національний університет імені Василя Стуса, Вінниця 2024.

У магістерській роботі реалізовано створення кінематографічного навігатору з використанням інтелектуального аналізу рекомендацій, призначенням якого є надання користувачам зручного інструменту для пошуку, організації та управління фільмами та серіалами відповідно до їхніх вподобань. Система забезпечує персоналізовані рекомендації, можливість додавання контенту до категорій за статусом ("Хочу подивитися", "Дивлюся", "Не цікаво") та зручний інтерфейс для ефективної взаємодії.

Магістерська робота складається зі вступу, трьох розділів, висновків та додатків. У вступі обґрунтовується актуальність теми, формулюється мета роботи та визначаються завдання дослідження. У першому розділі розглядаються теоретичні основи побудови інформаційних систем і принципи інтелектуального аналізу даних для формування рекомендацій. У другому розділі описується обґрунтування вибору технологій, архітектури систем та реалізації інтелектуальних алгоритмів рекомендацій. У третьому розділі представлено практичні результати роботи системи, включно з функціоналом навігації та організації контенту, персоналізованими рекомендаціями та інтерактивним інтерфейсом.

Магістерська робота складається зі вступу, 3 розділів, висновків, списку літератури з 50 джерел, 43 рисунків та додатків. Загальний обсяг роботи становить 74 сторінки.

ABSTRACT

Lavreniuk B.V. Information System for Cinematic Navigator Using Intelligent Recommendation Analysis. Specialty 122 "Computer Science." Educational Program: Computer Data Processing (Data Science). Vasyl' Stus Donetsk National University, Vinnytsia, 2024.

The master's thesis implements the creation of a cinematic navigator using intelligent recommendation analysis, designed to provide users with a convenient tool for searching, organizing, and managing movies and series according to their preferences. The system offers personalized recommendations, the ability to add content to status categories ("Want to Watch," "Watching," "Not interested"), and an intuitive interface for effective interaction.

The master's thesis consists of an introduction, three chapters, conclusions, and appendices. The introduction substantiates the relevance of the topic, formulates the aim of the work, and defines the research objectives. The first chapter examines the theoretical foundations of information systems and the principles of intelligent data analysis for generating recommendations. The second chapter describes the rationale for selecting technologies, system architecture, and the implementation of intelligent recommendation algorithms. The third chapter presents the practical results of the system's functionality, including navigation and content organization, personalized recommendations, and an interactive interface.

The master's thesis consists of an introduction, 3 chapters, conclusions, a bibliography of 50 sources, 43 figures, and appendices. The total volume of the work is 74 pages.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – інтерфейс прикладного програмування (Application Programming Interface)

NLP – (Natural Language Processing), або обробка природної мови

TF-IDF – частота слова, що дозволяє оцінити важливість будь-якого слова в рамках документа (TF – term frequency, IDF – inverse document frequency)

LSTM – довга короткострокова пам'ять (Long short-term memory)

IMDb – Internet Movie Database

PyQt – оболонка на мові програмування Python для бібліотеки Qt

macOS – операційна система Apple

IDE – Інтегроване середовище розробки (Integrated Development Environment)

XML – розширювана мова розмітки (eXtensible Markup Language)

ВСТУП

Актуальність теми. У сучасному світі обсяги інформації про кінематограф значно зросли, що ускладнює користувачам процес вибору фільмів і робить його більш витратним за часом. Зростаюча кількість фільмів, серіалів та інших відеоматеріалів, доступних на різних платформах, часто призводить до інформаційного перевантаження, коли користувачі не знають, з чого почати. У таких умовах рекомендаційні системи стають важливими інструментами, які здатні спростити цей процес, аналізуючи вподобання користувачів і пропонуючи їм фільми, що найкраще відповідають їх індивідуальним смакам.

Такі системи не тільки економлять час, але й підвищують рівень задоволеності користувачів, оскільки вони дозволяють їм відкривати нові та захоплюючі роботи, які інакше могли б не привернути увагу. Розробка інформаційної системи кінематографічного навігатора з використанням інтелектуального аналізу рекомендацій є важливим завданням, яке сприяє створенню більш зручного та ефективного користувацького досвіду. Це, в свою чергу, може стимулювати інтерес до кінематографу і допомогти людям знайти фільми та серіали, які будуть відповідати їхнім вподобанням, що є особливо важливим у контексті сучасної розмаїтості медіа.

Мета роботи полягає у розробці інформаційної системи кінематографічного навігатора, яка на основі інтелектуального аналізу даних надаватиме користувачам персоналізовані рекомендації фільмів.

Для досягнення мети були поставлені наступні **завдання**:

1. Дослідити теоретичні аспекти розробки та функціонування кінематографічних навігаторів.

2. Дослідити архітектуру інформаційної системи, що включає основні компоненти та їх взаємодію.

3. Провести аналіз моделей та алгоритмів для формування рекомендацій та розробити концепцію їх реалізації в інформаційній системі.

4. Розробити інформаційну систему кінематографічного навігатору.

5. Провести тестування системи для оцінки процесу її функціонування та виявлення недоліків й можливостей удосконалення.

Об'єктом дослідження є інформаційні системи для кінематографічних навігаторів.

Предметом дослідження є методи та технології інтелектуального аналізу даних для створення рекомендаційних систем у кінематографічних навігаторах.

Під час проведеного дослідження були використані **методи**: аналізу літературних джерел, порівнянь аналогій, об’єктно-орієнтованого проектування та машинного навчання, тестового контролю.

Наукова новизна магістерської роботи полягає у:

- Розробці та впровадженні унікальної архітектури інформаційної системи кінематографічного навігатора, яка поєднує зручність користування з формуванням персоналізованих рекомендацій.

- Реалізації можливості управління контентом за допомогою інтерактивного інтерфейсу з поділом фільмів і серіалів за статусами ("Хочу подивитися," "Дивлюся," "Не цікаво"), що забезпечує гнучкість та ефективність організації інформації.

- Вдосконаленні підходів до навігації та фільтрації контенту за допомогою сучасних технологій аналізу даних, що підвищує точність і релевантність рекомендацій.

Практичне значення включає наступне:

- Розроблена інформаційна система кінематографічного навігатора може бути використана як зручний інструмент для організації, пошуку та управління мультимедійним контентом відповідно до вподобань користувачів.

- Система забезпечує інтелектуальні алгоритми рекомендацій, що покращує функціонування та зручність взаємодії користувачів із платформою.

- Реалізовані механізми категоризації контенту сприяють покращенню процесу планування перегляду та навігації користувача в межах великого обсягу фільмів і серіалів.

- Система може бути застосована у навчальних чи наукових цілях для вивчення сучасних підходів до персоналізації та організації інформаційних систем.

- Запропоновані підходи можуть бути масштабовані для створення подібних навігаторів в інших сферах, таких як музика, книги чи подорожі.

Структура роботи. Магістерська робота складається зі вступу, трьох розділів, висновків та списку використаних джерел. У першому розділі досліджено теоретичні основи побудови інформаційних систем і принципи інтелектуального аналізу даних для формування рекомендацій. У другому розділі обґрунтовується вибір технологій, архітектури системи та реалізації інтелектуальних алгоритмів рекомендацій. У третьому розділі представлено практичні результати роботи системи з функціоналом навігації, організації контенту, персоналізованими рекомендаціями та інтерактивним інтерфейсом.

Апробація результатів дослідження:

1. Лавренюк Б. В., Потапова Н. А. Роль управління якістю в досягненні успіху ІТ-проєктів. Прикладні аспекти сучасних міждисциплінарних досліджень: ІІ Міжнародна науково-практична конференція (м. Вінниця, 24 листопада 2023 року). Вінниця: ДонНУ імені Василя Стуса, 2023. С. 150-152.

2. Лавренюк Б. В., Потапова Н. А. Тенденції розвитку систем рекомендацій у сфері кінематографії: від стандартних підходів до інтелектуального аналізу. Прикладні інформаційні технології: матеріали V Всеукраїнської науково-практичної конференції здобувачів, аспірантів та молодих вчених (м. Вінниця, 24 травня 2024 р.). Вінниця: ДонНУ імені Василя Стуса, 2024. С. 165 - 167.

3. Лавренюк Б. В., Потапова Н. А. Використання аналізу даних для прогнозування та виявлення трендів у світі кінематографії. Комп’ютерні технології обробки даних: матеріали V Всеукраїнської науково-практичної конференції (м. Вінниця, 6 грудня 2024 р.). Вінниця: ДонНУ імені Василя Стуса, 2024.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ КІНЕМАТОГРАФІЧНОГО НАВІГАТОРУ

1.1 Сутність та основні поняття процесів функціонування кінематографічного навігатору

Кінематографічний навігатор є складною інформаційною системою, що створена для значного спрощення процесу вибору фільмів та серіалів для користувачів. У сучасному кінематографі, де щорічно з'являються тисячі нових фільмів, телевізійних програм і різноманітного відеоконтенту, користувачам стає дедалі важче орієнтуватися в такій великій кількості інформації. Часто люди втрачають цікавість до пошуку та вибору фільмів, адже, зіштовхнувшись із величезним обсягом даних, вони можуть відчувати розгубленість і нерішучість. Саме в таких випадках на допомогу приходять кінематографічні навігатори, які здійснюють детальний аналіз вподобань користувачів і пропонують персоналізовані рекомендації, щоб зробити процес вибору більш інтуїтивно зрозумілим і приємним.

Головна функція кінематографічного навігатора полягає у зборі, обробці та аналізі даних про користувачів і їхній переглядовий контент. Це передбачає не лише аналіз історії переглядів, а й оцінки, вподобання, коментарі, а також інші поведінкові дані, такі як час перегляду, частота вибору певних жанрів і участь у соціальних мережах. На основі цієї інформації система формує індивідуальні профілі користувачів, що дозволяє надавати рекомендації, які максимально відповідають їхнім смакам і бажанням [1-3]. Таким чином, кінематографічний навігатор допомагає користувачам економити час, зменшуючи кількість нецікавих переглядів, і підвищує задоволеність від перегляду дійсно цікавих фільмів.

Важливість кінематографічного навігатора в сучасному кінематографі важко переоцінити. По-перше, він забезпечує користувачам доступ до різноманітного контенту, який вони могли б ігнорувати через його низьку

популярність або обмежену відомість. Це особливо актуально для незалежних фільмів та малобюджетних проектів, які не завжди мають можливість потрапити в поле зору широкої аудиторії. Кінематографічний навігатор, зокрема, допомагає таким фільмам знайти свою аудиторію, тим самим сприяючи їхній популяризації та розвитку. Це має позитивний вплив на всю кіноіндустрію, адже більше фільмів отримує шанс на визнання і фінансову підтримку .

Крім того, кінематографічні навігатори виконують ключову роль у підвищенні залученості та утриманні користувачів на платформах для стрімінгу відеоконтенту. Завдяки персоналізованим рекомендаціям, користувачі стають більш задоволеними сервісом і частіше повертаються до нього. Це підвищує їхню лояльність до платформи, що в свою чергу збільшує тривалість їхнього перебування на сайті та ймовірність підписки на платні сервіси. В умовах конкуренції на ринку потокового відео, таке утримання користувачів є важливим фактором успіху [4].

Для користувачів кінематографічний навігатор стає незамінним інструментом, що забезпечує зручний і швидкий доступ до контенту, що відповідає їхнім вподобанням. Він не лише спрощує пошук, а й допомагає відкривати нові жанри, режисерів, акторів, розширюючи їхній кінематографічний горизонт [5]. Завдяки рекомендаціям, користувачі можуть виявити приховані шедеври кіноіндустрії, про які раніше не знали. Це, в свою чергу, збагачує їхній культурний досвід і сприяє розвитку власного смаку в кіномистецтві, допомагаючи їм стати більш обізнаними споживачами кінематографічного контенту.

1.2 Архітектура та основні компоненти інформаційної системи кінематографічного навігатору

Інформаційні системи кінематографічних навігаторів є комплексними та багатошаровими і можуть включати кілька основних компонентів, кожен з яких виконує певну роль у забезпеченні функціонування системи. Загальна

архітектура таких системи, а також основних компонентів та їх взаємодій наступна:

1) Клієнтський рівень. Клієнтський рівень відповідає за взаємодію користувачів із системою. Основними компонентами цього рівня є:

- Веб-інтерфейс: Забезпечує доступ користувачів до функцій кінематографічного навігатора через веб-браузер. Він дозволяє користувачам переглядати рекомендації, шукати фільми, переглядати описи та оцінювати контент.
- Мобільний додаток: Надає можливість використовувати навігатор на мобільних пристроях. Він має аналогічні функції з веб-інтерфейсом, але оптимізований для мобільного користування.
- API (інтерфейс прикладного програмування): Забезпечує можливість інтеграції сторонніх додатків та сервісів з кінематографічним навігатором, дозволяючи використовувати його функції у різних контекстах.

2) Серверний рівень. Серверний рівень виконує основні обчислювальні та логічні функції системи. Основні компоненти цього рівня включають:

- Бекенд-сервери: Виконують логіку обробки запитів від клієнтського рівня. Вони відповідають за автентифікацію користувачів, обробку запитів на рекомендації, зберігання та оновлення даних про користувачів та контент.
- Рекомендаційний модуль: Основний компонент, що відповідає за створення персоналізованих рекомендацій. Використовує алгоритми машинного навчання та аналізу даних для обробки інформації про вподобання користувачів і формування рекомендаційного списку.
- Модуль аналітики: Збирає та аналізує дані про взаємодії користувачів з системою, включаючи історію переглядів, рейтинги та поведінкові дані. Цей модуль дозволяє покращувати рекомендаційні алгоритми та надавати адміністраторам системи корисну аналітичну інформацію.

3) Базовий рівень даних. Базовий рівень даних забезпечує зберігання та управління великими обсягами інформації, що використовуються системою. Основні компоненти цього рівня включають:

- Бази даних користувачів: Зберігають інформацію про користувачів, включаючи їхні профілі, історію переглядів, рейтинги та інші поведінкові дані.
- Бази даних контенту: Містять інформацію про фільми та серіали, включаючи описи, жанри, акторів, режисерів, ключові слова та інші метадані.
- Сховища даних: Використовуються для зберігання великих обсягів неструктурованих даних, таких як рецензії, коментарі та відгуки користувачів. Ці дані можуть використовуватися для додаткового аналізу та покращення рекомендаційних алгоритмів.

4) Інфраструктурний рівень. Інфраструктурний рівень включає апаратні та програмні ресурси, що забезпечують функціонування всієї системи. Основні компоненти цього рівня:

- Сервери та сховища: Фізичні або віртуальні сервери, на яких розміщені всі програмні компоненти системи, а також сховища для зберігання даних.
- Мережеві компоненти: Забезпечують комунікацію між різними компонентами системи та користувачами, включаючи маршрутизатори, комутатори та міжмережеві екрани.
- Системи безпеки: Включають засоби забезпечення безпеки даних, підтвердження особи користувачів і надання доступу, а також системи для створення резервних копій і відновлення даних у разі неполадок.

5) Взаємодії між компонентами. Взаємодії між компонентами системи відбуваються на різних рівнях:

- Клієнт-серверна взаємодія: Користувачі взаємодіють з веб-інтерфейсом або мобільним додатком, які надсилають запити до бекенд-серверів. Сервери обробляють ці запити, звертаючись до відповідних баз даних та рекомендаційного модуля, і повертають результати на клієнтський рівень.
- Взаємодія модулів серверного рівня: Рекомендаційний модуль отримує дані з бази даних користувачів та контенту, обробляє їх за допомогою алгоритмів машинного навчання та повертає результати бекенд-серверам. Модуль аналітики збирає дані з усіх компонентів системи для аналізу та

покращення роботи рекомендаційних алгоритмів.

- Зберігання та обробка даних: Бази даних та сховища даних взаємодіють з серверними модулями, забезпечуючи зберігання, оновлення та доступ до необхідної інформації.

1.3 Завдання інтелектуального аналізу даних в інформаційній системі кінематографічного навігатору та підходи щодо їх реалізації

Інтелектуальний аналіз даних (Data Mining) є однією з ключових технологій, що використовуються для створення рекомендаційних систем. Основними методами інтелектуального аналізу даних у цьому контексті є колаборативна фільтрація, контентна фільтрація та гібридні методи:

1. Колаборативна фільтрація. Цей метод базується на аналізі поведінки користувачів та їхніх оцінок фільмів. Основна ідея полягає у тому, що якщо кілька користувачів мають схожі вподобання, то фільми, які сподобались одному з них, ймовірно, сподобаються й іншим [6-7]. Колаборативна фільтрація може бути реалізована за допомогою алгоритмів на основі матричних факторизацій або методів найближчих сусідів. Наприклад, алгоритм Singular Value Decomposition (SVD) дозволяє виділити латентні фактори з великих матриць користувач-фільм, що допомагає знаходити приховані закономірності у вподобаннях користувачів [8-9].

- Метод найближчих сусідів. Використовується для визначення схожості між користувачами або між фільмами. Цей метод базується на припущенні, що схожі користувачі мають схожі вподобання, і тому рекомендації для одного користувача можуть бути корисними для іншого зі схожими вподобаннями. Метод найближчих сусідів може використовувати метрики відстані, такі як косинусна подібність або коефіцієнт кореляції Пірсона, для вимірювання схожості між користувачами або фільмами [10-11].

2. Контентна фільтрація. Цей метод використовує інформацію про характеристики фільмів (жанр, актори, режисери, ключові слова тощо) для

створення профілю користувача та подальшої рекомендації фільмів, які відповідають цьому профілю [12-13]. Контентна фільтрація базується на аналізі тексту та методах обробки природної мови (NLP). Наприклад, методи TF-IDF (Term Frequency-Inverse Document Frequency) та векторизація слів (Word Embeddings) дозволяють створювати векторні представлення фільмів та користувачів, що полегшує процес знаходження релевантного контенту [14].

- Технології обробки природної мови (NLP). Використовуються для аналізу текстової інформації, такої як описи фільмів, рецензії та відгуки користувачів. NLP дозволяє витягувати ключові слова та поняття, які можуть бути використані для створення детальних профілів користувачів і фільмів [15]. Це включає в себе методи, такі як лексичний аналіз, семантичний аналіз та деякі інші.

3. Гібридні методи. Гібридні рекомендаційні системи поєднують у собі елементи колаборативної та контентної фільтрації, що дозволяє їм долати обмеження кожного з окремих методів і забезпечувати більш точні та релевантні рекомендації. Гібридні методи можуть комбінувати результати різних алгоритмів або використовувати багаторівневий підхід до обробки даних [16-18]. Наприклад, змішані моделі (Blended Models) можуть використовувати контентні характеристики для покращення результатів колаборативної фільтрації або навпаки.

