Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Центр післядипломної освіти

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система онлайн кінотеатру
(тема)

Виконав:
студент 2 курсу, групи ПЗПІп-22-2

Душко А.В.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ст.викл. кафедри ПІ Олійник О.В.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Центр післядипломної освіти
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Душко Артему Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмна система онлайн кінотеатру
Затверджена наказом по університету від «17» червня 2024 р. № 588 Ст
2. Термін подання студентом роботи до екзаменаційної комісії «22» липня 2024р.
3. Вихідні дані до роботи у програмній системі передбачити: операції з даними відеокліпів, пошук за жанром та ключовий пошук, реєстрацію та авторизацію у системі, адміністрування контенту та користувачів, підрахунок та відображення рейтингу для кожного відеокліпу, вибір та додавання контенту у власний список перегляду, редагування власного списку перегляду, локалізацію інтерфейсу англійською мовою. Використовувати ОС Windows 11, СРБД MySQL 8.0.36 Community Server, середовище розробки Microsoft Visual Studio v.1.90.1.
4. Перелік питань, що потрібно опрацювати в роботі аналіз предметної галузі, огляд існуючих рішень, концептуальне моделювання предметної області, варіанти використання, архітектура системи, структура даних, описання програмної реалізації системи.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін	Відмітки про виконання
1	Інструктаж з безпеки життєдіяльності	18.06.2024 р.	*виконано*
2	Аналіз предметної області	18.06 – 19.06	*виконано*
3	Розробка постановки задачі	19.06 – 21.06	*виконано*
4	Формування вимог до ПЗ	24.06 – 26.06	*виконано*
5	UML проектування	26.06 – 28.06	*виконано*
6	Проектування архітектури ПЗ	28.06 – 03.07	*виконано*
7	Розробка структур зберігання даних	03.07 – 05.07	*виконано*
8	Проектування прототипів інтерфейсів	05.07 – 09.07	*виконано*
9	Підготовка звіту з роботи	10.07 – 12.07	*виконано*
10	Захист роботи	22.07	

Дата видачі завдання «*18*» червня 2024 р.

Студент _____ Артем ДУШКО
(підпис)

Керівник роботи _____ ст. викл. каф. ПІ Олійник О.В.
(підпис)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра: 66 с., 26 рис., 0 табл., 7 джерел.

ВЕБ-СИСТЕМА, ДОЗВІЛЛЯ, ЖАНР, КІНО, ОНЛАЙН КІНОТЕАТР, ПІДПИСКА, РЕЙТИНГ, BOOTSTRAP, DJANGO, JAVASCRIPT, MYSQL 8.0, PYTHON.

Об’єкт розробки – клієнтська та бекенд частина комплексного програмного забезпечення, яке автоматизує процес реєстрації, підписки, пошуку кінофільмів, створення власного списку для подальшого перегляду та присвоєння рейтингу на основі вражень. Система допоможе потенційним користувачам якісно проводити дозвілля за переглядом улюблених кінофільмів у зручному місці та у зручний час.

Об’єктом дослідження є створення клієнт орієнтованого рішення для планування й перегляду кінофільмів через розробку кабінету переглядача з налаштуваннями персоналізації.

Методи рішення – аналіз предметної області та вимог користувачів, концептуальне моделювання, UML-моделювання, створення веб-клієнта для системи з використанням мов програмування Python JavaScript, веб фреймворку Django та CSS фреймворку Bootstrap.

Кінцевим продуктом роботи стане клієнтське програмне забезпечення зі зручним користувацьким інтерфейсом. Користувачі зможуть використовувати систему для створення власного облікового запису та власної колекції відео контенту обираючи з загальної бібліотеки програмної системи.

WEB SYSTEM, ENTERTAINMENT, CINEMA, ONLINE CINEMA, SUBSCRIPTION, RATING, GENRE, BOOTSTRAP, DJANGO, JAVASCRIPT, PYTHON, MYSQL 8.0.

The object of development is both the client and background components of comprehensive software that automates the process of registration, subscription, movie search, creation of a personalized list for future viewing, and assigning a rating based on impressions. The system aims to help potential users enjoy their favorite movies conveniently and at their own leisure.

The focus of the research is to create a client for a targeted solution for planning and watching movies, through the development of a Viewing Cabinet with personalized settings.

The methods for this solution involve an analysis of the subject area and user requirements, conceptual modeling, UML modeling, and creating a web client for the Python JavaScript system, using Django and CSS Framework.

The final product will be client software with a user-friendly interface. Users will be able to use the system to create their own account and compile their own collection of video content, chosen from the general library of the software system.

Я, Душко Артем Володимирович, студент гр. ПЗПІп-22-2, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система онлайн кінотеатру», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

Враховуючи те, що зі зростом науки та прогресу повсякденне життя пришвидшується, дещо еволюціонує не тільки наша робота, а і дозвілля. Кінотеатри переходять у онлайн режим, додаючи гнучкості у наш графік та комфорту, що дозволяє відпочивати більш якісно, тому онлайн кінотеатр є досить актуальною темою. Онлайн кінотеатр – це платформа, на якій користувачі можуть переглядати відеоматеріали, будь де та будь коли, використовуючи за наявності інтернет-з'єднання. Цілі таких платформ можуть варіюватися від надання розважального контенту до надання документальних та інформаційних матеріалів. Основна мета – покращити якість дозвілля людей і зробити їхнє життя цікавішим [1].

Користувачі можуть отримати велику естетичну та рекреаційну користь від онлайн кінотеатрів, а більше розмаїття таких платформ можуть задовольнити потреби більшої кількості людей. Однак створення та підтримка онлайн-кінотеатру може бути складним завданням.

Метою цього проекту є створення клієнтської частини програмного забезпечення, яке автоматизує пошук, відбір та планування контенту для онлайн-кінотеатру, щоб підвищити ефективність, швидкість та доступність. Користувачі зможуть створювати індивідуальні колекції фільмів, або серіалів для подальшого перегляду, а система рейтингу допоможе поділитися загальними враженнями від перегляду, що допоможе у підборі найкращих матеріалів іншим. Модератори зможуть редагувати та класифікувати контент за жанрами та тегами, а користувачі зможуть переглядати доступний контент та давати відгуки.

Результати дослідження можуть бути актуальні для широкого кола галузей та організацій, включаючи кіностудії, студії перекладу, рекламні агентства та навчальні заклади.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Онлайн кінотеатр – це один з способів пасивного відпочинку, де під час перегляду кінофільмів людина відпочиває ментально та фізично. Основними передумовами є наявність пристрою для перегляду, та стабільного інтернет з'єднання, що відповідає вимогам для комфортної взаємодії з програмною системою.

Через розмаїття контенту та диверсифікацію онлайн кінотеатрів за специфікою контенту, кожен може обрати ресурс за вподобаннями та відповідно до власних цілей, наприклад художні фільми, або документалістика.

Привабливість онлайн кінотеатрів полягаю у тому, що більше немає потреби у особистому фізичному відвідуванні за фіксованим часом сеансу. А також більше немає досить вузького переліку опції, що притаманні класичним кінотеатрам, оскільки більше немає прив'язки до кінозалів. Людям подобається самим вирішувати який контент обрати, коли і де подивитись, або ж навіть розбити сеанс перегляду на частини. А наявність широкого переліку субтитрів та звукових доріжок різними мовами задовольняє освітні вимоги з вивчення іноземних мов, та культурного збагачення.

Серед найвідоміших так званих Over-The-Top (OTT) сервісів, тобто таких, які забезпечують поточний контент користувачам через мережу інтернет можна виділити Netflix, Amazon Prime, Disney+ тощо. Ці сервіси пропонують сотні тисяч годин контенту та знаходяться в центрі змін у моделях споживання відео. Послуги OTT мають значний економічний вплив, про що свідчить, наприклад, той факт, що розмір світового ринку OTT оцінювався в 121,61 мільярда доларів США в 2019 році та, за прогнозами, досягне 1039,03 мільярда доларів США до 2027 року, зростаючи з річним темпом зростання на 29,4% з 2020 по 2027 рік. [1]

Прогноз росту ринку сервісів поточного відео загалом, показано на рисунку 1.1.

Провідні учасники ринку також використовують технології блокчейну, машинного навчання та штучного інтелекту (AI) для покращення якості відео.

Кінематограф, редагування відео та голос за кадром, написання сценаріїв та інші аспекти створення та завантаження відео – усе це стає легшим завдяки цим технічним розробкам. Крім того, вони допомагають у організації, кодуванні та розподілі даних, спрощуючи цифрове середовище. Позитивний ринковий прогноз створюється завдяки цьому, а також зростаючому прийняттю хмарних рішень. [2]

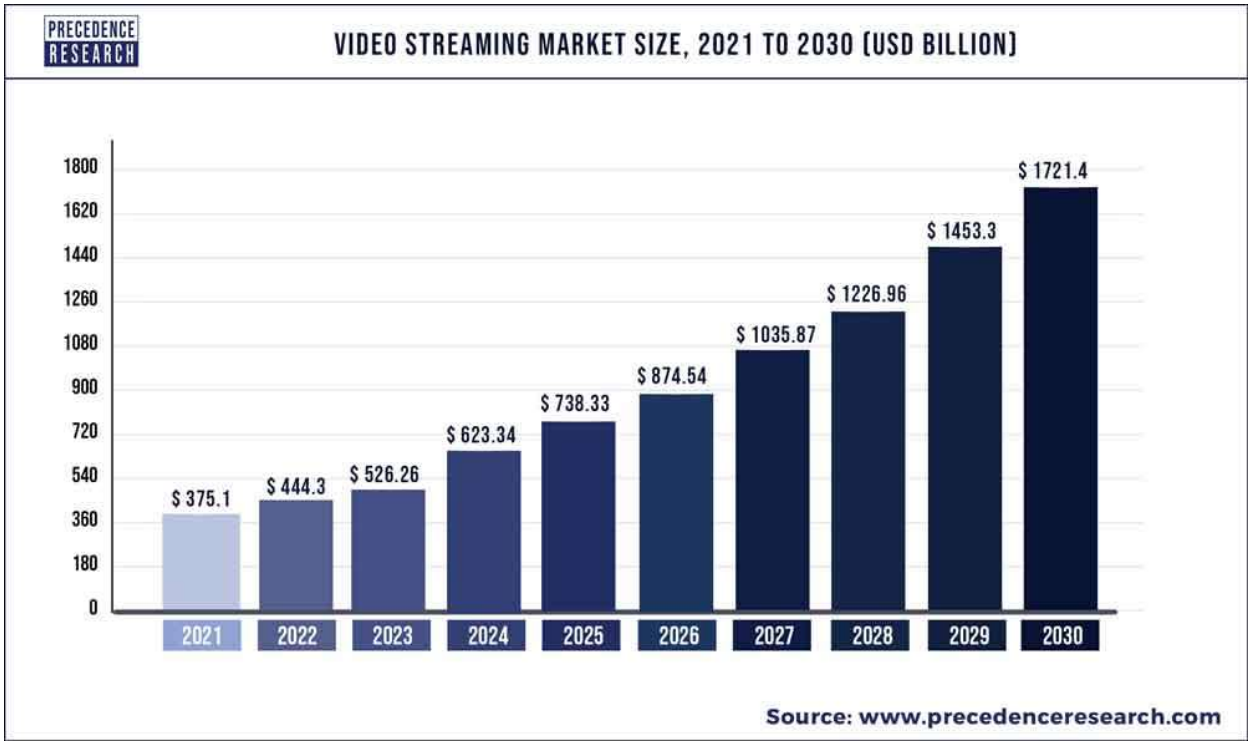


Рисунок 1.1 – Прогноз росту ринку сервісів поточного відео до 2030 р. [2]

Поширення потокових онлайн-платформ революціонізувало медіаландшафт, спонукаючи науковців до багатогранного впливу на різні аспекти суспільства. Науковці детально проаналізували економічні наслідки потокових онлайн-платформ, проливаючи світло на їхній руйнівний вплив на бізнес-моделі традиційних медіа. Джонсон (2018) та Чжан і Ван (2021) підкреслили, як потокові послуги змінили потоки доходів, що призвело до зниження доходів від реклами та тенденцій урізання кабелів, а також створило нові можливості для отримання прибутку через моделі на основі підписки. Крім того, дослідження підкреслюють наслідки ринкової конкуренції та стратегії ціноутворення в галузі потокового передавання.

Онлайн-платформи для потокового передавання є ключовими у формуванні суспільних норм, культурних наративів і споживчої поведінки. Лі та Кім (2019) обговорювали, як ці платформи сприяли міжкультурному обміну та кинули виклик гегемоністським уявленням у традиційних медіа, сприяючи різноманітності та інклюзивності. Надійні політичні рамки необхідні через численні правові та етичні міркування, які позначають нормативний ландшафт навколо потокових онлайн-платформ. Сміт (2019) досліджував напругу між регулюванням контенту та свободою вираження поглядів, наголошуючи на таких питаннях, як цензура, порушення авторських прав і культурна чутливість. Крім того, Chen et al. (2020) досліджував регуляторні проблеми, пов'язані з конфіденційністю даних, захистом споживачів і домінуванням на ринку, виступаючи за проактивні заходи для захисту прав користувачів і забезпечення чесної конкуренції. [1]

Ключовими рушіями ринку дослідження виділяє наступні:

- популярність потокового відео: одним із найкращих аспектів, який допомагає підвищити ринкову вартість індустрії потокового відео, є пряме передавання. Ця функція використовується для розвитку компанії шляхом безпосереднього спілкування з цільовим ринком. Щоб у них не було проблем із розумінням питань, які стосуються питань бізнесу. Останнім часом це стало джерелом збільшення прибутку. Завдяки своєму положенню на вершині. Таким чином, на світовому ринку потокового відео останнім часом спостерігається зростання потокового відео в прямому ефірі;
- розвиток технологій: розвиток ринку потокового відео став можливим завдяки останнім технологічним досягненням, таким як штучний інтелект і технологія блокчейн. Люди зацікавлені у використанні потокового відео в результаті розвитку технологій, тому що вони віддають перевагу цьому, і, як наслідок, якість відео підвищується. Останнім часом це стало головним фактором зростання ринку потокового відео.

Ключовим викликом є зростаюча стурбованість щодо піратства та захисту вмісту. Очікується, що зростаюче занепокоєння користувачів щодо піратства та захисту вмісту перешкоджатиме діловим операціям через зниження інтересу клієнтів до вмісту та споживання ним. Очікується, що в наступні роки це вплине на зростання ринку. Наприклад, спільне дослідження Digital Citizen Alliance у серпні 2020 року показало, що вартість піратських підписок лише в США становить мільярди доларів. У США близько 9 мільйонів користувачів Інтернету підписані на піратського провайдера IPTV. Для доступу до цих послуг користувачі використовують щонайменше 3500 неліцензійних веб-сайтів і сторінок у соціальних мережах.

Ключовими можливостями є:

– вплив потокового відео в освітньому секторі. Відео може допомогти учням запам’ятати матеріал ефективніше, що може мати величезний позитивний вплив на освіту. До цієї категорії входять, серед іншого, вебінари та відеолекції. Освітні організації, включаючи університети, школи та коледжі, тепер створюють інтерактивну інформацію та поширюють її за допомогою відеопрезентацій. Таким чином вони ефективно використовують технології для поширення знань. Обов’язки вчителів суттєво змінюються внаслідок зростання попиту на фільми, які можна використовувати в класі. Останній тепер може викладати за допомогою відео у віртуальних класах по всьому світу та використовувати добірку фільмів у класі. У результаті використання фільмів у класі допомагає створити мультимодальне навчальне середовище;
– підвищений попит на високошвидкісне підключення до Інтернету. Зростаюча потреба у високошвидкісному доступі до Інтернету вплинула на зростання потреби в потоковому відео. Крім того, у секторі навчання відбулися швидкі зміни, наприклад, у 2020 році Microsoft придбала Affirmed Networks. Affirmed Networks є постачальником хмарних мережевих рішень для операторів зв’язку зі штаб-квартирою в

Сполучених Штатах. Microsoft очікує, що ця співпраця допоможе у створенні технологій 5G і периферійних обчислень. Як наслідок, протягом прогнозованого періоду нові альянси та технічні досягнення рухатимуть вперед ринок послуг потокового відео. [2]

1.2 Аналіз існуючих рішень

Проведено пошук та аналіз деяких найбільших за аудиторією, функціоналом та кількістю доступного контенту програмних систем онлайн кінотеатрів. До набільших та найвідоміших на даний момент сервісів можна віднести Netflix, Amazon Prime та Disney+. Кожен з наведених конкурентів має свої специфічні особливості, переваги й недоліки відносно інших.

Програмна система Netflix (https://netflix.com) – це один з найбільших сервісів для стрімінгу відео, який пропонує широкий спектр фільмів, серіалів, документальних фільмів та оригінального вмісту (див. рис. 1.2).

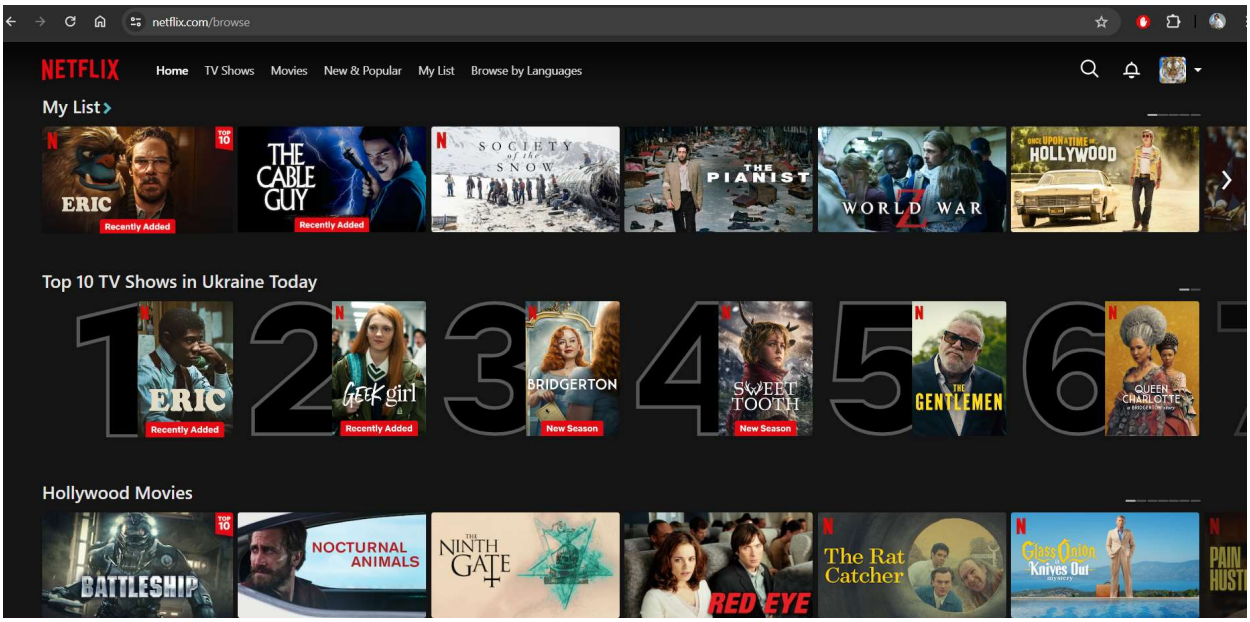


Рисунок 1.2 – Веб інтерфейс програмної системи Netflix [2]