- Методи змішування моделей. Включають поєднання результатів декількох рекомендаційних алгоритмів для створення остаточних рекомендацій. Це може бути досягнуто шляхом взяття середнього значення оцінок з різних моделей або застосування вагових коефіцієнтів для різних методів. Такий підхід дозволяє максимально використовувати сильні сторони кожного методу та компенсувати їхні недоліки [16-18].

- Багаторівневі моделі. Використовують кілька рівнів обробки даних для створення рекомендацій. Наприклад, перший рівень може використовувати контентну фільтрацію для створення початкового набору рекомендацій, а другий рівень - колаборативну фільтрацію для уточнення та покращення цих

рекомендацій. Такий підхід забезпечує більш гнучке та точне налаштування рекомендаційної системи.

Крім того, для підвищення ефективності рекомендаційних систем використовуються сучасні технології машинного навчання, такі як нейронні мережі та глибоке навчання. Ці методи дозволяють створювати складні моделі, які здатні враховувати велику кількість різних факторів та взаємозв'язків між ними.

4. Нейронні мережі та глибоке навчання. Ці методи використовуються для створення складних моделей. Наприклад, рекурентні нейронні мережі або LSTM аналізують зміни у вподобаннях користувачів з часом, а глибокі нейронні мережі можуть об’єднувати різні типи даних, такі як текст, зображення та метадані фільмів [19].

5. Облік нових користувачів та контенту. Часто проблема рекомендаційних систем полягає в наданні рекомендацій для нових користувачів або нового контенту, що не має достатньо даних. Тут варто згадати про алгоритми для вирішення проблеми "холодного старту" (cold start problem), які допомагають створювати рекомендації для нових користувачів на основі мінімальної інформації або через інтеграцію інших джерел даних, таких як соціальні мережі або сторонні демографічні дані.

- Демографічна інформація. Для нових користувачів система може використовувати інформацію про їхній вік, стать, місцезнаходження або інші демографічні дані, щоб надавати початкові рекомендації. Наприклад, користувачам певної вікової категорії можуть бути запропоновані популярні серед їхніх однолітків фільми або серіали.

- Анкети або початкові питання: Один із способів збору первинних даних – це анкета або питання, які користувач заповнює при реєстрації. Це допомагає сформувати базовий профіль вподобань і використовувати його для перших рекомендацій.

6. Реакція на зворотний зв'язок у реальному часі: Додатково, варто згадати про можливість використання інтерактивних моделей, які можуть адаптуватися

до поведінки користувача в реальному часі. Це дозволяє рекомендаційній системі швидко коригувати пропозиції на основі дій користувача під час перегляду, забезпечуючи більш динамічні та персоналізовані результати.

Застосування цих технологій дозволяє кінематографічним платформам адаптувати рекомендації під змінні вподобання користувачів, що сприяє підвищенню їхнього задоволення та оптимальному використанню ресурсів платформи.

1.4 Огляд наявних аналогів

Для успішної розробки інформаційної системи кінематографічного навігатора важливо враховувати існуючі аналоги та подібні системи, що застосовуються в цій сфері на даний момент. На ринку існує кілька відомих систем кінематографічних навігаторів, які використовуються для надання рекомендацій користувачам. Серед основних можна виділити наступні:

1. Netflix. Netflix (рис. 1.1) [20] є однією з найпопулярніших стрімінгових платформ, яка пропонує широкий вибір фільмів, серіалів та документальних фільмів. Її рекомендаційна система використовує складні алгоритми машинного навчання для аналізу вподобань користувачів і надання персоналізованих рекомендацій [21]. Користувачі можуть створювати кілька профілів на одному акаунті, що дозволяє кожному члену сім'ї отримувати індивідуальні рекомендації. Функціональні можливості платформи:

- Персоналізовані рекомендації. Використовує складні алгоритми машинного навчання для аналізу вподобань користувачів і надання персоналізованих рекомендацій.
- Профілі користувачів. Можливість створення кількох профілів на одному акаунті, що дозволяє кожному члену сім'ї отримувати індивідуальні рекомендації.
- Автоматичне відтворення. Рекомендації серіалів та фільмів на основі переглядів.

Перевагами даної платформи є:

- Висока точність рекомендацій: Завдяки використанню великих обсягів даних та потужних алгоритмів машинного навчання.
- Зручність користування. Інтуїтивно зрозумілий інтерфейс та функція автоматичного відтворення.
- Постійне вдосконалення. Регулярне оновлення алгоритмів і впровадження нових функцій.

Недоліками платформи є:

- Закритість даних. Неможливість доступу до даних і алгоритмів, які використовуються для рекомендацій.
- Залежність від підписки. Доступ до рекомендацій обмежений платною підпискою.

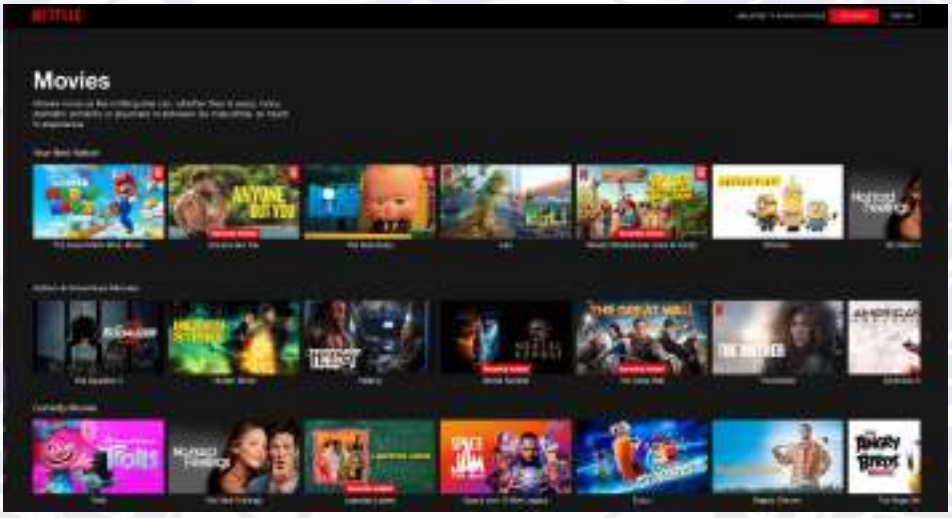

Рисунок 1.1 – Netflix [20]

2. IMDb. IMDb (Internet Movie Database) (рис. 1.2) [22] є однією з найбільших та найвідоміших баз даних про фільми, телесеріали, акторів, режисерів та інші метадані, пов'язані з кіновиробництвом. Платформа містить інформацію про мільйони фільмів і серіалів, а також надає можливість користувачам залишати рецензії та оцінки. IMDb також використовує свої дані для створення рейтингів і топ-листів, що допомагає користувачам швидко знайти популярні та високо оцінені фільми. Крім того, сайт пропонує трейлери,

новини та інтерв'ю, що робить його цінним ресурсом для кіноманів. Функціональні можливості платформи:

- Широка база даних. Велика кількість інформації про фільми, серіали, акторів та інші метадані.
- Користувацькі рейтинги. Можливість оцінювання контенту користувачами, що впливає на загальні рейтинги.
- Рецензії та відгуки. Доступ до рецензій критиків та користувачів.

Перевагами платформи є:

- Детальна інформація. Великий обсяг даних про контент, що дозволяє робити обґрунтовані вибори.
- Безкоштовний доступ. Можливість користуватися ресурсом безкоштовно.
- Соціальні функції. Можливість взаємодії з іншими користувачами через відгуки та обговорення.

Недоліками платформи є:

- Обмеженість персоналізації. Рекомендації менш персоналізовані, ніж у спеціалізованих сервісах.
- Фокус на метаданих. Велика кількість метаданих може бути корисною, але вона може переобтяжувати користувачів, які шукають прості та швидкі рекомендації.
- Рекламні оголошення: Наявність реклами у безкоштовній версії.

3. Letterboxd. Letterboxd (рис. 1.3) [23] є соціальною мережею для любителів кіно, яка дозволяє користувачам створювати профілі, вести щоденники переглядів, писати рецензії, оцінювати фільми та створювати списки перегляду. Користувачі можуть стежити за іншими користувачами, обмінюватися думками та рекомендаціями, що створює активну спільноту кінофанів. Інтерфейс платформи зручний та інтуїтивно зрозумілий, що сприяє комфортному користуванню. Letterboxd також пропонує можливість створення

та перегляду різних списків фільмів, наприклад, за жанром, режисером або темою.

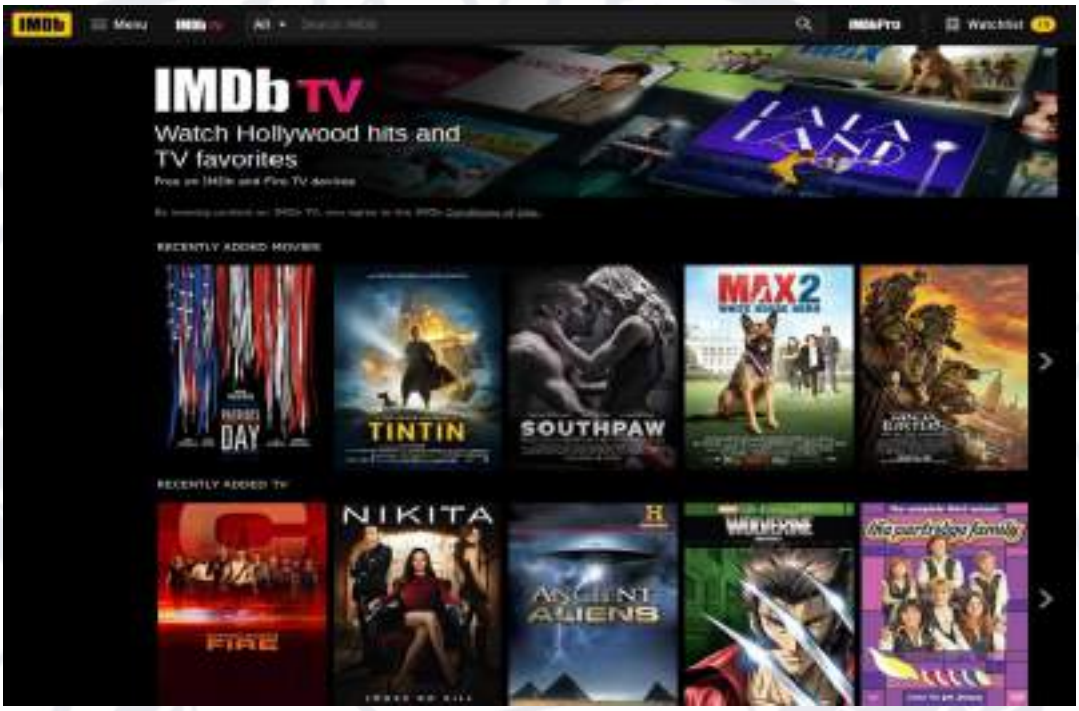

Рисунок 1.2 – IMDb [22]

Функціональні можливості платформи:

- Соціальна мережа для кінофанів. Можливість створення профілів, оцінювання фільмів та ведення щоденників переглядів.
- Персоналізовані рекомендації. Рекомендації на основі вподобань та оцінок користувачів.
- Списки фільмів. Можливість створення та перегляду списків фільмів за різними категоріями.

Переваги платформи:

- Соціальні функції. Можливість взаємодії з іншими кінофанами, обговорення фільмів та обмін рекомендаціями.
- Інтуїтивний інтерфейс. Зручний у використанні інтерфейс з можливістю персоналізації.

Недоліки:

- Обмеженість бази даних. Менша база даних порівняно з великими платформами, такими як IMDb.
- Платні функції. Деякі функції доступні тільки у платній версії.

Рисунок 1.3 – Letterboxd [23]

4. JustWatch. JustWatch (рис. 1.4) [24] – це багатофункціональна платформа, створена для пошуку та навігації контенту на різних стрімінгових сервісах. Її головна мета – допомогти користувачам швидко знайти, де саме доступний потрібний фільм або серіал, і спростити вибір серед величезної кількості онлайн-контенту.

Функціональні можливості:

- JustWatch інтегрує дані з багатьох популярних сервісів, таких як Netflix, Amazon Prime Video, Disney+, HBO Max, Hulu, Apple TV+, та інших. Ви можете побачити, чи доступний контент для перегляду в рамках підписки, оренди чи купівлі.
- Користувачі можуть налаштовувати пошук за жанрами, роком випуску, рейтингом IMDb або TMDb, тривалістю та навіть віковим рейтингом. Це спрощує вибір, якщо точно не знаєш, що хочеться подивитися.

- Платформа дозволяє обирати країну, що важливо, оскільки доступність фільмів та серіалів може варіюватися залежно від регіону.
- JustWatch пропонує функцію створення списків, де можна зберігати фільми та серіали для майбутнього перегляду.
- Користувачі можуть переглядати розділ новинок або отримувати інформацію про знижки на оренду чи купівлю контенту.

Переваги:

- Дозволяє швидко визначити, на яких платформах доступний потрібний контент (Netflix, Hulu, Amazon Prime Video, Disney+, тощо).
- Можливість фільтрувати за жанром, рейтингом IMDb, роком випуску, країною, типом (фільм/серіал).
- Інформація про новинки, знижки на оренду чи купівлю контенту оновлюється оперативно.
- Сервіс підтримує вибір країни, що допомагає дізнатись доступність контенту у вашому регіоні.
- Базові функції доступні безкоштовно, а додаток простий у використанні.

Недоліки:

- Для менш популярних стрімінгових платформ або регіонів база даних може бути не повною.
- Іноді у додатку з’являється реклама, що може заважати користувачеві.
- Деякі локальні стрімінгові сервіси або платформи можуть бути відсутні.
- JustWatch фокусується на пошуку доступності контенту, але не на персональних рекомендаціях.

Розробка інформаційної системи кінематографічного навігатора має на меті створення інструменту, який допоможе користувачам знаходити та обирати фільми відповідно до їхніх індивідуальних вподобань.

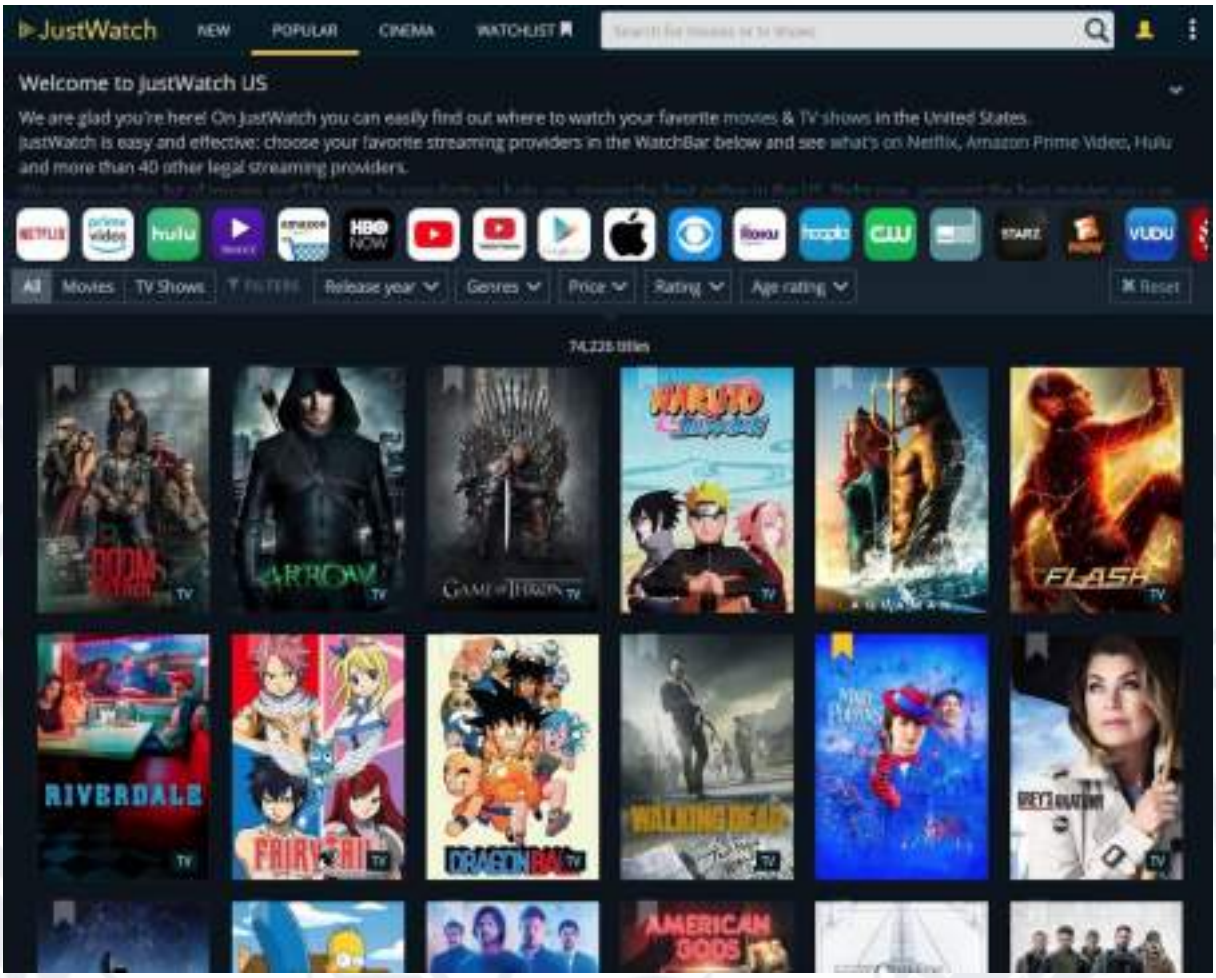

Рисунок 1.4 – JustWatch [24]

Програма повинна забезпечувати наступні функції:

- простий та інтуїтивно зрозумілий інтерфейс для навігації та пошуку.
- можливість додавання нових фільмів та серіалів до каталогу;
- можливість перегляду деталей про фільм;
- фільтрація фільмів за допомогою ключових слів;
- використання бази даних для зберігання інформації про фільми;
- регістрацію та авторизацію користувачів;
- можливість авторизованим користувачам додавати фільми/серіали в закладки;
- персоналізовані рекомендації завдяки алгоритмам аналізу даних;
- рекомендаційна система;
- функція «випадковий фільм/серіал».