Netflix доступний у більшості країн світу. Сервіс розпочався як нішеве відгалуження сервісу розсилки дисків компанії, та поступово став набагато більше, з надійним основним каталогом і чудовим оригінальним програмуванням. Він працює практично на будь-якому пристрої, дозволяє завантажувати офлайн

на мобільні платформи та містить вміст 4K. А опція з підтримкою реклами тепер допомагає йому краще конкурувати з більш доступними конкурентами. Завдяки різноманітному, першокласному вмісту та потужним програмам Netflix став переможцем вибору серед багатьох редакцій на кшталт PCMag.

Конкурентні особливості:

- дуже велика кількість контенту, включаючи старі й нові фільми, серіали, документальні фільми;
- власна студія, що знімає дуже якісний контент;
- дуже широкий вибір звукових доріжок та субтитрів багатьма мовами світу;
- безкоштовні мобільні ігри;
- можливість дивитись контент у оффлайн режимі;
- якісна програмна реалізація.

Переваги:

- широкий вибір контенту;
- має ексклюзивні серіали та документальні фільми;
- якісна система пошуку та рекомендацій на основі переглянутих відео;
- якісний переклад мови професійними студіями.

Недоліки:

- вартість підписки може бути вищою, ніж у інших сервісів;
- перелік контенту може варіюватися в залежності від регіону;
- деякий контент має строк годності, тобто виключається з доступу через певний проміжок часу.

Програмна система Amazon Prime (https://www.primevideo.com/) – це основний конкурент Netflix та є одним з найбільших сервісів для стрімінгу відео (див. рис. 1.3). Хоча Amazon Video спочатку був місцем для покупки та перегляду телевізійних шоу та фільмів, згодом він охопив світ необмеженої трансляції у форматі «шведського столу» за єдину місячну плату.

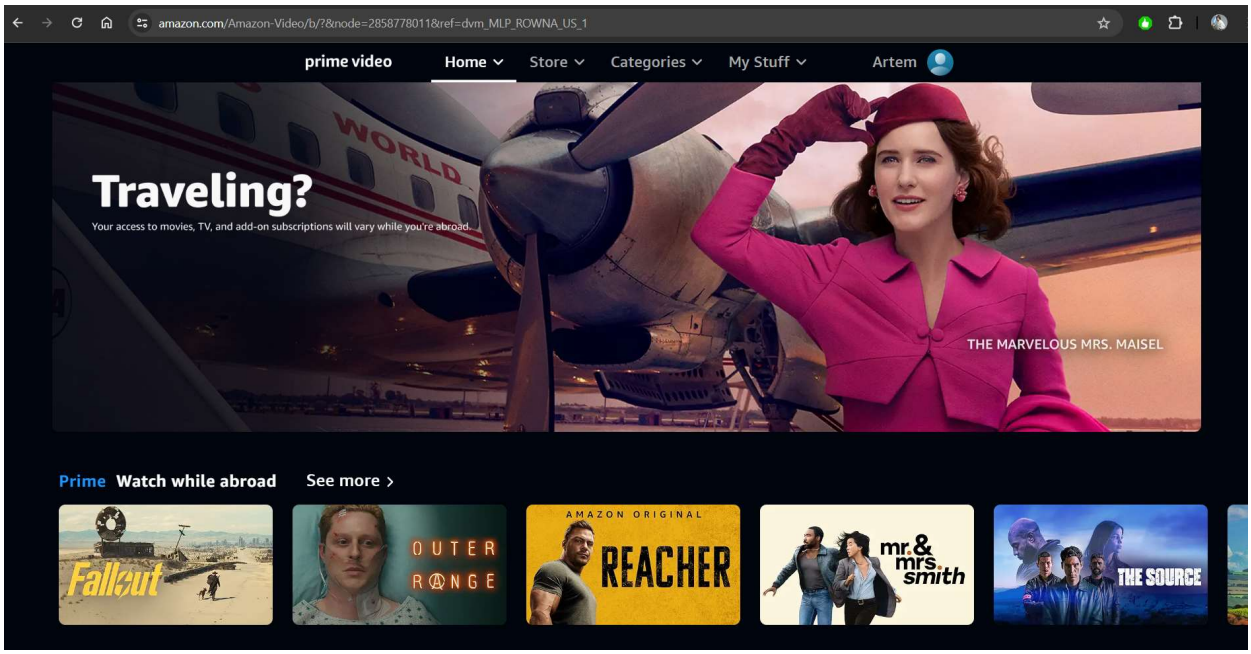


Рисунок 1.3 – Веб інтерфейс програмної системи Amazon Prime [2]

Конкурентні особливості:

– велика кількість контенту, включаючи можливість оренди чи разового перегляду;
– власна студія, що знімає дуже якісні розважальні програми та серіали;
– широкий вибір звукових доріжок та субтитрів багатьма мовами світу;
– онлайн трансляція ігор Night Football та WNBA щочетверга;
– можливість дивитись контент у оффлайн режимі;
– додаткові переваги для підписників онлайн кінотеатру, такі як безкоштовна та пришвидшена доставка з онлайн магазину компанії.

Переваги:

– невелика вартість підписки;
– має ексклюзивні серіали та документальні фільми розважального характеру;
– знижки на доставку товарів з власного магазину.

Недоліки:

– невелика кількість художніх фільмів;
– поступово втрачає контент, через «перетягування» конкурентами;

– кількість розважальних шоу дещо меньша ніж у конкурентів.

Програмна система Disney+ (https://www.disneyplus.com/) – є одним з найбільших сервісів для стрімінгу відео (див. рис. 1.4). На відміну від програмних систем розглянутих віще, Disney+, через величезне розмаїття мультиплікаційних фільмів та шоу, має скерованість на задоволення потреб молодої аудиторії.

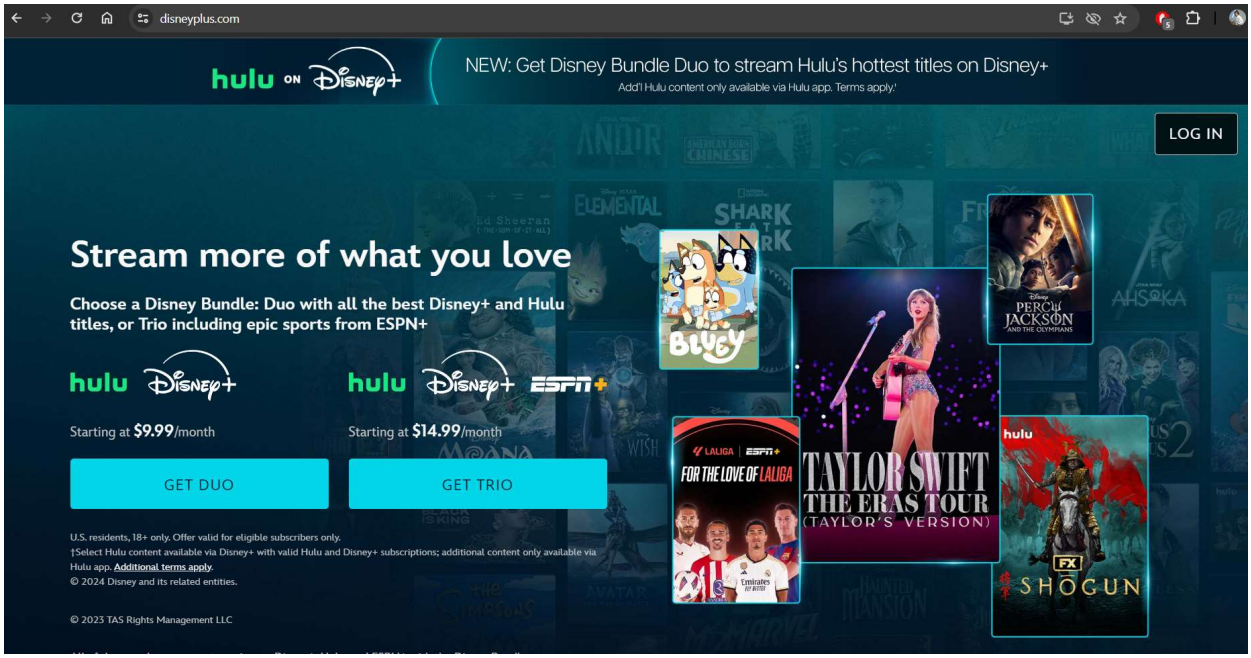


Рисунок 1.4 – Веб інтерфейс програмної системи Disney+[2]

Конкурентні особливості:

– величезна бібліотека мультиплікаційного контенту;
– власна студія, в тому числу студія анімації, що виробляє якісний контент до появи програмної системи та є на даний момент ексклюзивом, що не поширюється на конкурентів;
– центральний хаб, для пошуку та перегляду всих творінь студії Disney.

Переваги:

– величезна кількість мульфільмів та шоу виробництва одноіменної студії;
– контент у якості 4К;
– гнучкі тарифні плани.

Недоліки:

– наявність реклами на дешевих тарифних планах;

- Hulu контент потребує доплати;
- можливість offline перегляну доступна тільки певним тарифним планам;
- користування платформою для деяких регіонів, до яких належить і Україна, не доступно.

Головним та загальним недоліком цих платформ є зосередженість лише на відомих, або власних телесеріалах та кінофільмах, тобто продукти від арт хауз і маловідомих студій як правило оминаються. Також цінова політика могла б бути більш доступною для користувачів поза територією США.

Виходячи зі специфіки теми, можна виділити наступні цільові аудиторії, тобто групи людей, що можут бути зацікавлені у використанні програмної системи онайн вікотеатру:

- загальна категорія людей молодшого, середнього та старшого віку, що цікавляться фільмами різниж жанрів та серіалами, але надають перевігу домашньому перегляду з гнучкими налаштуваннями сеансів;
- сімейні пари, в тому числі з дітьми, що полюбляють проводити вільний час за переглядом кінофільмів;
- домогосподарки та домогосподарі, що полюбляють дивитись серіали чи кінофільми у фоновому режимі, під час виконання домашніх справ;
- заклади освіти, що зацікавлені у використанні документального видеоконтенту в програмі навчання, а також студенти та аспіранти;
- студії звукозапису, що надають послуги перекладу звуку та тексту на різні мови. Вони можут співпрацювати з власникмо програмної системи для росширення покриття цільової аудиторії та відповідно регіонів;
- маловідомі кіностудії, що зацікавленні в пошуку глядачів для своїх кінострічок через подібні стримінгові сервіси;
- студії, продюсери та режисери, що збирають статистику та преференції глядачів у певний проміжок часу, за для визнаення тем та жанрів для майбутніх зйомок.

Основна мета системи – поєднати та реалізувати інтереси визначених груп, що у перспективі сприятиме росту і розвитку сервісу. З одного боку платформа

допоможе популяризувати й рости малим, але перспективним студіям і акторам, з іншого допоможе знімати та надавати саме той контент, що потребують глядачі, та забезпечить його різними мовами для широкого покриття.

1.3Постановка задачі

Надання якісного сервісу користувачам, його відповідність очікуванням, та конкуреноспроможність, мають вирішальне значення і сучасному світі, де технології безперервно проникають та змінюють наше життя.

Основна мета – створити сучасний, але самодостатній сервіс, програмну систему онлайн кінотеатру. Система матиме не складний, але продуктивний, швидкий інтерфейс користувача. Запропоноване рішення доможе користувачам швидко знайти бажаний контент та зосередитись на перегляді, без зайвих кроків. Розробка планується у декілька етапів, що будуть поступово нарощувати й росширювати можливості програмною системи.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

2.1Функціональні вимоги до програмної системи

Перший реліз системи включатиме 2 ролі – адміністратор та користувач, а також міститиме серверну та клієнтську частини.

Роль адміністратора має на меті функцію наповнення та редагування основним контентом системи, а також нагляду та модерації існуючих користувачів.

Роль користувача це основна роль, що є споживачем системи. Роль повинна мати доступ до зручних інтерфейсів з реєстрації, авторизації, оформлення підписки, пошуку, збереження, перегляду та оцінювання відео контенту.

Серверна частина відповідатиме за систему членства за допомогою реєстрації та авторизації, розподілення доступу за допомогою ролей, оброблятиме основну логіку та зберігатиме дані клієнтів та сервісу.

Основні функції серверної частини:

- реєстрація користувача;
- авторизація користувача;
- авторизація адміністратора;
- редагування профілю авторизованого користувача;
- підрахунок клієнтського рейтингу відео контенту;
- збереження індивідуального списку перегляду користувача;
- оформлення підписки на сервіс користування та перегляду;
- відстеження статусу підписки користувача;
- поновлення підписки користувача;
- фільтрація та пошук контенту за параметрами та генерування результатів.

Клієнтська частина надаватиме зручний та безпечний інтерфейс для взаємодії із системою.

Основні функції клієнтської частини:

- відображення даних профілю користувача;

- відображення статусу підписки;
- інтерфейс поновлення підписки;
- інтерфейс пошуку відео контенту за категоріями;
- інтерфейс пошуку відео за ключовими словами;
- інтерфейс роботи з власним списком перегляду;
- відображення детальної інформації про обране відео;
- інтерфейс перегляду відео.

У Django клієнтська частина отримує дані з серверної частини за допомогою представлень (views) і шаблонів (templates).

Цей процес можна описати наступними кроками:

- клієнт (веб-браузер) надсилає HTTP-запит до програми на Django;
- URL диспетчер порівнює URL-адресу запиту з функцією перегляду, на яку посилається адреса;
- функція перегляду також взаємодіє з моделлю (яка представляє структуру бази даних), та за необхідності робить запити. Це може включати запит до бази даних для отримання конкретних даних, або запис даних до бази;
- представлення передає ці дані в шаблон. Шаблони в Django — це HTML-файли, які мають заповнювачі (placeholders) для даних, які потрібно вставити;
- шаблон відтворює HTML-сторінку з даними, наданими представленням;
- нарешті ця HTML-сторінка потім надсилається назад клієнту як HTTP відповідь.

2.2Нефункціональні вимоги до програмної системи

Простота та зручність. Інтерфейс користувача повинен бути простим та зручним для використання. Всі основні функції, такі як пошук, перегляд каталогу, управління підпискою, повинні бути легко доступні та інтуїтивно зрозумілі.

Адаптивний дизайн. Інтерфейс повинен бути адаптивним, щоб користувачі могли комфортно користуватися платформою на різних пристроях, таких як настільні комп'ютери, ноутбуки та планшети у горизонтальному розташуванні екрану.

Візуальна привабливість. Дизайн інтерфейсу повинен бути візуально привабливим та професійним, щоб заохотити користувачів до використання платформи. Водночас необхідно дотримуватись принципу мінімалізму та не перевантажувати його візуальними елементами.

Інформативність. Інтерфейс повинен надавати користувачам всю необхідну інформацію про відео, включаючи опис, рейтинг, тривалість, вікові обмеження тощо.

Легкість навігації. Система має містити логічну та нескладну структуру з метою легкої навігації в межах ресурсу користувачем.

Відсутність помилок. Привабливість системи прямо залежить від її надійності та плавності роботи. Система не має містити істотних програмних помилок, що заважатимуть взаємодії з ресурсами проекту.

Безпека. Доступ до ресурсів системи повинен забезпечувати безпеку даних користувача. Доступ до особистого профілю та вподобань має бути захищено системою аутентифікації.

2.3Вимоги до середовища розгортання

При розробці онлайн-кінотеатру можна виділити наступні вимоги до середовища розгортання.

Розробка програмного забезпечення виконується за допомогою Microsoft Visual Studio Code. Це зручне середовище розробки програмного забезпечення з цілою низкою додаткових плагінів, що полегшують та прискорюють програмування через наприклад автоматичну перевірку коду на помилки, певну автоматизацію та інші можливості. Частиною програми є термінал, що є дуже зручним для запуску програм з метою перевірки, або роботою з git сховищем. VS Code підтримує багато мов програмування та фреймворків за замовчуванням, або

через розширення. Цей редактор має вбудовану підтримку для багатьох мов, включаючи Python, JavaScript, PHP і багато інших.

В якості бази даних та СУБД має бути використано MySQL. Це одна з найпопулярніших відкритих систем управління реляційними базами даних (RDBMS). Вона використовується великим числом веб-додатків, включаючи такі, що розроблено на базі Django фреймворку.

До основних переваг можна віднести наступні: Підтримує стандартну мову SQL для управління даними;

- підтримує індексацію для швидкого пошуку даних;
- управління транзакціями через підтримку ACID (Atomicity, Consistency, Isolation, Durability), що гарантує надійність транзакцій;
- не потребує додаткових коштів за використання (ліцензію).

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1UML проєктування програмного забезпечення

Компоненти системи реалізуються з використанням технологій і бібліотек описаних нижче.

Django framework це високорівневий веб-фреймворк для розробки веб-додатків на мові програмування Python [4]. Він допомагає швидко та ефективно розробляти веб-сайти та веб-додатки, надаючи розробникам широкий набір інструментів та вбудованих компонентів. Це веб-фреймворк «з коробки», що містить власні Frontend та API server компоненти. Крім того він підтримує та легко інтегрується з «MySQL» базою даних.