ВИСНОВКИ ДО РОЗДІЛУ 1

У першому розділі розглянуто теоретичні та методологічні основи створення інформаційної системи кінематографічного навігатора з використанням інтелектуального аналізу рекомендацій. Визначено, що кінематографічний навігатор допомагає користувачам знаходити відповідний контент серед великої кількості доступних фільмів та серіалів.

Проаналізовано основні компоненти системи та методи інтелектуального аналізу даних. Вивчено існуючі аналоги кінематографічних навігаторів, що дозволило виокремити їхні переваги та недоліки.

Визначено основні вимоги до функціоналу системи: простий користувацький інтерфейс, ефективне зберігання даних, фільтрація фільмів та інші.

Таким чином, перший розділ заклав теоретичну основу для подальшої розробки кінематографічного навігатора, визначивши ключові підходи, методи та вимоги до його функціоналу.

РОЗДІЛ 2
ПРОГРАМНІ СЕРВІСИ ТА ІНСТРУМЕНТИ ДЛЯ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ КІНЕМАТОГРАФІЧНОГО НАВІГАТОРУ З ВИКОРИСТАННЯМ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ РЕКОМЕНДАЦІЙ

2.1 Мови програмування та основні засоби розробки.

Мова програмування Python [25]. Python – це високорівнева, інтерпретована мова програмування, яка набула популярності завдяки своїй простоті та читабельності. Вона широко використовується для розробки програмного забезпечення, включаючи веб-додатки, наукові обчислення, аналіз даних, штучний інтелект, а також для створення графічних інтерфейсів користувача [26-27].

Головними перевагами Python є його простий синтаксис та велика кількість бібліотек, які забезпечують високу продуктивність і гнучкість у процесі розробки. Вибір Python для створення програми пояснюється наступними факторами:

– Простота та доступність. Python дозволяє швидко реалізувати ідеї завдяки лаконічному та інтуїтивному коду. Це знижує ризик помилок і спрощує підтримку та розширення проекту.

– Велика кількість бібліотек. Python має багату екосистему бібліотек, таких як PyQt для створення графічних інтерфейсів, MongoDB для роботи з базами даних, та бібліотеки для роботи з медіаконтентом (наприклад, IMDb API), що суттєво прискорює розробку і додає функціональність програмі.

– Кросплатформеність. Python працює на різних операційних системах (Windows, macOS, Linux), що дозволяє створювати програми, які можуть бути запущені на різних пристроях без змін у коді.

– Спільнота та документація. Python має велику та активну спільноту розробників, що забезпечує доступ до рішень на різноманітні технічні питання.

Крім того, детальна документація сприяє швидкому навчанню та легкій інтеграції нових інструментів у проєкт.

Таким чином, Python є ідеальним вибором для розробки складних програм з багатим функціоналом та інтуїтивно зрозумілим інтерфейсом користувача.

Середовище розробки PyCharm [28]. PyCharm (рис. 2.1) – це інтегроване середовище розробки (IDE) для програмування на Python, розроблене компанією JetBrains. Воно надає багатий набір інструментів і можливостей, які забезпечують швидкість, ефективність і зручність у процесі розробки. PyCharm широко використовується як новачками, так і професіоналами завдяки своїй потужності та зручності [29].

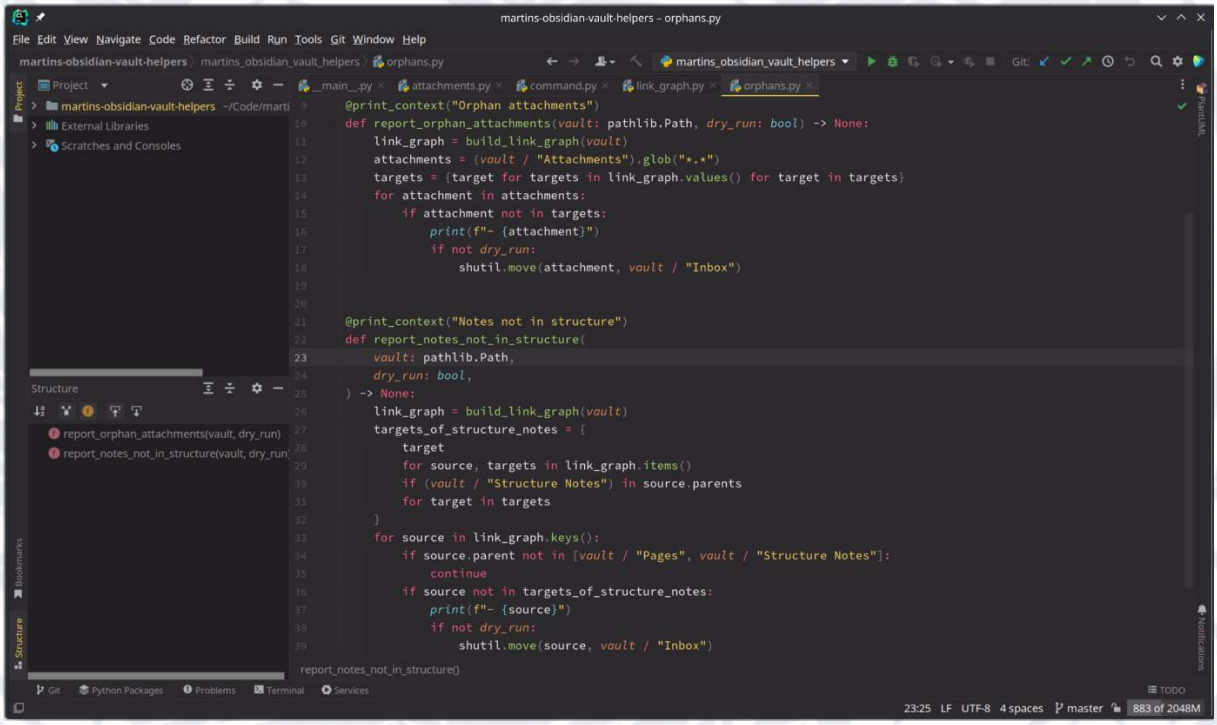

Рисунок 2.1 – Середовище PyCharm [28]

Основні переваги використання PyCharm для створення програмного забезпечення:

– Інтелектуальний редактор коду. PyCharm забезпечує автодоповнення, підсвічування синтаксису, перевірку коду на наявність помилок в реальному часі, що дозволяє розробникам писати код швидше і з меншою кількістю помилок.

– Інтеграція з системами управління версіями. PyCharm сумісний із популярними системами, такими як Git, Mercurial та Subversion, що дозволяє ефективно працювати в команді, відстежувати зміни в коді та керувати версіями проекту.

– Налагодження та тестування. Інтегрований налагоджувач дозволяє легко знаходити та виправляти помилки, надаючи можливість зупиняти виконання коду на певних етапах (breakpoints), переглядати значення змінних та відстежувати логіку виконання програми. PyCharm також підтримує інтеграцію з тестовими фреймворками, такими як pytest, що спрощує автоматизоване тестування.

– Інтеграція з базами даних. PyCharm пропонує зручний інструмент для роботи з базами даних, що дозволяє легко підключатися до MongoDB, MySQL, PostgreSQL та інших баз даних прямо з IDE, виконувати SQL-запити і управляти даними.

– Підтримка веб-розробки. PyCharm має вбудовану підтримку для веб-фреймворків, таких як Django і Flask, що дозволяє розробляти як десктопні, так і веб-застосунки, використовуючи єдине середовище розробки.

– Кросплатформеність. PyCharm доступний для різних операційних систем, таких як Windows, macOS та Linux, що робить його універсальним інструментом для розробки на різних платформах.

– Інтеграція з Docker та віртуальними середовищами. PyCharm підтримує інтеграцію з Docker-контейнерами та віртуальними середовищами Python, що дозволяє ізолювати проєкти, використовуючи різні залежності для кожного з них.

Загалом, PyCharm є потужним та зручним середовищем розробки, яке допомагає значно підвищити продуктивність програмування, полегшити налагодження та тестування коду, а також забезпечити інтеграцію з важливими інструментами та технологіями.

Бібліотека для створення графічних інтерфейсів PyQt [30]. PyQt (рис. 2.2) – це кросплатформена бібліотека для створення графічних інтерфейсів

користувача (GUI) на мові Python, заснована на інструментарії Qt. Вона дозволяє розробникам створювати високоякісні та функціональні інтерфейси для настільних і мобільних застосунків. PyQt надає потужні можливості для побудови інтерактивних програм, що працюють на різних операційних системах, таких як Windows, macOS та Linux, без необхідності змінювати код під кожну платформу [31-32].

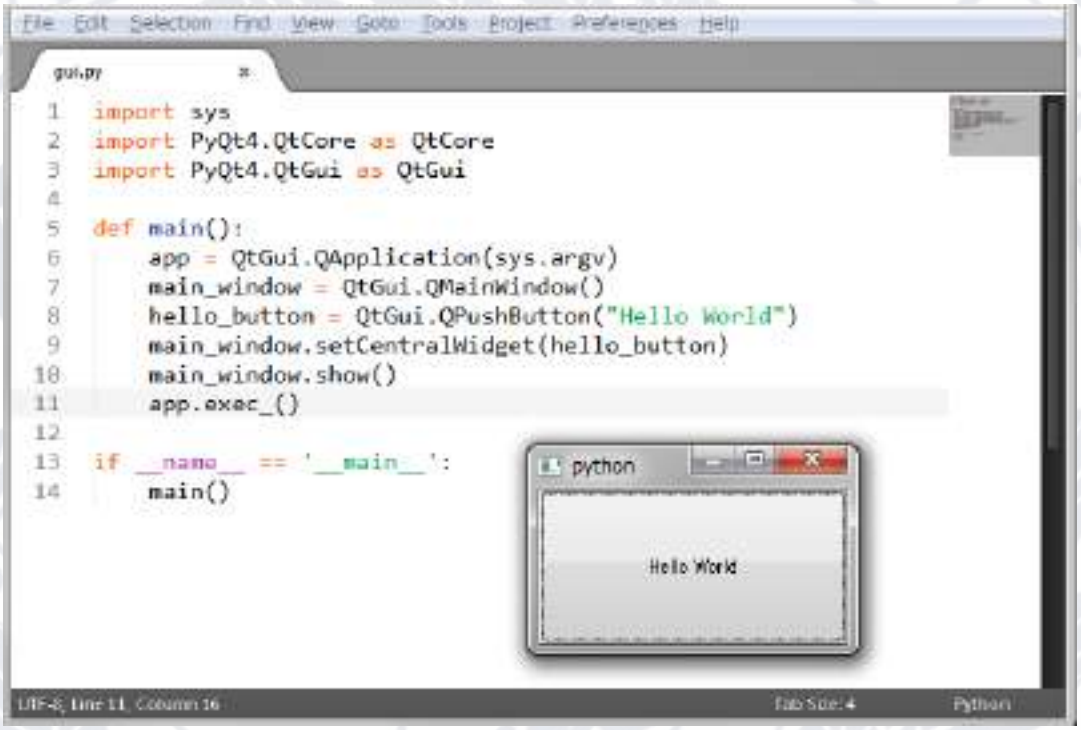

Рисунок 2.2 – Приклад вікна PyQt [30]

Однією з головних переваг PyQt є великий набір візуальних елементів – кнопки, текстові поля, меню, діалогові вікна та інші компоненти, які дозволяють створювати складні інтерфейси з багатим функціоналом. Бібліотека також забезпечує гнучкість у налаштуванні вигляду та поведінки цих елементів, що дозволяє створювати адаптивні інтерфейси, які гармонійно виглядають на різних екранах та роздільних здатностях [31-32].

Завдяки повній інтеграції з Python, PyQt спрощує розробку програмного забезпечення, дозволяючи використовувати можливості цієї мови, такі як простий синтаксис та велика кількість бібліотек. Це значно спрощує створення бізнес-логіки програми, а також інтеграцію з базами даних чи іншими зовнішніми системами.

PyQt підтримує роботу з мультимедійними файлами, такими як зображення та відео, і дозволяє створювати анімації та візуальні ефекти, що робить інтерфейс не тільки функціональним, але й привабливим для користувачів.

PyQt також відзначається своєю модульністю, що дозволяє будувати масштабовані проєкти. Це забезпечує можливість створювати великі додатки, де кожен модуль відповідає за певний функціонал або частину інтерфейсу, що робить підтримку та розвиток таких додатків легшими та ефективнішими.

Завдяки своїй потужності, гнучкості та кросплатформеності, PyQt є чудовим вибором для розробки сучасних програм з багатофункціональним графічним інтерфейсом, який поєднує естетику і продуктивність.

Інструмент для створення графічних інтерфейсів Qt Designer [33]. Qt Designer (рис. 2.3) – це зручний графічний інструмент для проєктування користувацьких інтерфейсів, який входить до складу фреймворку Qt. Його основна мета – спрощення процесу створення інтерфейсів для додатків, надаючи розробникам можливість створювати візуальні компоненти за допомогою перетягування елементів на форму. Це дозволяє значно прискорити процес розробки інтерфейсів без необхідності писати код вручну.

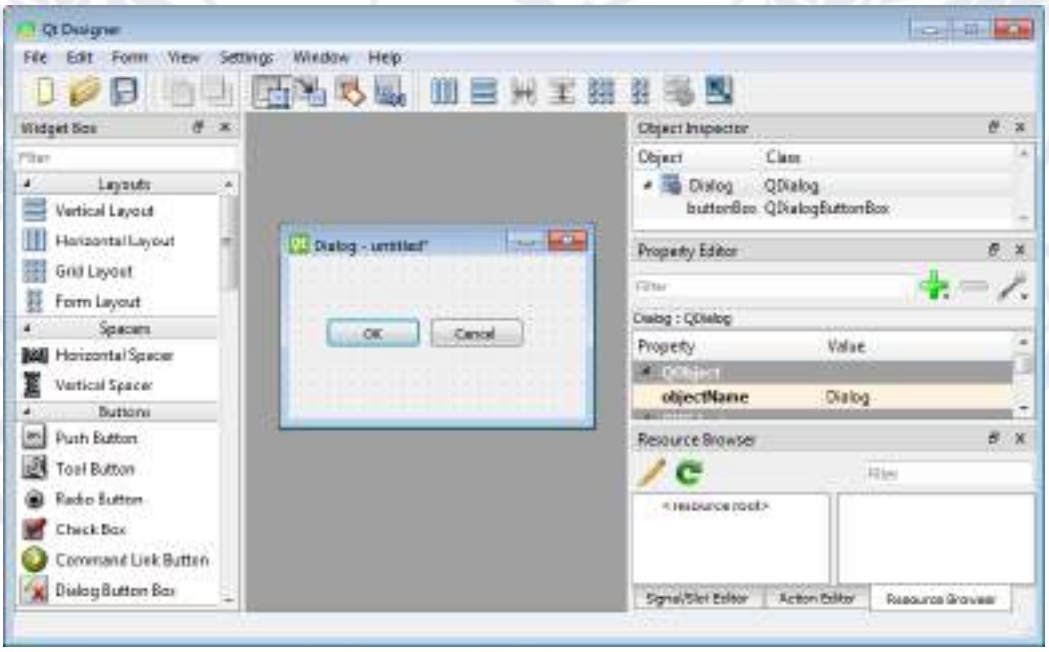

Рисунок 2.3 - Qt Designer [33]

Однією з головних переваг Qt Designer є його інтерактивний підхід до проєктування. Завдяки цьому інструменту розробники можуть візуально створювати інтерфейси, використовуючи попередньо визначені елементи (кнопки, поля введення, таблиці, вкладки тощо). Це полегшує налаштування дизайну, адже всі компоненти можна розташовувати на формі інтуїтивно, як у графічному редакторі. Окрім цього, Qt Designer надає можливість створювати складні ієрархії елементів, що дозволяє розробляти адаптивні інтерфейси з підтримкою різних розмірів вікон і екранних роздільностей.

Qt Designer генерує файли з розширенням .ui, що є XML-форматом, який описує структуру інтерфейсу. Після створення інтерфейсу цей файл може бути легко інтегрований у Python-проєкт за допомогою бібліотеки PyQt або PySide. Розробники можуть або імпортувати файл .ui безпосередньо, або конвертувати його в Python-код за допомогою інструменту pyuic, що дозволяє повністю контролювати програмну логіку, не змінюючи дизайн.

Окрім простоти створення інтерфейсів, Qt Designer також дозволяє налаштовувати сигнали та слоти для обробки подій, що спрощує інтеграцію інтерфейсу з основною логікою програми. Ця функція дозволяє з'єднувати взаємодії користувачів з елементами інтерфейсу та відповідними діями в програмі, забезпечуючи ефективну взаємодію між користувачем і програмою.

Qt Designer є важливим інструментом для будь-якого проєкту, де необхідно швидко та зручно створити графічний інтерфейс користувача. Завдяки можливості створення складних адаптивних інтерфейсів без необхідності писати код вручну, Qt Designer допомагає прискорити процес розробки та забезпечує високу якість кінцевого продукту.

База даних MongoDB [34]. MongoDB (рис. 2.4) – це нереляційна (NoSQL) база даних, яка використовує документи у форматі BSON (Binary JSON) для зберігання даних. Вона широко використовується у сучасній розробці завдяки своїй гнучкості, масштабованості та здатності працювати з великими обсягами даних. На відміну від традиційних реляційних баз даних, MongoDB не використовує таблиці та ряди, а організовує дані у вигляді колекцій документів,

що робить її зручнішою для роботи з динамічними і напівструктурованими даними [35-36].