Функціональна структура проекту Django дозволяє розділяти задачі та відповідальність між окремими компонентами. Це сприяє створенню модульного коду та полегшує розробку та супровід веб-додатків.

Python – це високорівнева, інтерпретована, багатоцільова, об'єктно-орієнтована мова, що означає, що виконується безпосередньо, без необхідності компіляції.

Можна визначити наступні переваги цієї мови програмування:

- Python має простий синтаксис, який легко читається і зрозумілий, навіть для новачків;
- має велику стандартну бібліотеку, яка включає модулі для різноманітних задач;
- має активну спільноту, яка постійно розробляє та підтримує додаткові бібліотеки та інструменти;
- підтримує різні парадигми програмування, включаючи процедурне, об'єктно-орієнтоване та функціональне програмування;
- широко використовується в наукових обчисленнях, штучному інтелекті, машинному навчанні, веб-розробці, автоматизації та багатьох інших областях [4].

Bootstrap 5 – це найновіша версія популярного відкритого фреймворку для розробки адаптивних веб-додатків та мобільних додатків, що використовує HTML, CSS і JavaScript. Bootstrap був розроблений у Twitter і став одним з найбільш використовуваних фреймворків для розробки відгуків на веб-сайтах та інтерфейсів користувачів. Серед переваг цього фреймворку доречним є порівняно легка можливість вдосконалення коду для підтримки веб інтерфейсів для мобільних девайсів [5].

MySQL – це реляційна база даних (RDBMS), що використовується для зберігання та управління структурованими даними. Вона є однією з найпопулярніших відкритих систем управління базами даних і використовується в широкому спектрі додатків, від невеликих веб-сайтів до великих корпоративних систем.

Основні характеристики MySQL:

- реляційна структура. MySQL зберігає дані у вигляді таблиць зі стовпцями і рядками. Вона підтримує відносини між таблицями, що дозволяє ефективно управляти даними і забезпечує цілісність даних;
- мова запитів. MySQL використовує мову запитів SQL (Structured Query Language) для взаємодії з базою даних. SQL надає потужні можливості для створення, читання, оновлення та видалення даних;
- підтримка індексів. MySQL підтримує індекси, які допомагають прискорити пошук та фільтрацію даних. Індекси дозволяють ефективно виконувати запити і забезпечують швидкий доступ до даних;
- транзакції. MySQL підтримує транзакції, що забезпечують атомарність, консистентність, ізольованість та стійкість (ACID properties) під час операцій з базою даних. Це важливо для забезпечення цілісності даних та уникнення проблем конкуренції при одночасному доступі до даних;
- масштабованість. MySQL дозволяє масштабувати базу даних, починаючи з невеликих проектів і розширюючись до великих розподілених систем. Вона підтримує горизонтальне масштабування (шарування) і вертикальне масштабування (покращення обладнання);

– безпека. MySQL надає можливості для захисту даних, включаючи аутентифікацію користувачів, контроль доступу та шифрування даних. Вона дозволяє налаштовувати рівні доступу для різних користувачів і забезпечує захист від несанкціонованого доступу до даних.

MySQL широко використовується як основна база даних для веб-додатків, систем управління вмістом, електронної комерції, аналітики даних та багатьох інших сфер. Вона має велику спільноту розробників і підтримується на різних платформах, що робить її потужним інструментом для зберігання та управління даними [6].

Для моделювання та створення програмних широко використовуються мова UML (Unified Modeling Language). Вона пропонує метод для створення й документування архітектури та графічного зображення поведінки готового програмного забезпечення. Використовуючи графічні діаграми, UML допомагає розбити систему на складові частини та проілюструвати зв’язки між ними [7].

Діаграма Use Case – це особливий вид діаграми, яка представляє функціонування системи з точки зору зовнішніх учасників та їх взаємодії з системою. Вона демонструє функціонування системи та взаємодію з акторами, якими можуть бути користувачі, інші системи або зовнішні служби. Діаграма висвітлює основні можливості, визначає учасників та їх взаємодію.

Адміністратор та користувач – основні ролі, які мають уникальні характеристики для взаємодії з програмною системою через веб інтерфейс.

Діаграма Use Case, що показує взаємодію адміністратора з системою, наведена нижче (див. рис. 3.1).

Адміністратор має наступні фунціональні характеристики:

– можливість перегладати облікові дані користувачів, за необхідності редугувати ці дані, видалити профіль користувача;
– можливість переглядати жанри, що використовуються для категорізацію відео контенту. Додавати, змінювати, або видаляти жанри;

- можливість переглядати загальну базу контенту, додавати нові запизи у базу вказуючи дані з інформацією про назву, тривалість, короткий опис та присвоювати жанр з переліку наявних;
- можливість перегладати та редагувати дані власного профілю.

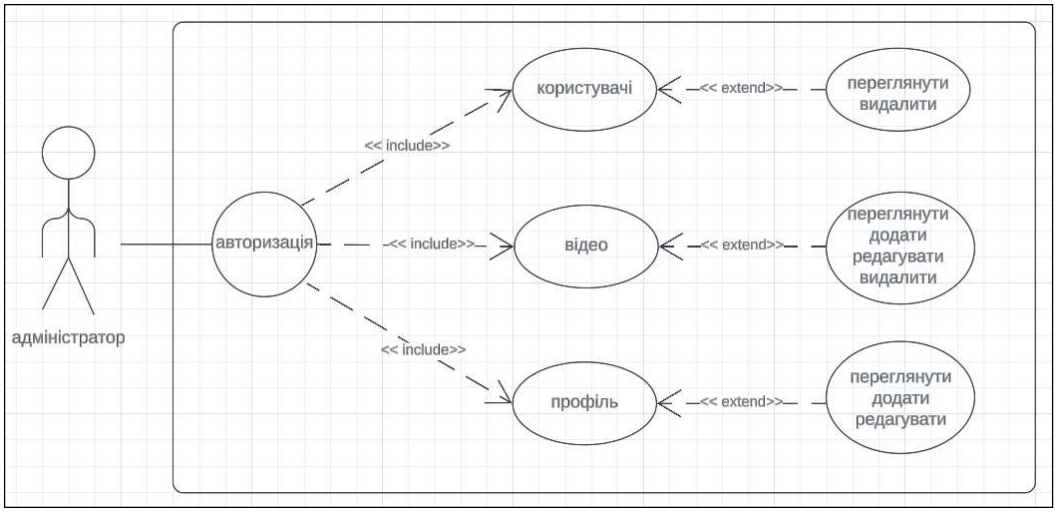


Рисунок 3.1 – Діаграма Use Case для ролі адміністратора (створено самостійно)

Діаграма Use Case, що показує взаємодію користувача з системою, наведена нижче (див. рис. 3.2).

Користувач має наступні фунціональні характеристики:

- можливість створити новий обліковий запис;
- можливість використати існуючий обліковий запис для входу в систему;
- можливість перегладати та редагувати облікові дані власного профілю;
- можливість переглянути статус підписки на сервіс та дату її закінчення;
- можливість обрати нові умови підписки (у разі закінчення) та подовжити підписку;
- можливість пошуку контенту з використанням фільтру за категоріями та\або ключового пошуку;
- можливість перегляду деталей обраного контенту та відтворення відео;
- можливість оцінити обраний контент за власним бажанням та побачити загальний рейтинг з урахуванням оцінок інших користувачів.

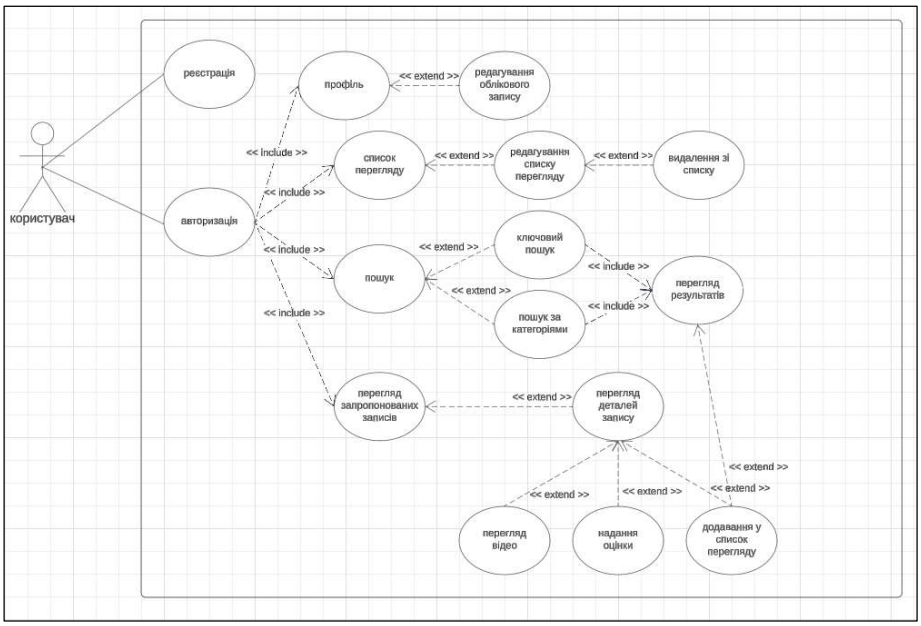


Рисунок 3.2 – Діаграма Use Case для ролі користувача(створено самостійно)

На рисунку 3.3 показано діаграму станів для ролі користувача.

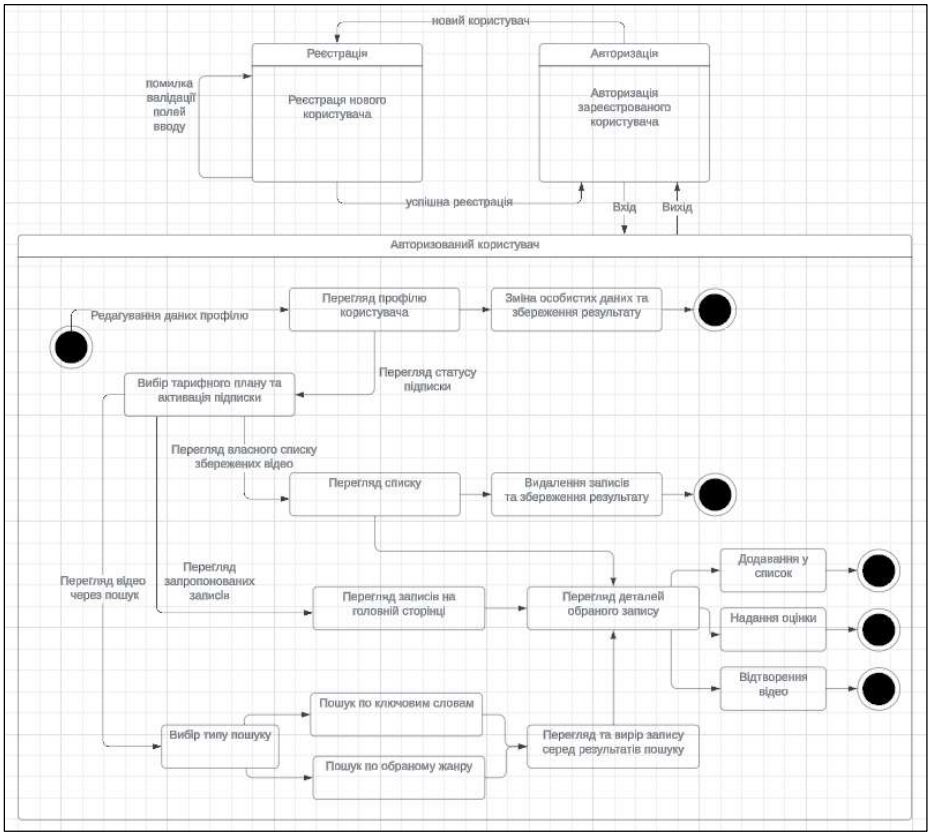


Рисунок 3.3 – Діаграма станів програми для ролі користувача(створено самостійно)

Діаграма станів корисна при моделюванні життєвого циклу об'єкта. Від інших діаграм діаграма станів відрізняється тим, що описує процес зміни станів тільки одного примірника певного класу – одного об'єкта, поведінка якого характеризується його реакцією на зовнішні події. Вона використовується для опису та аналізу різних можливих станів об'єкта в системі, того, як цей об'єкт переходить з одного стану в інший, і що може статися з об'єктом, коли він перебуває в кожному з цих станів.

3.2Проєктування архітектури програмного забезпечення

У Django функціональна структура проекту складається з різних компонентів. Високорівневу діаграму розгортання проекту можна побачити на рисунку 3.4.

Розглянемо основні компоненти діаграми розгортання нашого Django проекту.

Browser. Компонент клієнтської частини, користувач використовує веб браузер для надсилання запитів до веб серверу, отримання\дешифрування та перегляду відповіді.

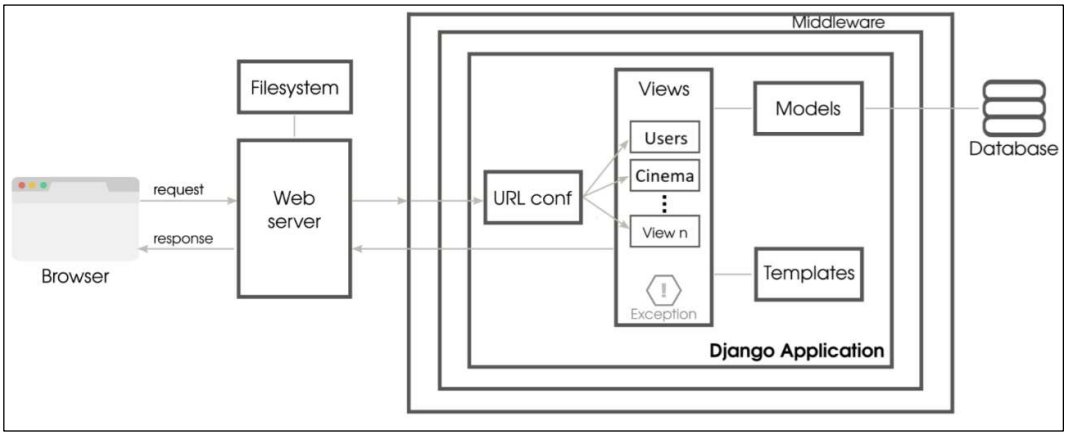


Рисунок 3.4 – Діаграма розгортання для ПС онлайн кінотеатру(створено самостійно)

Web server. Приймає HTTP запити від клієнтів (наприклад, веб браузерів), перенаправляє їх до Django застосунку, а потім повертає відповіді назад до клієнтів.

Filesystem. Безпосередньо середовище де зберігаються файли проекту та його компонентів.

Middleware. Системний компонент. За замовчуванням керує сесіями для користувачів, додає деякі заголовки безпеки, контролює переадресацію та забезпечує захист від атак виду CSRF.

URL conf. Головна роль компоненту – маршрутизація запитів та організація URL структури. URL конфігурація в Django це набір шаблонів, що співставляються з URL запитом, отриманим від веб сервера. Кожен шаблон URL асоціюється з певною функцією представлення (view function), яка викликається, коли URL запиту відповідає шаблон.

Views у Django відповідають за обробку запитів і формування відповідей. Вони визначають, що саме потрібно зробити з вхідним запитом, який отриманий через URL conf, і які дані потрібно відобразити користувачу. Для отримання відповіді на запити, цей компонент використовує моделі для доступу у базу даних.

Models визначають структуру бази даних, представляючи таблиці як класи Python, а стовпці таблиць як атрибути класу. Зазвичай моделі знаходяться в файлі `models.py` кожного додатка.

Templates**.** Шаблонізатор, для поєднання логіки серверної частини та розмітки, для відображення у клієнтській частині. Темплейти Django - це файли HTML, які допомагають у створенні сторінок веб-додатка. Вони дозволяють вставляти динамічний контент із змінних контексту переданих від представлень. Зазвичай темплейти зберігаються в папці `templates/` кожного додатка.

Static Files. Статичні файли – це файли, які не міняються під час роботи додатка, такі як CSS-стилі, JavaScript-сценарії, зображення та інші ресурси. Зазвичай статичні файли зберігаються в папці `static/` кожного додатка.

Проект планується логічно розділити на наступні складові, що у обраному фреймворку використовуються та мають назву «додатків»:

- users. Відповідає за реєстрацію, авторизацію, збереження даних користувачів, перегляд та редагування облікових даних у профілі, а також за підписку та вибір тарифного плану, відстеження його стану;
- mylist. Відповідає за збереження та редагування індивідуального списку для перегляду кожного користувача;
- rating. Обробляє та зберігає індивідуальні оцінку кожного користувача відносно кожного відео з загальної бібліотеки;
- videos. Відповідає за збереження мета даних та посилань на відео контент, його відображення та класифікацію. Поєднується та взамодіє з іншими компонентами системи.

3.3Проектування структури зберігання даних

Перша ітерація продукту передбачає використання бази MySQL в межах віртуальної машини, де буде розгорнуто веб сервер. Друга ітерація передбачатиме перехід у хмарний сервіс AWS, та заміну бази на AWS RDS Aurora, що забезпечить подальше масштабування.

Django використовує бібліотеку Python під назвою MySQLclient для підключення до баз даних MySQL. Ця бібліотека забезпечує інтерфейс API бази даних Python (DB-API) до MySQL, який Django потім використовує для взаємодії з базою даних.

При налаштуванні підключення у Django, окрім інших параметрів, таких як хост, назва бази, користувач, пароль, порт, буде використано параметр "django.db.backends.mysql", що вказує на відвідний двигун для підключення до бази даних MySQL.

Для проєктування структури зберігання даних було використано MySQL Workbench 8.0. На рисунку 3.5 зображено ER діаграму таблиць бази даних для програмної системи онлайн кінотеатру. Слід зазначити наступне:

- Django додає деякі системні таблиці, тож на рисунку зображено тільки таблиці, що було безпосередньо створено нами для програмної системи;
- назви таблиць генеруються Django у форматі «додаток_назва_моделі».