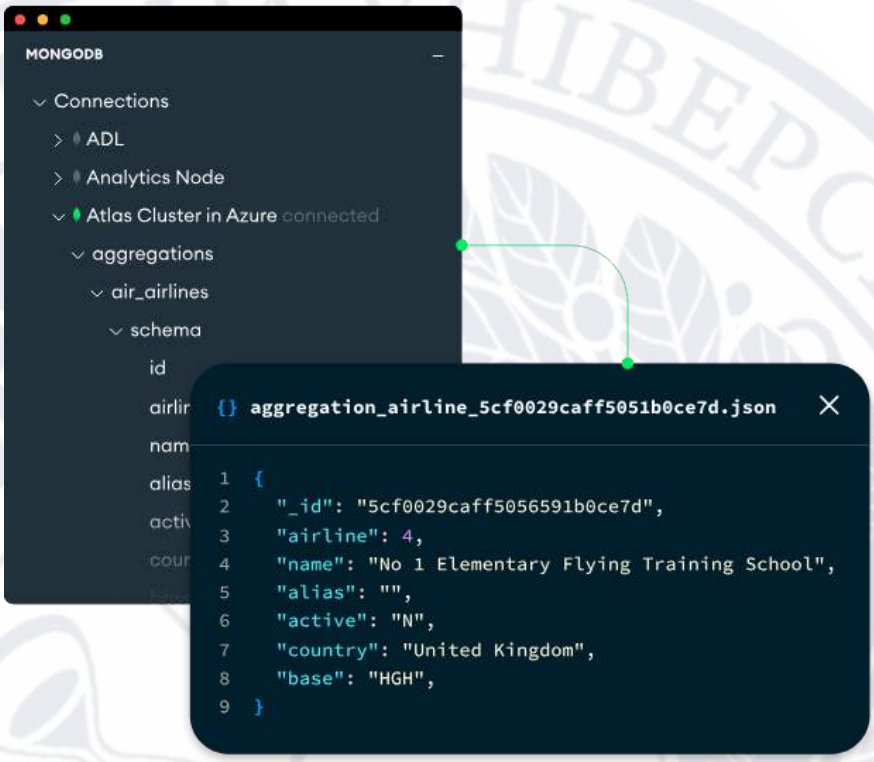

Рисунок 2.4 – Приклад зберігання в MongoDB [34]

Однією з головних переваг MongoDB є її схема-орієнтованість. Це означає, що кожен документ у колекції може мати різну структуру, що дозволяє гнучко керувати даними та легко вносити зміни без необхідності перебудовувати всю базу даних. Такий підхід особливо корисний у проєктах, де структура даних може змінюватися протягом розвитку системи.

MongoDB також відзначається своєю високою продуктивністю та можливістю горизонтального масштабування. Завдяки вбудованій підтримці шардінгу (розподілу даних між декількома серверами), MongoDB дозволяє ефективно обробляти великі обсяги даних і запити, що робить її ідеальною для проєктів, які розраховані на роботу з великим навантаженням і потребують масштабованості в майбутньому [35-36].

Інтеграція MongoDB з Python реалізується через бібліотеку PyMongo, яка

дозволяє здійснювати всі основні операції з базою даних: створення, читання, оновлення і видалення документів. Ця бібліотека спрощує взаємодію програми з базою даних та забезпечує швидкий доступ до даних, дозволяючи ефективно працювати як з одиночними запитами, так і з масовими операціями.

Крім того, MongoDB підтримує складні операції пошуку та фільтрації даних завдяки гнучким запитам, що дозволяє легко знаходити потрібні документи за різними критеріями. Вона також забезпечує високий рівень безпеки даних за допомогою вбудованих механізмів автентифікації, авторизації та шифрування.

MongoDB використовується у багатьох сучасних веб- та мобільних застосунках завдяки своїй гнучкості та зручності у використанні. У проєктах, де необхідна швидка робота з великим обсягом динамічних даних, ця база даних є одним із найкращих варіантів. Завдяки своїй потужності, масштабованості та простоті використання, MongoDB допомагає створювати високоефективні програми, що відповідають вимогам сучасної індустрії програмного забезпечення.

2.2 Функціональні вимоги до інформаційної системи кінематографічного навігатора

Програма кінематографічного навігатора призначена для полегшення процесу пошуку фільмів та серіалів. Для реалізації цієї мети в системі необхідно забезпечити наступні функції:

1) Гість:
- перегляд в галереї доступних фільмів та серіалів;
- відкриття деталей про фільми/серіали.

2) Користувач (Клієнт):
- перегляд в галереї доступних фільмів та серіалів;
- реєстрація користувача;
- авторизація користувача;

– відкриття деталей про фільми/серіали;

– додавання фільмів/серіалів в закладки;

– відкриття та перегляд закладок;

– отримання персоналізованих рекомендацій;

– відкриття «випадкового фільму/серіалу»;

– рекомендаційна системи.

3) Адміністратор:

– додавання нових фільмів/серіалів в галерею;

– перегляд в галереї доступних фільмів та серіалів;

– відкриття деталей про фільми/серіали;

– додавання фільмів/серіалів в закладки;

– відкриття та перегляд закладок.

Характеристики програмного та апаратного забезпечення:

1) Програмні вимоги:

– операційна система має бути Windows 7 або новіша, 64-розрядна.

2) Апаратні вимоги:

– процесор: Intel® Atom із тактовою частотою від 1.60 GHz або продуктивніший;

– кількість ядер: щонайменше 2;

– оперативна пам’ять: мінімум 4 GB;

– графічний адаптер: інтегрований або дискретний, із пам’яттю не менше 512 MB;

– накопичувач: жорсткий диск об’ємом від 70 GB;

– периферійні пристрої: клавіатура та миша (або touchpad);

– дисплей: монітор чи екран із роздільною здатністю не менше 1280×720;

– мережевий адаптер: підтримка Wi-Fi 802.11b/g/n або сучасніший стандарт.

Вимоги до вхідних/вихідних/проміжних даних:

1) Вимоги до вхідних даних:

- Кінобаза: Система має підтримувати завантаження даних про фільми та серіали, включаючи їх назви, описи та іншу специфічну інформацію.
- Інформація про користувачів: Система повинна збирати й зберігати дані клієнтів, зокрема імена, прізвища, логіни та паролі.

2) Вимоги до вихідних даних:

- Інформація про контент: Система має забезпечувати користувачів детальною інформацією про фільми/серіали, які вони переглядають під час пошуку.

3) Вимоги до проміжних даних:

- Аналітичні дані: Для виконання аналізу, наприклад, відстеження вподобань користувачів, система може потребувати тимчасового зберігання проміжної інформації.

4) Вимоги до безпеки та конфіденційності даних:

- Захист даних: Система повинна гарантувати безпеку та конфіденційність інформації користувачів відповідно до чинних стандартів.
- Контроль доступу: Необхідно впровадити механізми аутентифікації та обмеження доступу, що дозволяють отримувати доступ лише авторизованим.
- Резервне копіювання: Система має включати механізми резервного зберігання даних і процедури відновлення для запобігання втратам або пошкодженню інформації у разі аварій.

2.3 Проектування графічного інтерфейсу та архітектури програми

Для розробки графічного інтерфейсу застосовуються інтегроване середовище розробки PyCharm і дизайнер інтерфейсу Qt Designer. З їх допомогою будуть створені такі вікна автоматизованої системи: головне вікно; вікно авторизації Користувача; вікно регістрації Користувача; вікно детальної інформації про фільм; вікно детальної інформації про серіал; вікно закладок; вікно Адміністратора; вікно додавання нових фільмів; вікно додавання нових серіалів.

Для ілюстрації структури та архітектури програми застосовуються дві схеми, а саме 1 блок-схема та 1 структурна схема Константайна (рис. 2.5 – 2.6).

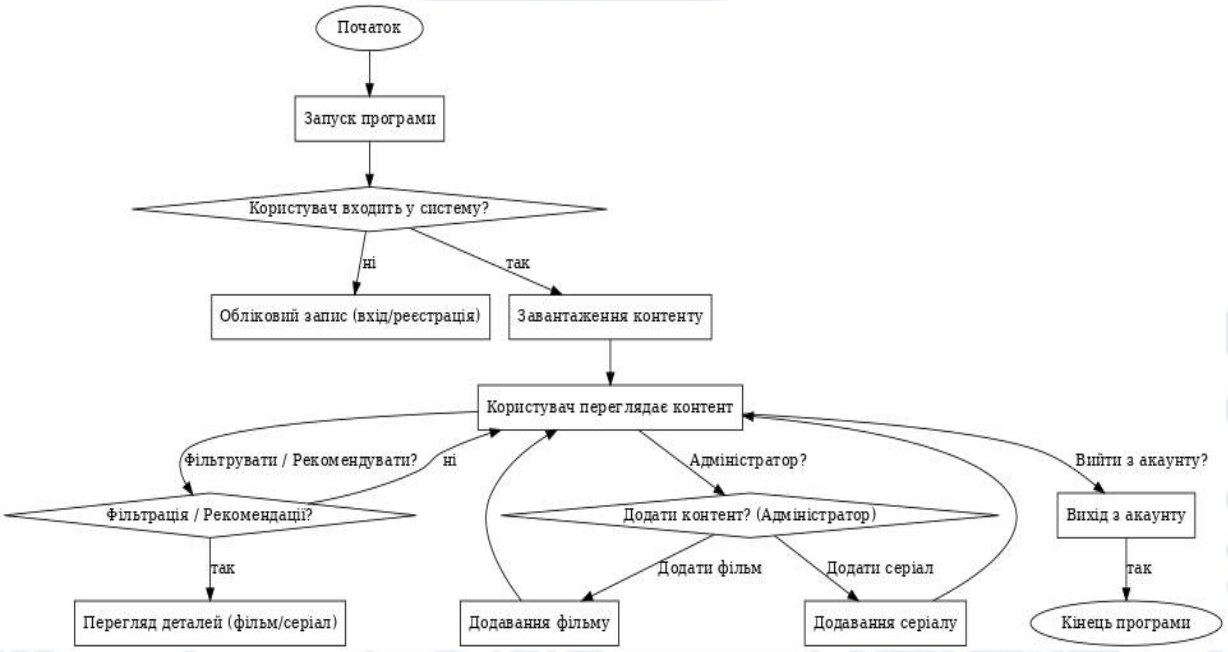

Рисунок 2.5 – блок-схема кінематографічного навігатора

Блок-схема пояснює послідовність дій, прийняття рішень, взаємозв’язки між елементами процесу, а також вхідні та вихідні дані. Вона візуалізує алгоритм для полегшення розуміння й аналізу роботи програми [37].

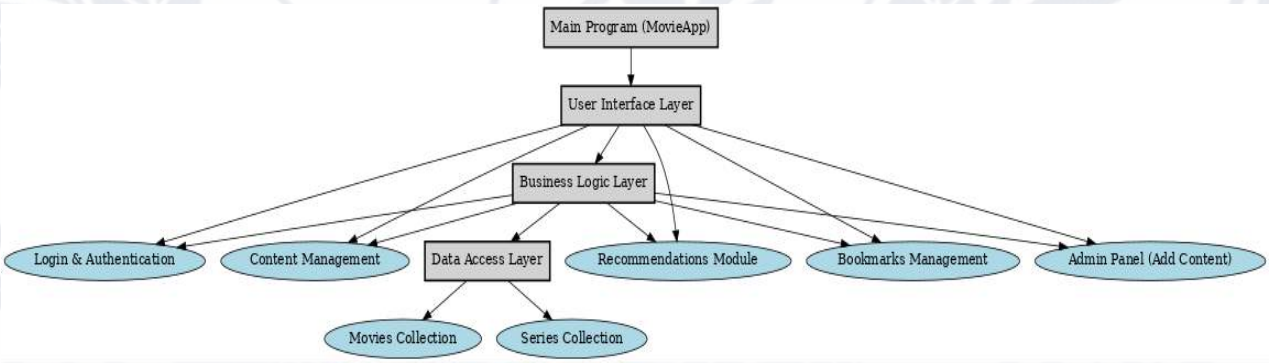

Рисунок 2.6 - Структурна схема Константайна кінематографічного навігатора

Структурна схема Константайна – це графічне зображення структури системи, яке демонструє її основні складові та зв'язки між ними. Вона заснована на принципі розподілу системи на менші підсистеми та модулі з урахуванням їх взаємодії [38].

ВИСНОВКИ ДО РОЗДІЛУ 2

У другому розділі проведено комплексний аналіз програмних сервісів та інструментів, актуальних для розробки інформаційної системи кінематографічного навігатора з використанням інтелектуального аналізу рекомендацій. Було визначено функціональні вимоги до системи, які включають базові та розширені можливості для забезпечення персоналізованого користувацького досвіду, а також вимоги до програмного та апаратного забезпечення, що гарантують стабільність і продуктивність роботи системи.

Також увагу було приділено вимогам до вхідних, вихідних та проміжних даних, що дозволяють коректно обробляти та зберігати інформацію для ефективної роботи алгоритмів машинного навчання. Крім того, у розділі виконано проектування структури та архітектури програми, що включає ключові модулі, взаємодію між ними, а також вибір оптимального підходу до реалізації архітектурного рішення.

Отримані результати закладають основу для ефективної реалізації інформаційної системи, яка буде забезпечувати точність рекомендацій та простоту використання. Виконане проектування дозволяє перейти до наступного етапу розробки – безпосередньої реалізації програмного забезпечення.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОТОТИПУ ІНФОРМАЦІЙНОЇ СИСТЕМИ КІНЕМАТОГРАФІЧНОГО НАВІГАТОРА

3.1 Засоби реалізації та компоненти програми

Мова програмування: Python 3.9.

Середовище розробки: PyCharmEdu 2022.2.3 – для редагування, компіляції, інтерпретації коду.

Використанні бібліотеки:

- sys – це стандартна бібліотека, яка надає можливість використовувати функції для взаємодії з інтерпретатором;
- PyQt5 – бібліотека, що застосовується для розробки інтерфейсу та реалізації функціоналу для взаємодії користувача з програмою;
- pymongo – бібліотека, призначена для роботи з сервером MongoDB (і базою даних);
- bson – бібліотека, яка використовується для роботи з форматом даних BSON (Binary JSON), який є форматом серіалізації даних, що застосовується в MongoDB;
- os – бібліотека, яка забезпечує функції для взаємодії з операційною системою.

База даних прототипу кінематографічного навігатора містить п’ять таблиць. Перша таблиця, movies (рис. 3.1), зберігає фільми та деталі про них. Вона має 14 полів: id – ідентифікаційний номер фільму, name – назва фільму, description – опис фільму, image_path – шлях до зображення постеру фільма, release_date – дата випуску, countries – країни, які працювали над фільмом, genres – жанри, age_rating – вікова категорія, duration – тривалість, budget – бюджет фільму, imdb_rating – оцінка на IMDb, director – режисер, actors – актори, trailer_link – посилання на трейлер.

Друга таблиця, series (рис. 3.2), похожа до попередньої таблиці, зберігає серіали і деталі про них, та має наступні поля: id – ідентифікаційний номер

серіалу, name – назва серіалу, description – опис серіалу, image_path – шлях до зображення постеру серіалу, release_date – дата випуску, countries – країни, які працювали над серіалом, genres – жанри, age_rating – вікова категорія, episode_duration – тривалість епізоду серіалу, imdb_rating – оцінка на IMDb, seasons – кількість сезонів, episodes – сумарна кількість епізодів серіалу, director – режисер, actors – актори, trailer_link – посилання на трейлер.

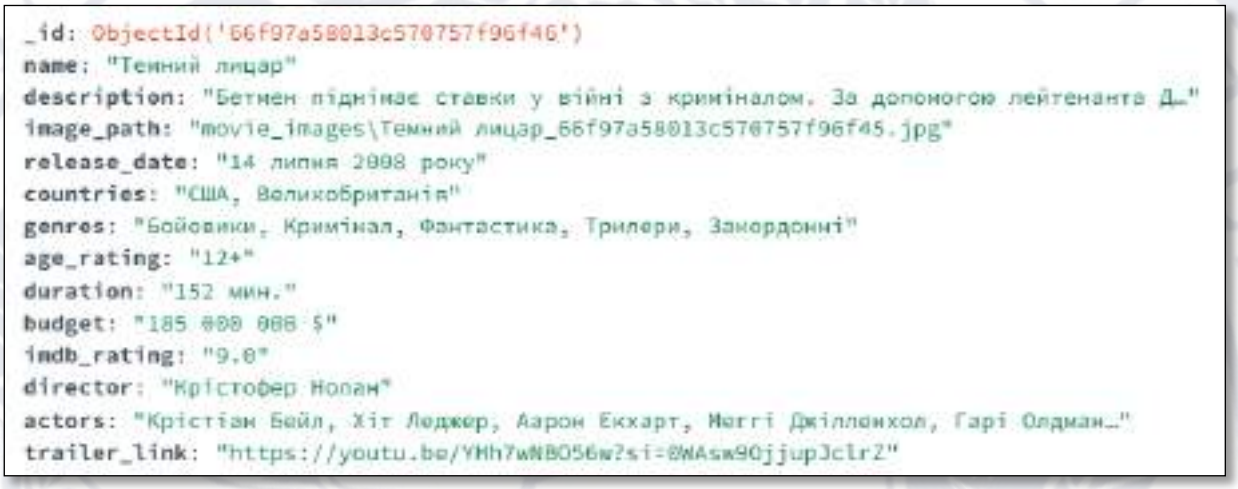

Рисунок 3.1 – Таблиця movies

Рисунок 3.2 – Таблиця series

Третя таблиця, users (рис. 3.3), зберігає дані про зареєстрованих користувачів та має наступні поля: id – ідентифікаційний номер користувача,

first_name – ім’я, last_name – фамілія, login – логін до акаунту, password – пароль до акаунту.

Рисунок 3.3 – Таблиця users

Останні три таблиці, will_watch, watching, not_interested (рис. 3.4), однакові й слугують як закладки для користувачів, та містять по 3 поля: id – ідентифікаційний номер закладки, user_login – логін користувача, який добавив фільм або серіал в закладку, movie_id – ідентифікаційний номер фільму або серіалу, який був добавлений в закладку.

Рисунок 3.4 – Приклад однієї з трьох таблиц

3.2 Опис класів та методів класів, використаних в реалізації програми

1) Імпорт бібліотек

```
import sys
from PyQt5.QtWidgets import (QApplication, QLabel, QPushButton,
QVBoxLayout, QHBoxLayout, QWidget,
                             QScrollArea, QDialog, QLineEdit, QFileDialog, QMessageBox,
QFormLayout, QTextEdit,
                             QComboBox, QTabWidget)
from PyQt5.QtGui import QPixmap, QIcon
from PyQt5.QtCore import Qt, QSize, QTimer
```

```
from pymongo import MongoClient
import os
from bson import ObjectId
import random
```

- sys: Використовується для управління виконанням програми. Наприклад, при запуску GUI з PyQt5, важливо забезпечити коректне завершення програми через sys.exit.
- PyQt5.QtWidgets: Набір графічних компонентів для створення інтерфейсу (кнопки, текстові поля, форми).
  - Компоненти, як-от QApplication, QLabel, QVBoxLayout, дозволяють створювати складні вікна для користувачів.
- PyQt5.QtGui:
  - QPixmap – для завантаження та обробки зображень (наприклад, обкладинок фільмів).
  - QIcon – для іконок додатка чи кнопок.
- PyQt5.QtCore:
  - Qt – набір констант (наприклад, центрування тексту, напрямок вирівнювання).
  - QSize – для встановлення розмірів.
  - QTimer – для таймерів, що можуть бути потрібні для оновлення інтерфейсу чи виконання періодичних завдань.
- pymongo:
  - MongoClient – основний компонент для підключення до MongoDB.
- os: Надає функції роботи з файловою системою, як створення директорій.
- bson.ObjectId: Використовується для взаємодії з унікальними ідентифікаторами в MongoDB.
- random: Може застосовуватись для генерації випадкових ID, значень чи вибору (поки що не використовується).

2) Підключення до MongoDB

```
client = MongoClient("mongodb://localhost:27017/")
db = client['movie_gallery']  # База даних 'movie_gallery'
movies_collection = db['movies']  # Колекція для фільмів
users_collection = db['users']  # Колекція для користувачів
series_collection = db['series']
```

- MongoClient: Підключення до локального екземпляра MongoDB через стандартний порт 27017.
- db: Створює або використовує існуючу базу даних movie_gallery.
- Колекції:
  - movies_collection: Зберігає документи про фільми.
  - users_collection: Для користувачів (логін, пароль, особисті дані).
  - series_collection: Аналогічно movies, але для серіалів.

3) Робота із зображеннями

```
IMAGE_PATH = "movie_images"
# Перевіряємо чи існує директорія для збереження зображень
if not os.path.exists(IMAGE_PATH):
   os.makedirs(IMAGE_PATH)
```

- IMAGE_PATH:
  - Константа визначає шлях до папки, де зберігатимуться зображення фільмів.
  - Це може бути, наприклад, постери чи заставки.
- Перевірка існування папки:
  - os.path.exists(IMAGE_PATH) перевіряє, чи вже існує папка.
  - Якщо ні, папка створюється за допомогою os.makedirs(IMAGE_PATH).
- Чому це важливо:
  - Забезпечує впорядковане збереження файлів.
  - Уникнення помилок, якщо програма намагається записати файл у неіснуючу директорію.

4) Клас DatabaseManager

class DatabaseManager:

```
def __init__(self):
    # Підключення до MongoDB
    self.client = MongoClient("mongodb://localhost:27017/")
    self.db = self.client['movie_gallery']
    # Колекції для закладок
    self.will_watch_collection = self.db['will_watch']
    self.watching_collection = self.db['watching']
    self.not_interested_collection = self.db['not_interested']
    self.movies_collection = self.db['movies']  # Колекція для фільмів
```

- Конструктор __init__:
  - Ініціалізує підключення до бази даних.
  - Підключається до чотирьох колекцій:
    - will_watch_collection: Фільми, які користувач хоче подивитись.
    - watching_collection: Фільми, які переглядаються.
    - not_interested_collection: Фільми, які не цікавлять користувача.
    - movies_collection: Загальна інформація про всі фільми.
- Роль цього класу:
  - Відповідає за всі операції з базою даних:
    - Додавання фільмів до категорій.
    - Отримання даних.
    - Переміщення між категоріями.

➢ Метод add_to_bookmarks:

```
def add_to_bookmarks(self, user_id, user_login, movie_id, category):
data = {'user_id': user_id, 'user_login': user_login, 'movie_id': movie_id}
if category == 'will_watch':
    self.will_watch_collection.insert_one(data)
elif category == 'watching':
    self.watching_collection.insert_one(data)
```

```
elif category == 'not_interested':
    self.not_interested_collection.insert_one(data)
```

- Вхідні дані:
  - user_id – ID користувача.
  - user_login – логін користувача.
  - movie_id – ID фільму.
  - category – категорія, куди додається фільм.
- Логіка:
  - Дані додаються до відповідної колекції.
  - Залежно від значення category, виконується insert_one(data).

➢ Метод get_user_bookmarks:

```
def get_user_bookmarks(self, user_login, category):
if category == 'will_watch':
    return self.will_watch_collection.find({'user_login': user_login})
elif category == 'watching':
    return self.watching_collection.find({'user_login': user_login})
elif category == 'not_interested':
    return self.not_interested_collection.find({'user_login': user_login})
```

- Повертає всі фільми з певної категорії, прив'язані до user_login.
- Використовує метод find для пошуку документів у відповідній колекції.

➢ Метод check_and_move_bookmark:

```
def check_and_move_bookmark(self, user_login, movie_id, new_category):
    …
```

- Перевірка поточного місцезнаходження:
  - Метод перевіряє, чи фільм уже є в одній із трьох категорій.
- Запит користувачу:
  - Якщо фільм уже існує в іншій категорії, з’являється QMessageBox для підтвердження перенесення.
- Перенесення:

- Видаляється запис із поточної категорії та додається в нову.

➢ Метод get_movie_by_id:

```
def get_movie_by_id(self, movie_id):
    return self.movies_collection.find_one({"_id": movie_id})
```

- Повертає документ про фільм за його унікальним ID.
- find_one повертає лише один збіг (у цьому випадку, шукається за _id).

5) Клас RegistrationDialog

Цей клас реалізує діалогове вікно для реєстрації нових користувачів.

```
def __init__(self, parent=None):
    super().__init__(parent)
    self.setWindowTitle("Registration")
    layout = QFormLayout()
```

- parent: Опціональний параметр, який дозволяє передати батьківський об'єкт. Якщо не передано, вікно працює незалежно.
- setWindowTitle: Встановлює заголовок вікна як "Registration".
- QFormLayout: Організовує елементи форми у вигляді рядків із підписами зліва та відповідними полями вводу справа.

➢ Поля вводу:

```
self.first_name_input = QLineEdit(self)
self.first_name_input.setPlaceholderText("Enter first name")
```

- QLineEdit: Поле для вводу тексту.
- setPlaceholderText: Встановлює текст-заповнювач, який зникає, коли користувач починає вводити дані.

➢ Елементи форми:

```
layout.addRow("First Name:", self.first_name_input)
```

- addRow: Додає елементи форми у форматі "Мітка: Поле вводу".

➢ Кнопка реєстрації:

```
register_button = QPushButton("Register")
register_button.clicked.connect(self.register_user)
```

```
layout.addWidget(register_button)
```

- QPushButton: Кнопка "Register".
- clicked.connect(self.register_user): Підключає функцію register_user до сигналу натискання кнопки.

➢ Метод register_user:

```
def register_user(self):
    first_name = self.first_name_input.text()
    last_name = self.last_name_input.text()
    login = self.login_input.text()
    password = self.password_input.text()
…
```

- text(): Отримує текст із полів вводу.

➢ Валідація даних:

```
if first_name and last_name and login and password:
    # Перевіряємо чи користувач вже існує
    if users_collection.find_one({"login": login}):
        QMessageBox.warning(self, "Registration Error", "User with this login already exists.")
…
```

- Перевіряє, чи заповнені всі поля.
- Шукає в базі MongoDB користувача з однаковим логіном, щоб уникнути дублювання.

➢ Успішна реєстрація:

```
users_collection.insert_one({
    "first_name": first_name,
    "last_name": last_name,
    "login": login,
    "password": password
})
QMessageBox.information(self, "Registration Success", "Registration successful!")
```

self.accept()

- Додає нового користувача до колекції users.
- Використовує QMessageBox.information для підтвердження реєстрації.
- self.accept(): Закриває діалог із позитивним результатом (Accepted).

➢ Помилки:

QMessageBox.warning(self, "Input Error", "Please fill all fields.")

- Якщо поля не заповнені, показує попередження.

6) Клас LoginDialog

Цей клас реалізує діалогове вікно для входу користувачів.

self.login_input = QLineEdit(self)
self.login_input.setPlaceholderText("Enter login")
self.password_input = QLineEdit(self)
self.password_input.setPlaceholderText("Enter password")
self.password_input.setEchoMode(QLineEdit.Password)

- setEchoMode(QLineEdit.Password): Замість тексту у полі пароля відображаються точки або зірочки.

➢ Кнопка входу:

login_button = QPushButton("Login")
login_button.clicked.connect(self.login_user)

- Підключає функцію login_user до кнопки.

➢ Метод login_user:

user = users_collection.find_one({"login": login, "password": password})

- Шукає в базі користувача з відповідними логіном і паролем.

➢ Успішний вхід:

if user:
  QMessageBox.information(self, "Login Success", f"Welcome, {user['first_name']}!")
  self.logged_in_user = user
  self.accept()

- Показує вітання з ім'ям користувача та завершує діалог.

7) Клас AccountDialog

Цей клас дозволяє користувачам вибирати між входом і реєстрацією.

self.logged_in_user = None

```
login_button = QPushButton("Login")
login_button.clicked.connect(self.open_login_dialog)
register_button = QPushButton("Register")
register_button.clicked.connect(self.open_register_dialog)
```

- Встановлює два режими роботи:
  - Вхід через open_login_dialog.
  - Реєстрація через open_register_dialog.
- Метод open_login_dialog:

```
login_dialog = LoginDialog(self)
if login_dialog.exec_() == QDialog.Accepted:
    self.logged_in_user = login_dialog.logged_in_user
    self.accept()
```

- Відкриває діалог для входу.
- Якщо вхід успішний (Accepted), зберігає інформацію про користувача у self.logged_in_user.

- Метод open_register_dialog:

```
register_dialog = RegistrationDialog(self)
register_dialog.exec_()
```

- Відкриває діалог реєстрації, але не змінює стан self.logged_in_user.

- Загальний огляд архітектури:
  - Модульність:
    - Реєстрація (RegistrationDialog) та вхід (LoginDialog) реалізовані як окремі діалоги.
    - AccountDialog виступає точкою доступу для цих функцій.
  - Безпека:

- Паролі приховуються у полі вводу.
- Однак паролі зберігаються у базі даних у відкритому вигляді. Необхідно додати хешування паролів для безпеки.

- Користувацький досвід:
  - Валідація даних під час реєстрації.
  - Інформування через QMessageBox.
- MongoDB:
  - Використовується для зберігання даних користувачів.
  - Забезпечує швидкий пошук через індексацію.

8) Клас MovieDetailDialog

Цей клас відповідає за створення діалогового вікна з детальною інформацією про фільм.

```
def __init__(self, movie, user_login, db_manager, parent=None):
```

- movie: Словник з інформацією про фільм, отриманий з бази даних.
- user_login: Логін користувача, необхідний для роботи з закладками.
- db_manager: Об'єкт класу DatabaseManager для взаємодії з базою даних.
- parent: Батьківське вікно, щоб оновлювати його після додавання закладок.

➢ Назва фільму:

```
title_label = QLabel(movie['name'])
title_label.setAlignment(Qt.AlignCenter)
title_label.setStyleSheet("font-size: 18pt; font-weight: bold;")
```

- Виводить назву фільму, вирівняну по центру, з великим шрифтом.

➢ Деталі фільму:

```
details_text = f"""
<b>Release Date:</b> {movie['release_date']}<br>
<b>Countries:</b> {movie['countries']}<br>
…
```

- Описує інформацію про фільм, включаючи дату випуску, країни, жанри, рейтинг, бюджет тощо.

➢ Зображення фільму:

```
pixmap = QPixmap(movie['image_path']).scaled(200, 300, Qt.KeepAspectRatio)
movie_image.setPixmap(pixmap)
```

- Завантажує постер фільму та масштабує його для відображення.

➢ Трейлер:

```
trailer_label = QLabel(f'<a href="{movie["trailer_link"]}">Watch Trailer</a>')
trailer_label.setOpenExternalLinks(True)
```

- Відображає посилання на трейлер, яке відкривається у браузері.

➢ Додавання до закладок. Вибір категорії:

```
self.bookmark_combo = QComboBox(self)
self.bookmark_combo.addItems(["Буду дивитись", "Дивлюсь", "Не цікаво"])
```

- Дозволяє користувачеві вибрати одну з категорій для закладок.

Рисунок 3.5 – 2 функції класу MovieDetailDialog

➢ Обробка закладок:

```
result = self.db_manager.check_and_move_bookmark(self.user_login,
self.movie['_id'], category)
```

- Використовує метод check_and_move_bookmark з DatabaseManager, щоб:

a. Перевірити, чи фільм уже є в обраній категорії.

b. Додати фільм або перемістити його до нової категорії.

➢ Оновлення батьківського вікна:

```
if isinstance(self.parent_window, BookmarksWindow):
    self.parent_window.refresh_tabs()
```

- Якщо батьківське вікно – це вікно закладок, викликається його оновлення.

Рисунок 3.6 – 3 функції класу MovieDetailDialog

9) Клас AddMovieDialog

Діалог для додавання нового фільму до бази даних.

- Інтерфейс. Поля для вводу:

```
layout.addRow("Movie Name:", self.movie_name_input)
layout.addRow("Description:", self.movie_description_input)
...
```

- Включає поля для вводу всіх необхідних даних: назви, опису, жанрів, режисера тощо.

- Кнопка вибору зображення:

```
select_image_button = QPushButton("Select Image")
select_image_button.clicked.connect(self.select_image)
```

- Відкриває діалог вибору файлів.

- Додавання фільму. Формування даних:

```
movie_data = {
  "name": self.movie_name_input.text(),
  ...}
```

- Створює словник із даними про фільм, взятими з полів вводу.

Рисунок 3.7 - Функції класу AddMovieDialog

- Обробка зображення:

```
image_filename = f"{movie_data['name']}_{ObjectId()}.jpg"
image_path = os.path.join(IMAGE_PATH, image_filename)
```

- Зберігає обране зображення з унікальним ім'ям до локальної папки.

- Додавання до бази даних:

```
movies_collection.insert_one(movie_data)
```

- Фільм додається до колекції movies у базі даних MongoDB.

10) Клас AddSeriesDialog

Аналогічний до AddMovieDialog, але для серіалів.

- Інтерфейс. Містить додаткові поля кількість сезонів та епізодів:

```
self.seasons_input = QLineEdit(self)
self.episodes_input = QLineEdit(self)
```

- Формування даних аналогічне фільмам, але з урахуванням специфіки серіалів (сезони, епізоди).
- Дані додаються до колекції series.

Рисунок 3.8 - Функції класів AddMovieDialog та AddSeriesDialog

11) Клас SeriesDetailDialog

Відображає деталі про серіал.

- Побудований за аналогією до MovieDetailDialog.
- Додаються специфічні для серіалів поля:

```
<b>Episode Duration:</b> {series['episode_duration']} minutes<br>
<b>Seasons:</b> {series['seasons']}<br>
<b>Episodes:</b> {series['episodes']}<br>
```

➢ Загальний аналіз

- Модульність:
  - Класи чітко розділені за функціональністю: перегляд, додавання, обробка фільмів та серіалів.
- Безпека:
  - Дані про фільми зберігаються локально, а потім додаються до бази.
  - Проте відсутня валідація, чи всі поля коректно заповнені (можна додати).
- Взаємодія з базою даних:
  - Весь функціонал додавання й перегляду фільмів/серіалів інтегрується з MongoDB.
- UI/UX:
  - Інтерфейс зручний і зрозумілий. Використання стилів покращує вигляд.

Рисунок 3.9 – функції класу SeriesDetailDialog

12) Клас RecommendationPanel

RecommendationPanel – це віджет, який відповідає за бокову панель для фільтрації контенту (фільмів чи серіалів) за заданими критеріями.

```
class RecommendationPanel(QWidget):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setFixedWidth(300)  # Ширина бокової панелі
        self.setStyleSheet("background-color: #f0f0f0; border: 1px solid black;")
```

Рисунок 3.10 – Клас RecommendationPanel

```
        layout = QVBoxLayout()
        # Фільтр за жанром
        self.genre_label = QLabel("Жанр")
        self.genre_filter = QLineEdit()
        self.genre_filter.setPlaceholderText("Введіть жанр")
        layout.addWidget(self.genre_label)
        layout.addWidget(self.genre_filter)
        # Фільтр за країною
        self.country_label = QLabel("Країна")
        self.country_filter = QLineEdit()
        self.country_filter.setPlaceholderText("Введіть країну")
        layout.addWidget(self.country_label)
        layout.addWidget(self.country_filter)
...
```

Рисунок 3.11 – Фільтри класу RecommendationPanel

13) Клас Sidebar

Sidebar – це бокова панель, яка надає функції для доступу до закладок та виходу з акаунту.

```
class Sidebar(QWidget):
    def __init__(self, logged_in_user, db_manager, parent=None):
        super().__init__(parent)
        self.logged_in_user = logged_in_user  # Ініціалізуємо користувача
        self.db_manager = db_manager  # Ініціалізуємо менеджер бази даних
        self.setFixedWidth(200)
        self.setStyleSheet("background-color: #f0f0f0; border: 1px solid black;")
        layout = QVBoxLayout()
        layout.addStretch()  # Цей віджет відсуває кнопки вниз
        # Кнопка "Закладки"
        bookmarks_button = QPushButton("Закладки")
        bookmarks_button.clicked.connect(self.open_bookmarks_window)
```

Рисунок 3.12 – 1 функції класу Sidebar

```
        layout.addWidget(bookmarks_button)
        # Кнопка "Вийти"
        logout_button = QPushButton("Вийти")
        logout_button.clicked.connect(self.logout)
        layout.addWidget(logout_button)
        layout.addStretch()
        self.setLayout(layout)
    def open_bookmarks_window(self):
        try:
            # Передаємо `self.logged_in_user`, який містить дані про
авторизованого користувача
            self.bookmarks_window = BookmarksWindow(self.parent().logged_in_user,
self.db_manager)
            self.bookmarks_window.show()
        except Exception as e:
            QMessageBox.critical(self, "Error", f"An error occurred: {str(e)}")
    def logout(self):
        self.parent().logout_user()  # Викликаємо функцію для виходу з акаунту
```

Рисунок 3.13 – 2 функції класу Sidebar

- Загальний огляд архітектури
  - RecommendationPanel:
    - Інструмент для фільтрації контенту з простим і зрозумілим інтерфейсом.
    - Використовує MongoDB для пошуку даних за складними умовами (регулярні вирази, порівняння чисел).
  - Sidebar:
    - Надає базові функції для користувача:
    - Перехід до закладок.
    - Вихід із системи.
    - Добре інтегрований із батьківським вікном (використовує методи parent()).