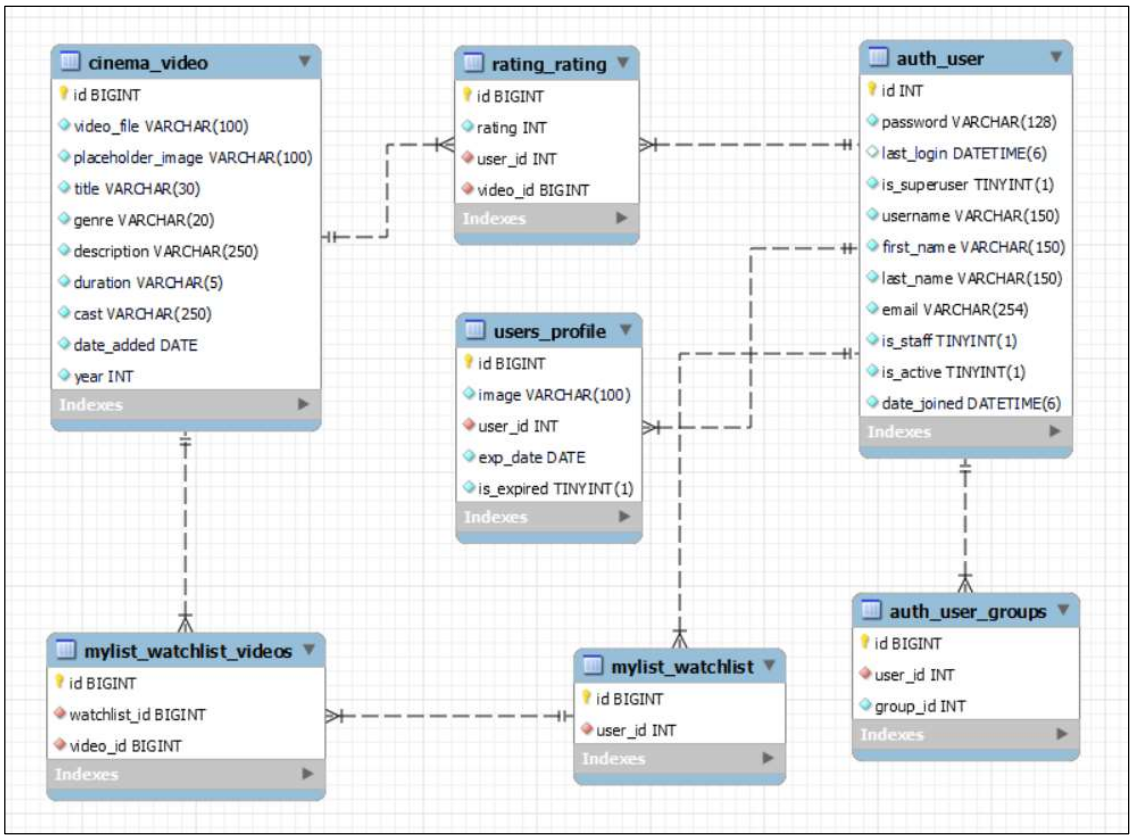


Рисунок 3.5 – ER діаграма таблиць бази даних для ПС онлайн кінотеатру (створено самостійно)

3.4Створення UX дизайну системи

UX-дизайн відповідає за функції, адаптивность продукту і те, які емоції він викликає у користувачів. Чим зрозуміліший інтерфейс, тим легше користувачеві отримати результат і зробити цільову дію.

Основна мета UX дизайну – поліпшити задоволеність та задоволення користувачів шляхом покращення доступності, зручності використання та задоволення взаємодією з продуктом.

Для проекту було обрано використання лаконічного, але зрозумілого та функціонального стилю й інтуїтивно зрозуміле росташування основних елементів керування. З метою покращення візуального представлення у єдиному стилі з нейтральними тонами, планується використати Bootstrap в якості CSS фреймворку.

Приклади UX дизайну деяких сторінок викладено нижче. Головна сторінка перегляду загального переліку відео з елементами пошуку та фільтрації показана нижче (див. рис. 3.6).

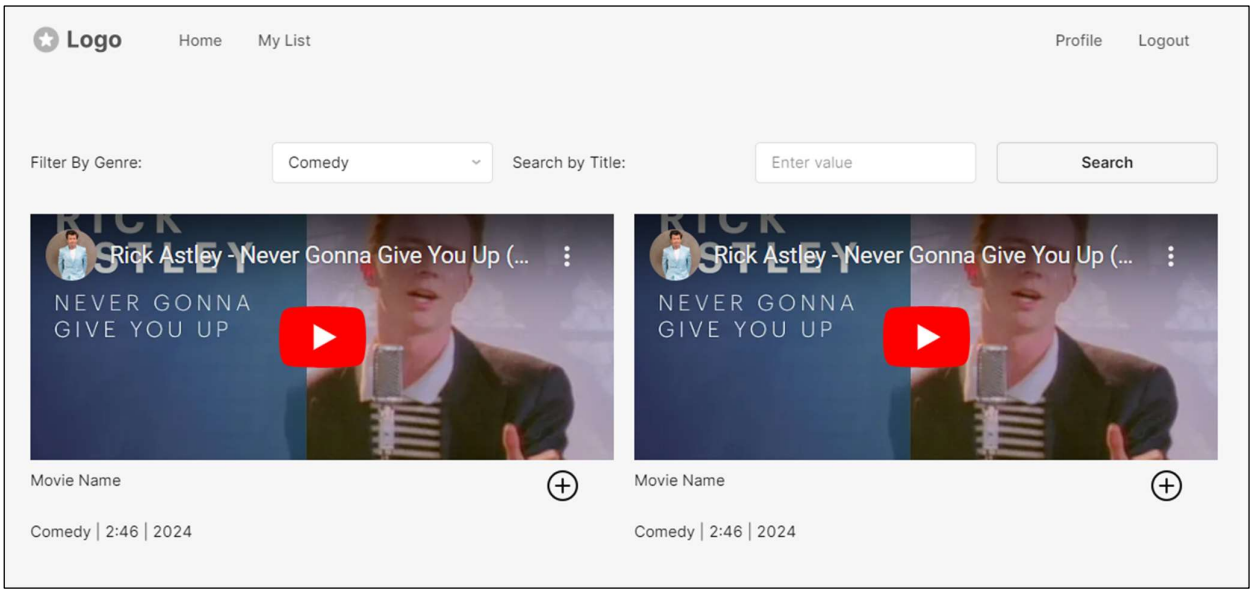


Рисунок 3.6 – UX дизайн домашньої (Home) сторінки (створено самостійно)

На головній сторінці ми одразу бачимо загальний каталог відео, наявних у системі.

Навігація складається з наступинх елементів.

Home – домашня (Home) сторінка, на яку ми потрапляємо виконавши авторизацію у системі (за умови наявності активної підписки).

My List – сторінка з обраними відео активного користувача.

Profile – сторінка перегляду стану підписки, а також перегляу та редагування облікового запису.

Logout – для виходу з облікового запису.

Під навігацією, кожний відео запис міститиме базову інформацію, таку як зображення з назвою, дублювання назви під відео, а також назву жанру, тривалість, рік видання.

З правого боку під відео показано візуальний елемент, натискання на нього додасть обране відео до власного списку перегляду.

Над відео розташовано фільтр зі список жанрів та строка пошуку за ключовими словами.

Приклад UX дизайну сторінки деталей відео показано нижче на рисунку 3.7.

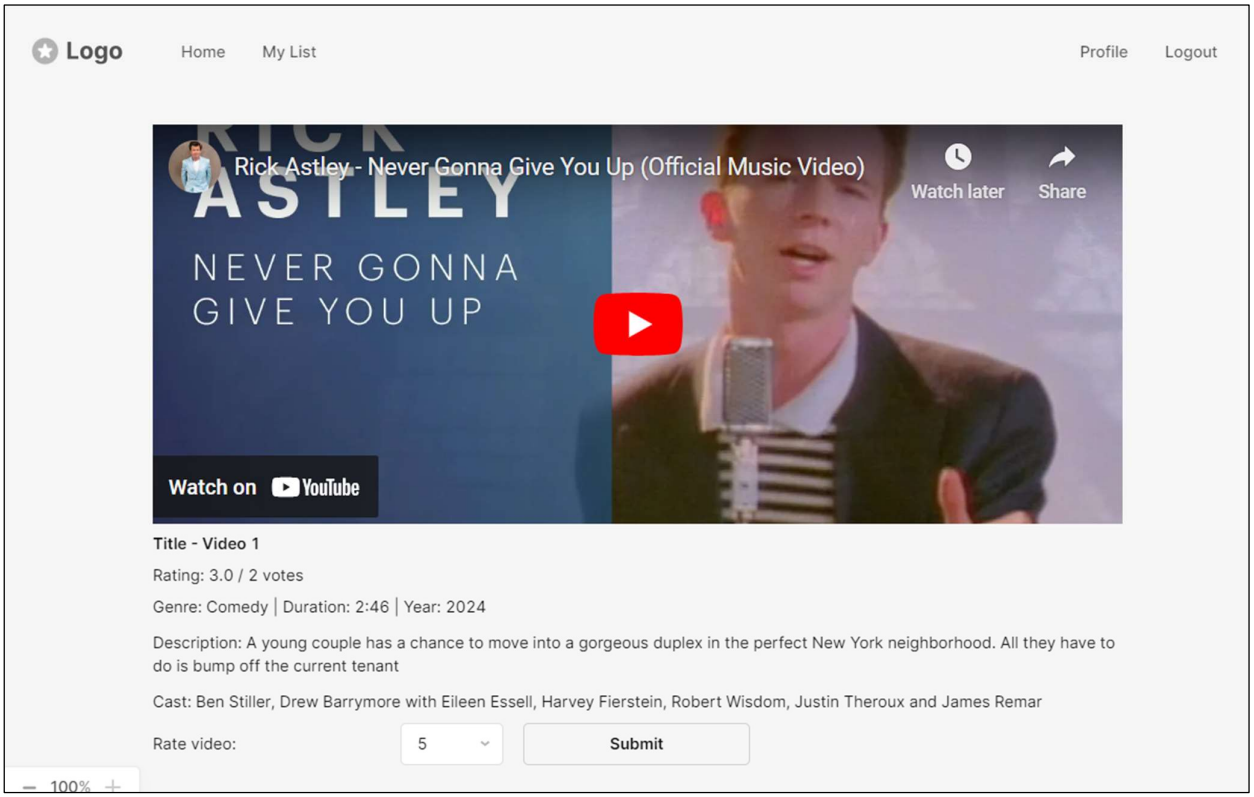


Рисунок 3.7 – UX дизайн детальної сторінки відео (створено самостійно)