Рисунок 3.14 – 3 функції класу Sidebar

14) Клас BookmarksWindow

Клас BookmarksWindow реалізує вікно закладок, яке дозволяє користувачеві переглядати фільми у різних категоріях: "Буду дивитись", "Дивлюсь", "Не цікаво". Також воно інтегрує простий алгоритм рекомендацій на основі жанрів фільмів.

```
class BookmarksWindow(QDialog):
    def __init__(self, logged_in_user, db_manager, parent=None):
        super().__init__(parent)
        self.setWindowTitle("Закладки")
        self.setFixedSize(800, 600)
        self.user_login = logged_in_user['login'] if logged_in_user else None
        self.db_manager = db_manager
        # Створюємо вкладки
        self.tabs = QTabWidget()
        self.will_watch_tab = QWidget()
        self.watching_tab = QWidget()
        self.not_interested_tab = QWidget()
```

Рисунок 3.15 – 1 функції класу BookmarksWindow

```
        # Додаємо вкладки
        self.tabs.addTab(self.will_watch_tab, "Буду дивитись")
        self.tabs.addTab(self.watching_tab, "Дивлюсь")
        self.tabs.addTab(self.not_interested_tab, "Не цікаво")
        # Налаштовуємо вкладки
        self.setup_tab(self.will_watch_tab, "will_watch")
        self.setup_watching_tab(self.watching_tab)  # Спеціальний метод для
"Дивлюсь"
        self.setup_tab(self.not_interested_tab, "not_interested")
...
```

Рисунок 3.16 – 2 функції класу BookmarksWindow

15) Клас MovieApp

MovieApp є основним вікном програми для роботи з фільмами та серіалами. Він інтегрує функції відображення контенту, пошуку, рекомендаційної системи, а також управління обліковими записами.

```
class MovieApp(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('Movie & Series Gallery')
        self.setGeometry(100, 100, 1000, 500)
        # Задній фон
        self.background_image =
QPixmap("C:\\Users\\HP\\PycharmProjects\\CinemaNavigator\\resources\\fon.jpg")
        # Створення мітки для фону
        self.background_label = QLabel(self)
        self.background_label.setPixmap(self.background_image)
        self.background_label.setScaledContents(True)  # Масштабування зображення
        self.background_label.setGeometry(self.rect())  # Задаємо розміри мітки
відповідно до вікна
        # Ініціалізація менеджера бази даних
        self.db_manager = DatabaseManager()
        ...
```

Рисунок 3.17 – Функції класу MovieApp

16) **Завершення**

```
if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = MovieApp()
    window.show()
    sys.exit(app.exec_())
```

- Цей блок є стандартною точкою входу для запуску програми на базі PyQt5.

Рисунок 3.18 – Завершення коду

3.3 Опис інтерфейсу користувача

Весь інтерфейс написаний на українській мові. Програма має одне головне вікно та декілька допоміжних вікон. Після запуску програми відкривається головне вікно (рис. 3.19), де користувач може переглядати список фільмів або, за допомогою кнопки-перемикача, серіалів (рис. 3.20) та здійснювати пошук за ключовими словами.

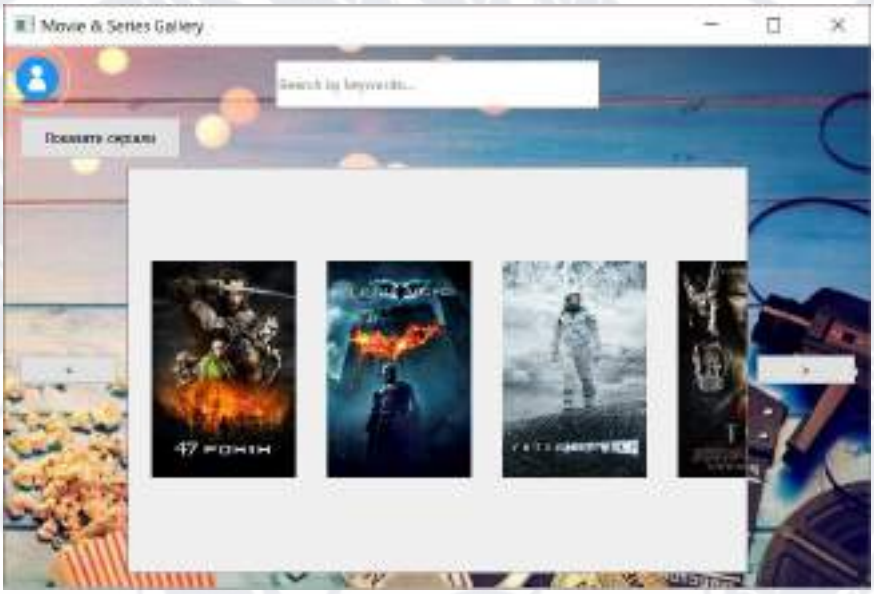

Рисунок 3.19 – Головне вікно (з стрічкою фільмів)

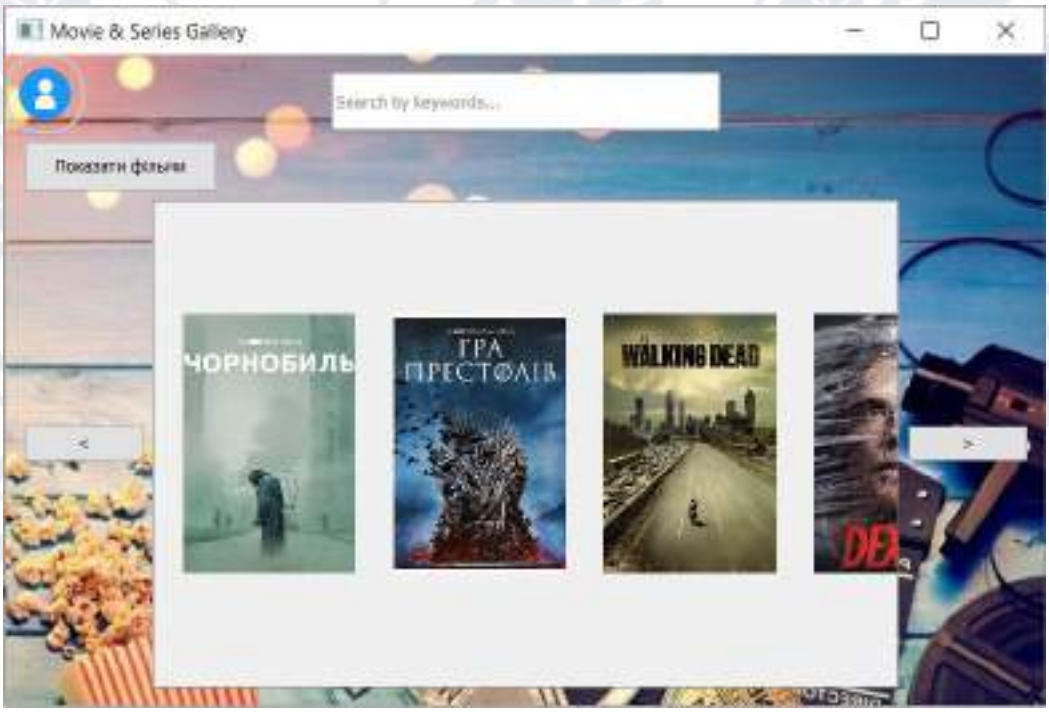

Рисунок 3.20 – Головне вікно (з стрічкою серіалів)

Також після кліку по іконкі в лівому верхньому куті, відкривається вікно акаунту (рис. 3.21), з допомогою якого користувач може зареєструватися (рис. 3.22) та/або авторизуватися (рис. 3.23) в систему.

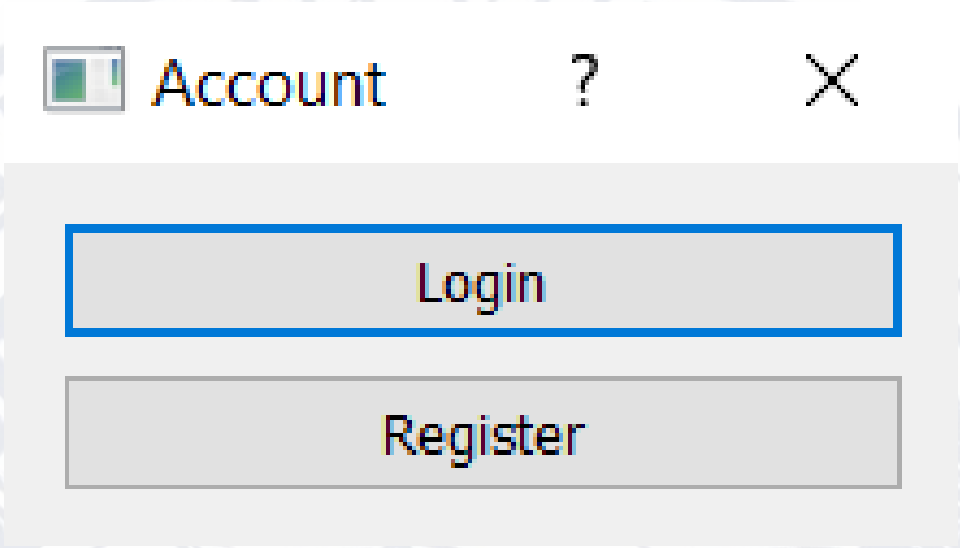

Рисунок 3.21 – Вікно для авторизації акаунту

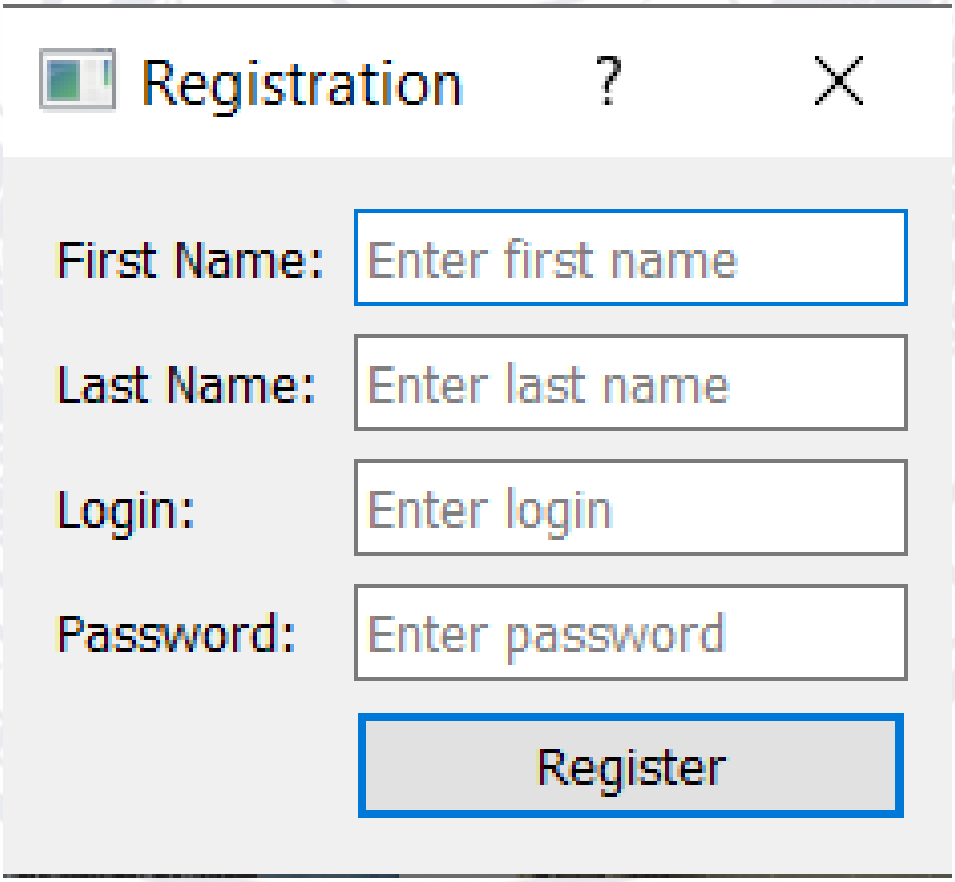

Рисунок 3.22 – Вікно реєстрації

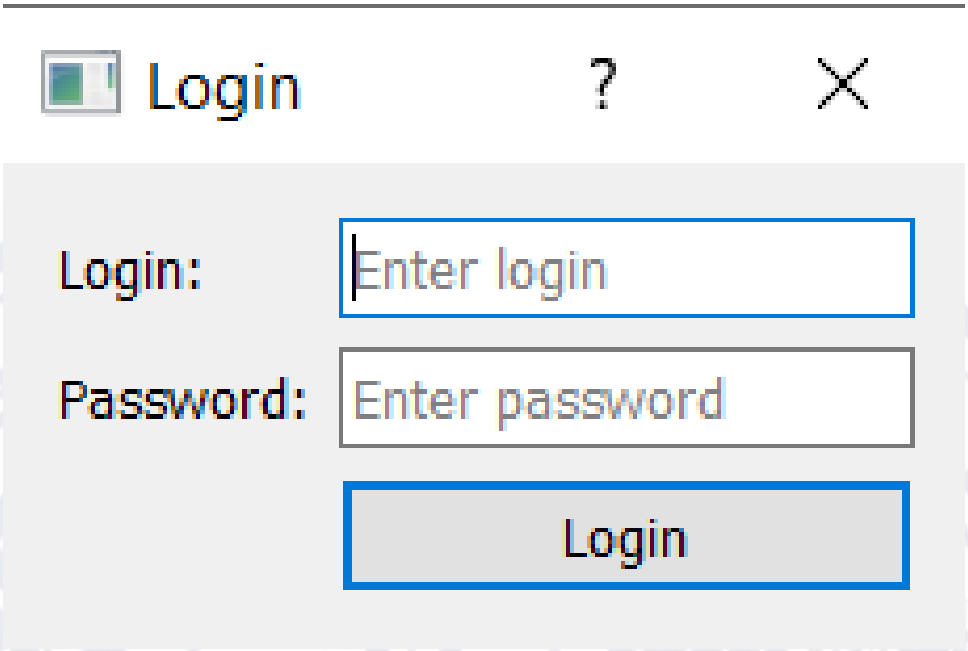

Рисунок 3.23 – Вікно авторизації

При авторизації користувача іконка в кутку змінює колір на жовтий (рис. 3.24), після чого користувач при натискані на фільм чи серіал, зможе не тільки переглядати детальну інформацію, а й добавляти фільм/серіал в закладки (рис. 3.25), знаходити випадковий фільм/серіал та користуватися рекомендаційною системою для кращої фільтрації контенту (рис. 3.26).

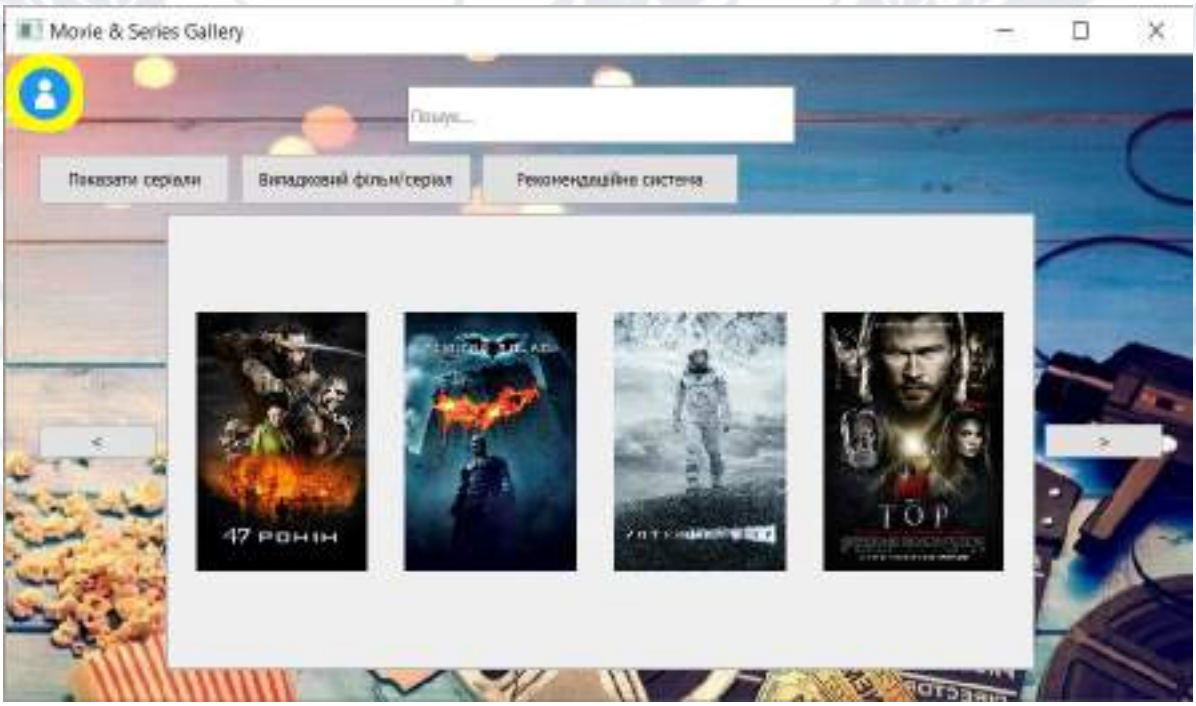

Рисунок 3.24 – Оновлене головне вікно користувача

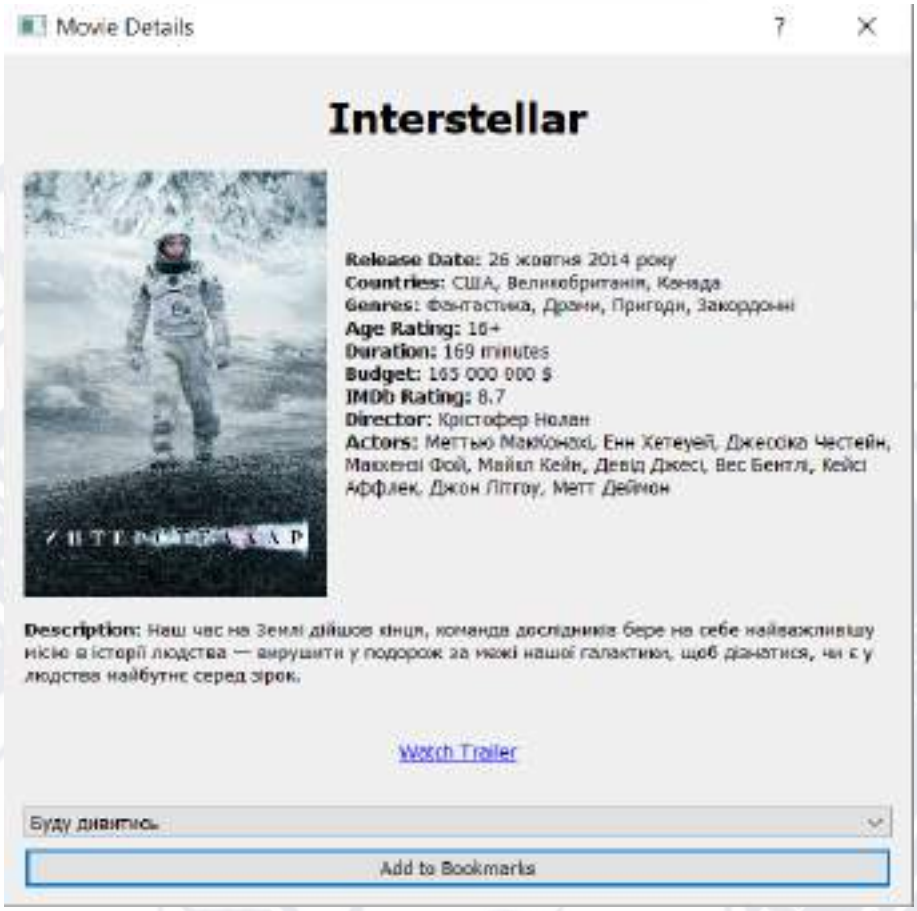

Рисунок 3.25 – Вікно детальної інформації про фільм/серіал

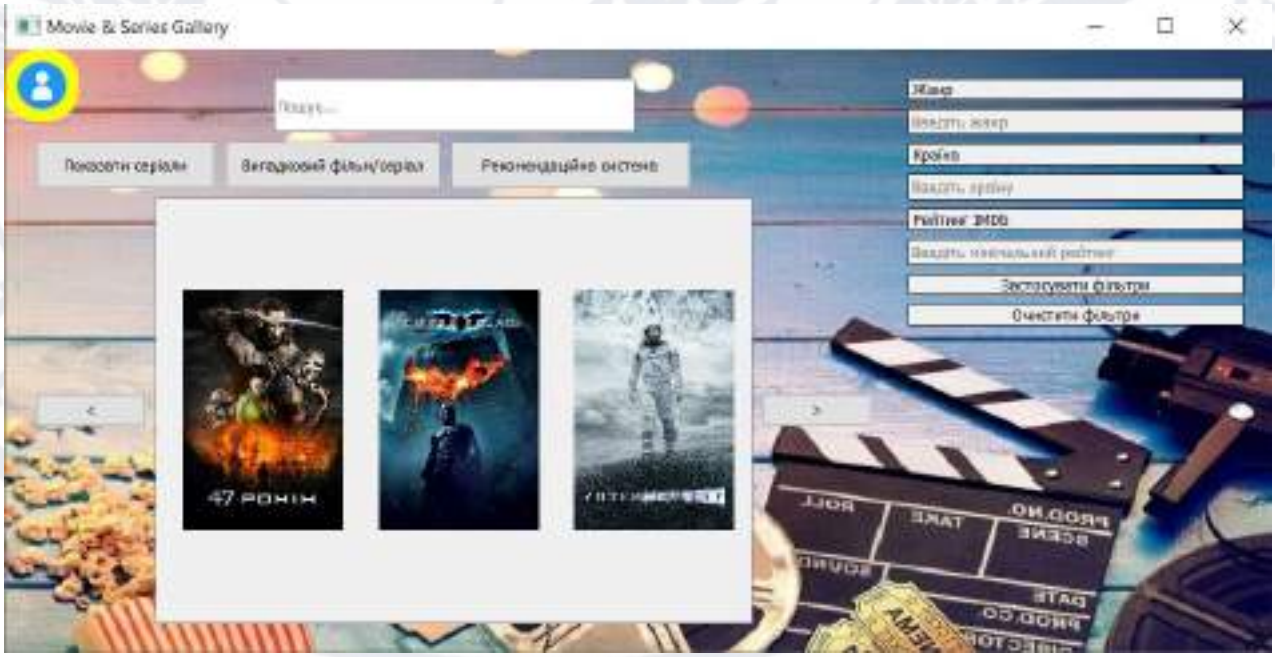

Рисунок 3.26 – Рекомендаційна система

Також після авторизації і натискання на жовту іконку тепер для користувача відкривається невелика бокова панель (рис. 3.27), де є дві кнопки: з допомогою кнопки «Закладки» відкриваються закладки (рис. 3.28), а з допомогою кнопки «Вийти» користувач виходить зі свого акаунту.

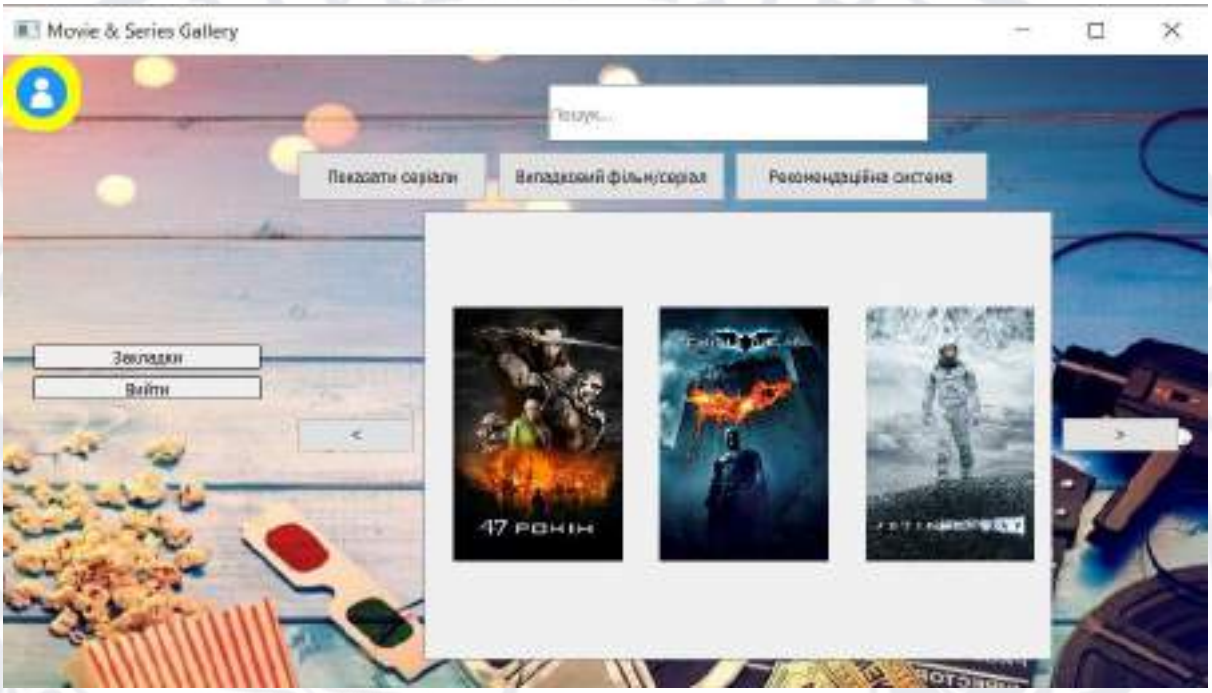

Рисунок 3.27 – Бокова панель з 2 кнопками

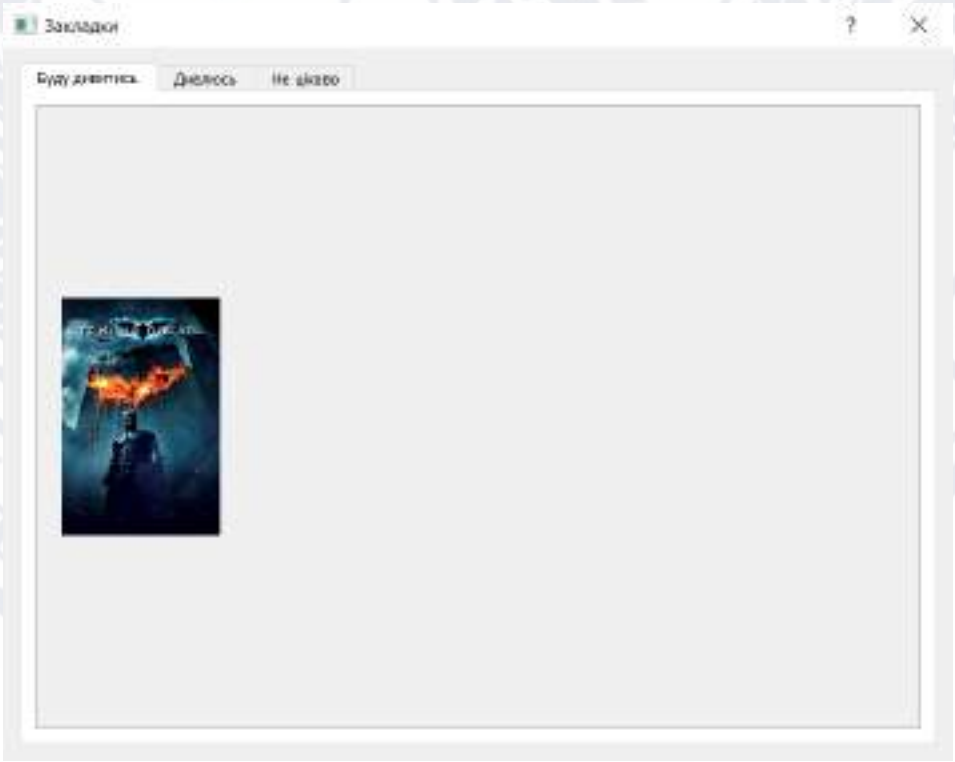

Рисунок 3.28 – Вікно закладок

Якщо ж авторизується Адмін, то іконка змінюється на червону і з’являються дві спеціальні кнопки (рис. 3.29) з допомогою, яких адміністратор може додавати в галерею нові фільми та серіали (рис. 3.30).

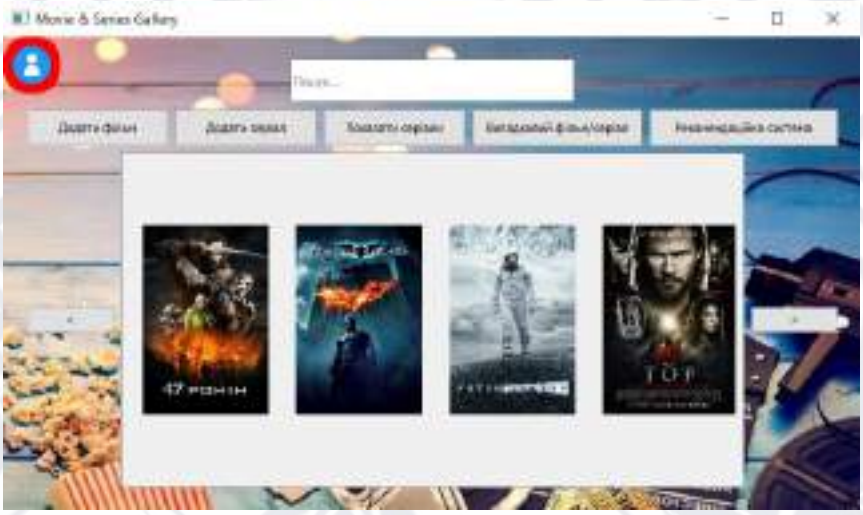

Рисунок 3.29 – Вікно авторизованого адміністратора

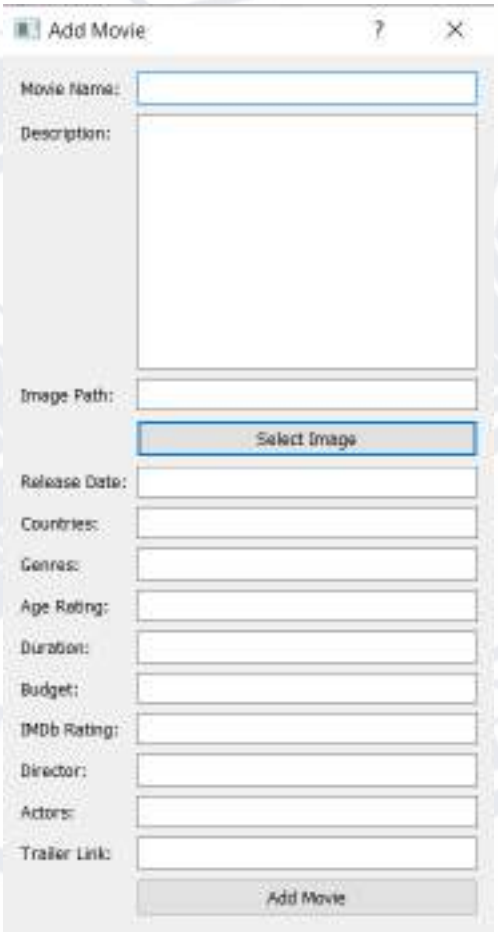

Рисунок 3.30 – Вікно для додавання нового фільму

3.4 Тестування програмного забезпечення

Тест №1. Спроба реєстрації на вже зареєстрований логін.

Запускаємо програму та після відкриття головного вікна, відкриваємо вікно авторизації, на якому клацаємо на кнопку “Реєстрація”, щоб відкрити вікно реєстрації. Спочатку реєструємо перший акаунт, наприклад на логін “bodiken”. Після вдалої реєстрації, виходимо і запускаємо вікно реєстрації ще раз та пробуємо зареєструватись на той самий логін. Результатом буде те, що програма викине наступне повідомлення: “Користувач з таким логіном вже існує” (рис. 3.31).

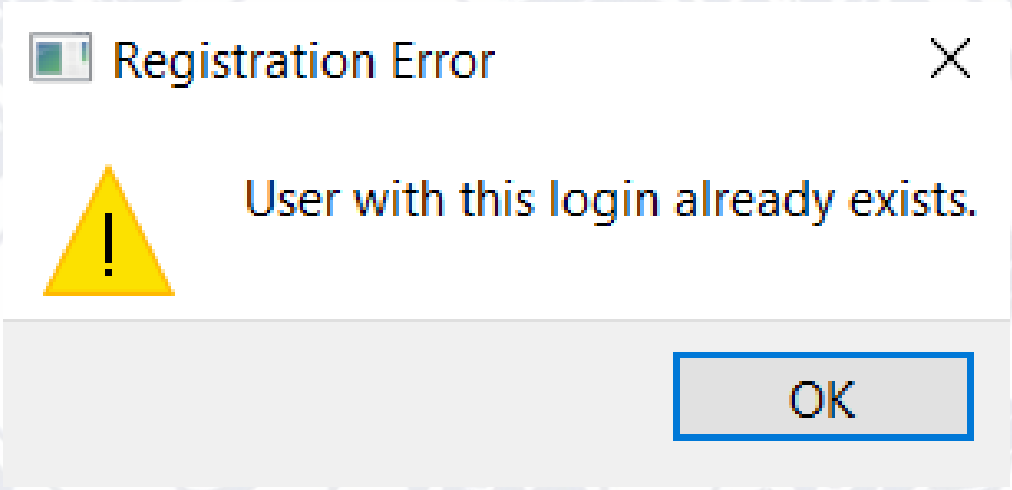

Рисунок 3.31 – Повідомлення про помилку

Тест №2. Додавання нового фільму в галерею

Запускаємо програму та після відкриття головного вікна, авторизуємося за адміністратора. Далі натискаємо кнопку «Додати фільм» та заповнюємо необхідні дані про фільм (рис. 3.32), вибираємо постер, вказуємо посилання на трейлер. Після заповнення всіх необхідних полів, додаємо фільм та перевіряємо чи він був точно доданий. Як видно, галерея оновилася і фільм був успішно добавлений (рис. 3.33).

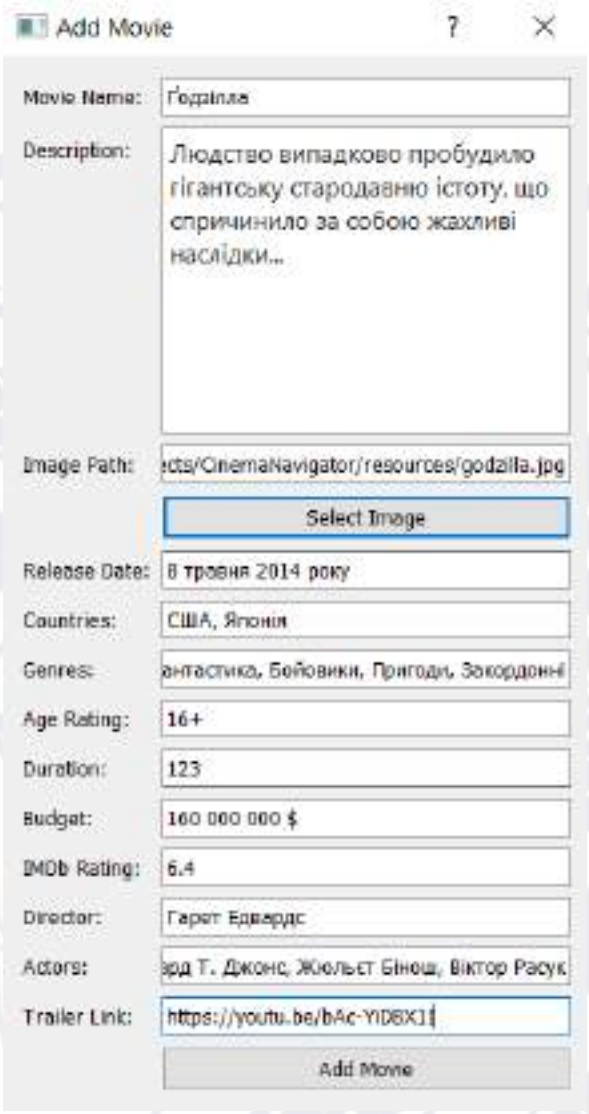

Рисунок 3.32 – Додавання нового фільму

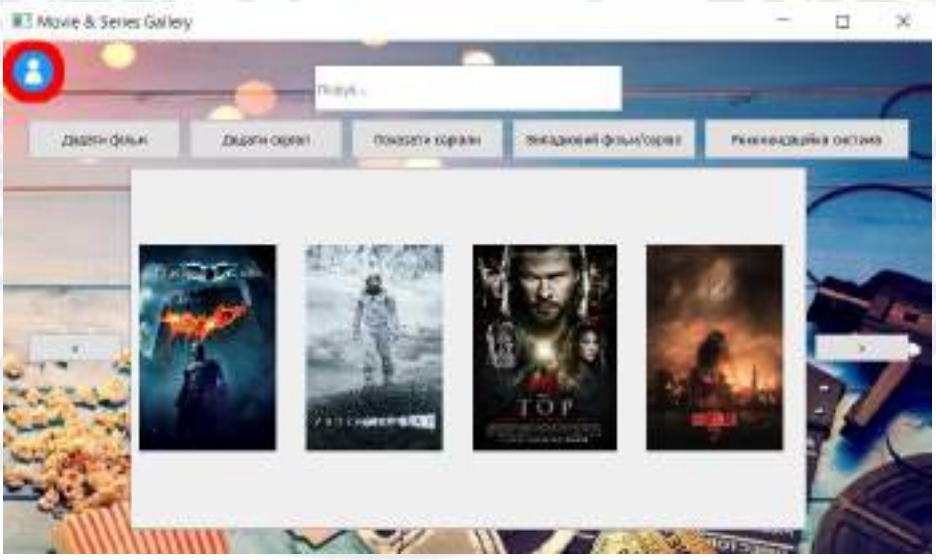

Рисунок 3.33 – Оновлена галерея з новим фільмом

ВИСНОВКИ ДО РОЗДІЛУ 3

У третьому розділі було виконано розробку та тестування прототипу інформаційної системи кінематографічного навігатора. Система була реалізована на мові програмування Python 3.9 у середовищі розробки PyCharmEdu з використанням бібліотек sys, pymongo, PyQt5 та деяких інших.

Було створено базу даних для зберігання інформації про фільми, а також розроблено відповідні класи для додавання нових фільмів, відображення їх детальної інформації та інтеграції з базою даних.

Інтерфейс користувача, виконаний українською мовою, забезпечує зручний доступ до функцій додавання, пошуку та перегляду фільмів. Тестування прототипу показало, що система ефективно виконує поставлені завдання та надає користувачам необхідні інструменти для зручного пошуку та управління інформацією про фільми.

ВИСНОВКИ

Під час дослідження було успішно розроблено та протестовано інформаційну систему кінематографічного навігатора, який стане зручним інструментом для користувачів і дозволить швидко знаходити й отримувати детальну інформацію про фільми, що найбільше відповідають їхнім інтересам і вподобанням. Це завдання передбачало ретельний підхід до аналізу потреб користувачів, а також вивчення технологічних аспектів розробки подібних систем. За результатами дослідження можна зробити наступні висновки:

1. Досліджено теоретичні основи розробки проєктів кінематографічних навігаторів та специфіку їх використання в сучасних умовах. Було визначено ключові аспекти роботи кінематографічного навігатора, його важливість для користувачів у контексті сучасного кінематографу, а також його потенційну роль у покращенні користувацького досвіду. Особлива увага була приділена методам інтелектуального аналізу даних, зокрема розглянуто рекомендаційні системи, які є фундаментом для функціонування подібних платформ.

2. Обґрунтовано вибір інструментів для розробки системи. Розглянуті різноманітні технології та інструменти, які можуть бути використані в процесі створення кінематографічного навігатора.

3. Обґрунтовано вибір мови програмування Python для розробки інформаційної системи на підставі таких характеристик, як-от: універсальність, простота використання та потужні бібліотеки для роботи з даними. Використання бібліотек PyQt5 і pymongo дозволило розробити зручний інтерфейс для користувачів та забезпечити ефективну взаємодію з базою даних MongoDB, що зберігає інформацію про фільми. Вибір цих інструментів був обґрунтований їхньою популярністю та надійністю у сфері розробки інформаційних систем.

4. Було здійснено реалізацію інформаційної системи. Реалізована система надала базовий функціонал, який включає можливість додавання нових фільмів до бази даних, пошуку фільмів за ключовими словами, а також відображення

детальної інформації про кожен фільм, збережений у системі. Проведене тестування підтвердило, що всі основні завдання системи були виконані належним чином, а функціонал відповідав попередньо визначеним вимогам.

5. Розроблений прототип кінематографічного навігатора, забезпечує користувачів необхідними інструментами для швидкого пошуку та управління інформацією про фільми. Це рішення створює основу для подальшого вдосконалення системи, що включатиме розширення функціональних можливостей, поліпшення користувацького інтерфейсу, а також інтеграцію з додатковими джерелами даних для покращення алгоритмів рекомендацій і підвищення загальної ефективності системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ

1. Кинаш Ю. Є., Петрушинський О. О., Мищишин В. М. Розроблення рекомендаційної системи підбору фільмів. Автоматика, Вимірювання та Керування 2019; Випуск 1, Номер 1: сс. 53 – 58.

2. Sambandam Jayalakshmi, Narayanan Ganesh, Robert Čep, Janakiraman Senthil Murugan. Movie Recommender Systems: Concepts, Methods, Challenges, and Future Directions. Sensors 2022, 22(13), 4904.

3. Ramya Vidiyala. How to Build a Movie Recommendation System. URL: https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109

4. Mahesh Goyani, Neha Chaurasiya. A Review of Movie Recommendation System: Limitations,Survey and Challenges. Electronic Letters on Computer Vision and Image Analysis 19(3):18-37, 2020.

5. Ayush Agrawal. MOVIE RECOMMENDATION SYSTEM. Jaypee University of Information Technology. 2019.

6. Anwar T., Uma V. Comparative study of recommender system approaches and movie recommendation using collaborative filtering. International Journal of System Assurance Engineering and Management. 12, 426–436 (2021).

7. Leshchynskyi V. Вдосконалення методу колаборативної фільтрації з неявним зворотнім зв'язком на основі ранжування негативних результатів в матриці вхідних даних / V. Leshchynskyi, I. Leshchynska // Системи управління, навігації та зв'язку. Збірник наукових праць. – Полтава: ПНТУ, 2018. – Т. 3 (49). – С. 73-77.

8. Aditya Bhardwaj, Chirla Rushil Reddy, Palak Arora. Movie Recommendation System Using SVD(Letterboxd). International Journal of Advanced Research in Computer and Communication Engineering Vol. 12, Issue 10, October 2023.

9. Guo, G., Zhang, J., & Yorke-Smith, N. (2015). TrustSVD: Collaborative Filtering with Both the Explicit and Implicit Influence of User Trust and of Item

Ratings. Proceedings of the AAAI Conference on Artificial Intelligence, 29(1).

10. Cunningham, P.; Delany, S.J. K-nearest neighbour classifiers-a tutorial. ACM Comput. Surv. (CSUR) 2021, 54, 1–25.

11. Метод k найближчих сусідів. URL: http://om.univ.kiev.ua/users_upload/15/upload/file/pr_lecture_03.pdf

12. Content-Based Filtering over “Which Movie?”. URL: https://medium.com/cits-tech/content-based-filtering-over-which-movie-37abdaf11019

13. What is content-based filtering? URL: https://www.ibm.com/topics/content-based-filtering

14. Recommendation Systems: Content-Based Filtering. URL: https://medium.com/@zbeyza/recommendation-systems-content-based-filtering-e19e3b0a309e

15. N. Kapoor, S. Vishal and K. K. S., "Movie Recommendation SystemUsing NLP Tools," 2020 5th International Conference onCommunication and Electronics Systems (ICCES), Coimbatore, India,2020, pp. 883-888.

16. G. Geetha, M. Safa, C. Fancy, and D. Saranya, “A hybrid approach using collaborative filtering and content-based filtering for recommender system,” J. Phys. Conf. Ser.,vol. 1000, p. 012101, Apr. 2018.

17. Hwang, S., Ahn, H. & Park, E. iMovieRec: a hybrid movie recommendation method based on a user-image-item model. Int. J. Mach. Learn. & Cyber. 14, 3205–3216, 2023.

18. Yassine Afoudi, Mohamed Lazaar, Mohammed Al Achhab, Hybrid recommendation system combined content-based filtering and collaborative prediction using artificial neural network, Simulation Modelling Practice and Theory, Volume 113,102375.

19. Saurabh Sharma, Harish Kumar Shakya, Hybrid recommendation system for movies using artificial neural network, Expert Systems with Applications, Volume 258, 125-194.

20. Netflix. URL: https://www.netflix.com/ua-ru/browse/genre/839338

21. Як працює система рекомендацій Netflix. URL: https://help.netflix.com/uk/node/100639

22. IMDb. URL: https://www.imdb.com/

23. Letterboxd. URL: https://letterboxd.com/

24. JustWatch. URL: https://www.justwatch.com/

25. Python. URL: https://www.python.org/

26. Костюченко А. О. Основи програмування мовою Python: навчальний посібник. Ч.: ФОП, Баликіна С.М., 2020. 180 с.

27. Путівник мовою програмування Python. URL: https://pythonguide.rozh2sch.org.ua/

28. PyCharm Guide. URL: https://www.jetbrains.com/pycharm/guide/

29. PyCharm як найкраща IDE для розробки ПЗ на Python. URL: https://foxminded.ua/pycharm-tse/

30. PyQt5 Guide. URL: https://www.riverbankcomputing.com/static/Docs/PyQt5/

31. Martin Fitzpatrick, Create GUI Applications with Python & Qt5 (PyQt5 Edition): The hands-on guide to making apps with Python, 2020. 674 p.

32. Mark Summerfield. Rapid GUI programming with Python and Qt, 2007. 588 p.

33. Qt Designer. URL: https://build-system.fman.io/qt-designer-download

34. MongoDB. URL: https://www.mongodb.com/

35. Ситник Н. В., Зінов'єва І. С. Сучасні бази даних NoSQL у підготовці бакалаврів спеціальності "комп'ютерні науки". Інформаційні технології і засоби навчання, 2021, Том 81, №1. С. 255-271.

36. Основи MongoDB. URL: https://codeguida.com/post/519

37. Блок-схеми алгоритму. URL: https://yevshan.com.ua/info/006/content/content3.html

38. Структурні карти Константайна. URL: https://studfile.net/preview/9445425/page:36/

39. Об'єктно-орієнтоване програмування мовою PYTHON: Конспект

лекцій; уклад.: Д. Д. Татарчук, Ю. В. Діденко. Київ : КПІ ім. Ігоря Сікорського, 2021. 129 с.

40. Замуруєва О. В., Кримусь А. С., Ольхова Н. В. Об’єктно-орієнтоване програмування в Python: курс лекцій. Луцьк: Вежа-Друк, 2018. 64 с.

41. Конспект лекцій з дисципліни «Архітектура та проектування програмного забезпечення» – «Інженерія програмного забезпечення» / Укл. В. В. Завгородній, К. М. Ялова. Кам’янське: ДДТУ, 2019.– 144с.

42. Якість та тестування інформаційних систем. Навчальний посібник. Київ: ННІІТ ДУТ, 2020. 128 с.

43. Проектування інформаційних систем: навчальний посібник / В.С. Авраменко, А.С. Авраменко. Черкаси: Черкаський національний університет ім. Б. Хмельницького, 2017. 434 с.

44. Концепція баз даних. Вимоги, що пред'являються до даних. URL: https://stud.com.ua/35671/informatika/kontseptsiya_baz_danih

45. Добролюбова М. В. Програмування баз даних: конспект лекцій. Київ: КПІ ім. Ігоря Сікорського, 2021. 275 с.

46. PyMongo 4.10.1 Documentation. URL: https://pymongo.readthedocs.io/en/stable/

47. Об’єктно-орієнтоване програмування: [Підручник] / В.В. Бублик. – К.: ІТкнига, 2015. 624 с.

48. Креневич А. П. Алгоритми і структури даних. Підручник. – Київ: ВПЦ "Київський Університет", 2021. 200 с.

49. Berkovsky S., Cantador I., Tikk D. Collaborative recommendations: algorithms, practical challenges and applications. (2019). World scientific publishing. https://doi.org/10.1142/11131

50. Авторський GIT репозиторій програми. URL: https://github.com/Bogdan-Lavrenyuk/Qualification-Master-s-thesis.git

ДОДАТКИ

Додаток А

main.py

```
import sys
from PyQt5.QtWidgets import (QApplication, QLabel, QPushButton, QVBoxLayout,
QHBoxLayout, QWidget,
                             QScrollArea, QDialog, QLineEdit, QFileDialog,
QMessageBox, QFormLayout, QTextEdit,
                             QComboBox, QTabWidget)
from PyQt5.QtGui import QPixmap, QIcon
from PyQt5.QtCore import Qt, QSize, QTimer
from pymongo import MongoClient
import os
from bson import ObjectId
import random

# Підключення до MongoDB для фільмів і користувачів
client = MongoClient("mongodb://localhost:27017/")
db = client['movie_gallery']  # База даних 'movie_gallery'
movies_collection = db['movies']  # Колекція для фільмів
users_collection = db['users']  # Колекція для користувачів
series_collection = db['series']


# Шлях для збереження зображень
IMAGE_PATH = "movie_images"

# Перевіряємо чи існує директорія для збереження зображень
if not os.path.exists(IMAGE_PATH):
    os.makedirs(IMAGE_PATH)

class DatabaseManager:
    def __init__(self):
        # Підключення до MongoDB
        self.client = MongoClient("mongodb://localhost:27017/")
        self.db = self.client['movie_gallery']

        # Колекції для закладок
        self.will_watch_collection = self.db['will_watch']
        self.watching_collection = self.db['watching']
        self.not_interested_collection = self.db['not_interested']
        self.movies_collection = self.db['movies']  # Колекція для фільмів

    # Метод для додавання фільму до закладок
    def add_to_bookmarks(self, user_id, user_login, movie_id, category):
        data = {'user_id': user_id, 'user_login': user_login, 'movie_id':
movie_id}
        if category == 'will_watch':
            self.will_watch_collection.insert_one(data)
        elif category == 'watching':
            self.watching_collection.insert_one(data)
        elif category == 'not_interested':
            self.not_interested_collection.insert_one(data)

    # Метод для отримання закладок користувача
    def get_user_bookmarks(self, user_login, category):
        if category == 'will_watch':
            return self.will_watch_collection.find({'user_login': user_login})
        elif category == 'watching':
            return self.watching_collection.find({'user_login': user_login})
        elif category == 'not_interested':
```

```
            return self.not_interested_collection.find({'user_login':
user_login})

    def check_and_move_bookmark(self, user_login, movie_id, new_category):
        # Перевіряємо, чи фільм уже є в одній із закладок
        collections = {
            "will_watch": self.will_watch_collection,
            "watching": self.watching_collection,
            "not_interested": self.not_interested_collection
        }

        current_category = None
        for category, collection in collections.items():
            if collection.find_one({"user_login": user_login, "movie_id":
movie_id}):
                current_category = category
                break

        # Якщо фільм є в іншій закладці
        if current_category and current_category != new_category:
            # Питання користувачу про перенесення
            response = QMessageBox.question(
                None,
                "Перенести фільм",
                f"Фільм уже є в закладці '{current_category}'. Бажаєте
перенести його в '{new_category}'?",
                QMessageBox.Yes | QMessageBox.No
            )
            if response == QMessageBox.Yes:
                # Видаляємо з поточної колекції та додаємо в нову
                collections[current_category].delete_one({"user_login":
user_login, "movie_id": movie_id})
                collections[new_category].insert_one({"user_login": user_login,
"movie_id": movie_id})
                return True  # Означає, що фільм перенесено
            else:
                return False  # Користувач відмовився від перенесення

        # Якщо фільм ніде не знайдено, можна просто додати
        if not current_category:
            collections[new_category].insert_one({"user_login": user_login,
"movie_id": movie_id})
            return True  # Означає, що фільм додано новий

        return False  # Фільм вже у потрібній закладці

    # Метод для отримання фільму за ID
    def get_movie_by_id(self, movie_id):
        return self.movies_collection.find_one({"_id": movie_id})  # Повертаємо
документ фільму


class RegistrationDialog(QDialog):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setWindowTitle("Registration")
        layout = QFormLayout()

        self.first_name_input = QLineEdit(self)
        self.first_name_input.setPlaceholderText("Enter first name")
        self.last_name_input = QLineEdit(self)
        self.last_name_input.setPlaceholderText("Enter last name")
        self.login_input = QLineEdit(self)
```

```
        self.login_input.setPlaceholderText("Enter login")
        self.password_input = QLineEdit(self)
        self.password_input.setPlaceholderText("Enter password")
        self.password_input.setEchoMode(QLineEdit.Password)

        layout.addRow("First Name:", self.first_name_input)
        layout.addRow("Last Name:", self.last_name_input)
        layout.addRow("Login:", self.login_input)
        layout.addRow("Password:", self.password_input)

        register_button = QPushButton("Register")
        register_button.clicked.connect(self.register_user)
        layout.addWidget(register_button)

        self.setLayout(layout)

    def register_user(self):
        first_name = self.first_name_input.text()
        last_name = self.last_name_input.text()
        login = self.login_input.text()
        password = self.password_input.text()

        if first_name and last_name and login and password:
            # Перевіряємо чи користувач вже існує
            if users_collection.find_one({"login": login}):
                QMessageBox.warning(self, "Registration Error", "User with this
login already exists.")
            else:
                users_collection.insert_one({
                    "first_name": first_name,
                    "last_name": last_name,
                    "login": login,
                    "password": password
                })
                QMessageBox.information(self, "Registration Success",
"Registration successful!")
                self.accept()
        else:
            QMessageBox.warning(self, "Input Error", "Please fill all fields.")
                                        ...
```

ДЕКЛАРАЦІЯ

про дотримання академічної доброчесності

Я, _____

_____

_____

_____

_____

Повністю вказується ПІБ та статус (освітня (освітньо-наукова) програма – для здобувачів вищої освіти, назва кваліфікаційної роботи)

що нижче підписалась/підписався, розуміючи та підтримуючи загальновизнані засади справедливості, доброчесності та законності,

ЗОБОВ'ЯЗУЮСЬ:

дотримуватися принципів та правил академічної доброчесності, що визначені законодавством України, локальними нормативними актами Донецького національного університету імені Василя Стуса, положеннями, правилами, умовами, визначеними іншими суб'єктами, та не допускати їх порушення.

ПІДТВЕРДЖУЮ:

що мені відомі положення статті 42 Закону України «Про освіту»;

що у даній роботі не представляла/представляв чиїсь роботи повністю або частково як свої власні. Там, де я скористалася/скористався працею інших, я зробила/зробив відповідні посилання на джерела інформації;

що дана робота не передавалась іншим особам і подається вперше, не порушує авторських та суміжних прав закріплених статтями 21-25 Закону України «Про авторське право та суміжні права», а дані та інформація не отримувались в недозволений спосіб.

УСВІДОМЛЮЮ:

що ця робота може бути перевірена університетом на плагіат або інші порушення академічної доброчесності, в тому числі з використанням спеціалізованих сервісів;

що у разі порушення академічної доброчесності, до мене можуть бути застосовані процедури, передбачені законодавством України та Кодексом академічної доброчесності та корпоративної етики Донецького національного університету імені Василя Стуса, іншими локальними нормативними актами університету, та я можу бути притягнута/притягнутий до академічної відповідальності.

_____ _____

(дата) (підпис)