Детальна сторінка міститиме додаткову інформацію, а саме рейтинг відео, опис, склад акторів, що прийняли участь у стрічці, функціонал оцінювання відео (що після надання оцінки буде замінено програмою на стрічку з нагадуванням, що оцінку вже зроблено).

3.5Приклади найцікавіших алгоритмів та методів

Розглядемо деякі приклади реалізації цікавих рішень. Першим прикладом є реалізація та поєднання фільтрації та ключового пошуку за назвою. Алгоритм бізнес логіки та кліенської частини представлено в наступному коді.

```
# cinema/views.py
@login_required
def home(request):
```

```
    if request.user.profile.is_expired:
        return redirect('profile')
    selected_genre = request.GET.get('genre', '')
    title = request.GET.get('title', '')
    videos = Video.objects.all()

    if selected_genre:
        videos = videos.filter(genre=selected_genre)

    if title and len(title) >= 3:
        videos = videos.filter(title__icontains=title)

    genres = Video.objects.values_list('genre', flat=True).distinct()
    return render(request, 'cinema/home.html', {'videos': videos, 'genres':
genres, 'selected_genre': selected_genre, 'title': title})

# cinema/templates/cinema/home.html
{% for video in videos %}
<div class="col-md-6">
    <div class="card mb-4">
        <div class="card-img-top">
            <img src="{{ video.placeholder_image.url }}" alt="{{
video.title }}" style="width: 100%; height: 192px; object-fit: cover;">
            </div>
           <div class="card-body">
               <div class="row">
                   <div class="col-md-8">
                       <a href="{% url 'cinema-detail' video.id %}">
                           <h5 class="card-title">{{ video.title }}</h5>
                       </a>
                       <p class="card-text">{{ video.genre }} | {{
video.duration }} | {{ video.year }}</p>
                   </div>
                   <div class="col-md-4 text-right">
                      <a href="#" class="toggle-watch-list" data-video-
id="{{ video.id }}">
                        <i class="fas fa-plus"></i>
                        </a>
                </div>
            </div>
        </div>
    </div>
</div>
{% if forloop.counter|divisibleby:2 and not forloop.last %}
```

Фільтрування за жанром. Рядок selected_genre = request.GET.get('genre', '') отримує вибраний жанр із параметрів запиту. Якщо жанр не було обрано, за замовчуванням передається пусте значення, що буде дорівнювати відображенню всих записів. Якщо вибирається жанр, рядок videos = videos.filter (genre = selected_genre) фільтрує записи (відео) за обраним жанром.

Пошук. Рядок title = request.GET.get('title', '') отримує пошуковий запит із параметрів запиту. Якщо довжина пошукового запиту становить 3, або більше символів, рядок videos = videos.filter (title__icontains = title) фільтрує запит відеороликів, шукаючи співпадіння із назвами відео без урахування малих чи великих літер.

Робота фільтру та пошуку разом. Фільтрування за жанром та пошуком працює разом, звужуючи перелік результатів. Якщо обрано жанр, запит відеороликів фільтрується цим жанром. Потім, вже відфільтрований результат подається для виконання пошуку за ключовими словами.

Як бачимо з прикладу шаблону розмітки, Django вміє працювати з змінними та операторами, в нашому випадку використано цикл для виводу результатів пошуку.

Звернемо також увагу на обмеження до 2 елементів у рядку {% if forloop.counter|divisibleby:2 and not forloop.last %} у нашому шаблоні виконує перехід на новий рядок кожні 2 відео.

Ще одне рішення, яке ми розглянемо, це реалізація оцінювання та виводу рейтингу кожного відео. Код реалізації:

```
# cinema/views.py
def detail(request, video_id):
    if request.user.profile.is_expired:
        return redirect('profile')
    video = get_object_or_404(Video, pk=video_id)
    if request.method == 'POST':
        rating = request.POST.get('rating')
        Rating.objects.update_or_create(user=request.user, video=video,
defaults={'rating': rating})
    user_has_voted = Rating.objects.filter(user=request.user,
video=video).exists()
    average_rating =
Rating.objects.filter(video=video).aggregate(Avg('rating'))['rating__avg']
    votes = Rating.objects.filter(video=video).count()
    return render(request, 'cinema/detail.html', {'video': video,
'average_rating': average_rating, 'votes': votes, 'user_has_voted':
user_has_voted})

# cinema/templates/cinema/detail.html
<div class="col-md-12">
    <video width="100%" controls autoplay controlsList="nodownload"
oncontextmenu="return false;">
        <source src="{{ video.video_file.url }}" type="video/mp4">
```

```
    </video>
    <h2>{{ video.title }}</h2>
    <p><i>Rating: </i> <b>{{ average_rating|default:"No ratings yet" }} /
{{ votes }} votes</b></p>
   <p><i>Genre:</i> {{ video.genre }} | <i>Duration:</i> {{ video.duration
}} | <i>Year:</i> {{ video.year }}</p>
   <p><i>Description:</i> {{ video.description }}</p>
   <p><i>Cast</i>: {{ video.cast }}</p>
   {% if not user_has_voted %}
   <form method="post">
       {% csrf_token %}
       <label for="rating">Rate video:</label>
       <select name="rating" id="rating">
           <option value="1">1</option>
           <option value="2">2</option>
           <option value="3">3</option>
           <option value="4">4</option>
           <option value="5">5</option>
     </select>
     <button type="submit">Submit</button>
   </form>
   {% else %}
   <p>Your rating has been submitted. Thank you!</p>
   {% endif %}
</div>
```

Дане рішення кикористовує дані з таблиці rating, та виводить форму для оцінювання лище до моменту, коли користувач надасть власну оцінку. Надалі форма заміщується тестовим повідомленням. Це реалізовано через перевірку наявності оцінки у відвовідній комірці бази даних для активного користувача user_has_voted = Rating.objects.filter(user=request.user, video=video).exists() і надалі перевіркою цієї змінної у шаблоні {% if not user_has_voted %}. Вивід загальної оцінки та кількості оцінок виконується безпосередньо при завантаженні сторінки, а підрахунок виконується – рядками average_rating = Rating.objects.filter(video=video).aggregate(Avg('rating'))['rating__avg'] та votes = Rating.objects.filter(video=video).count(). У подальшому планується вдосконалити цей код з метою оптимізації затрачених ресурсів. А саме підраховувати загальну середню оцінку та оновлювати кількість оцінок у окремомі полі таблиці для кожного відео. Таким чином при завантаженні сторінки перегляду відео не буде виконуватись підрахунок, а лише считування, що зменшить навантаження на ресурси сервера. Це особливо актуально з активним зростом числа користувачів програмної системи.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1Програмне рішення для створення серверної частини

Для створення сервеної частини було обрано Django фреймворк, як потужне рішення з широким переліком додаткових бібліотек, підтримкою обраної бази даних, та можливостями реалізації будь яких алгоритмів та налаштувань.

Загальна архітектура фреймворку Django включає створення проекту та додатків, що є логічними блоками – складовими проекту.

Після створення проекту фреймворк створює основну директорію з назвою проекту, та піддиректорію з такою з назвою. Вона виконує декулька функцій, наприклад містить файл url.py, що служить загальним роутером для правильного посилання на відповідні додатки, settings.py, де ми налаштовуємо підключення до бази MySQL, реєструємо наші додатки, налаштовуємо домен сайту, директорію для зберігання медіа файлів за замовчуванням, тощо.

Приклад налаштування підключення до бази даних у файлі settings.py показано в коді нижче.

```
DATABASES = {
    "default": {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'onlinecinema',
        'USER': os.environ.get('DB_USER),
        'PASSWORD': os.environ.get('DB_PASSWORD'),
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
```

Слід зазначити, що з метою обмежити виток чутливих даних, таких як ім’я користувача та пароль доступу до бази, було використано змінні на рівні операційної системи, тож замість статичних даних рекомендується використовувати посилання на ці змінні через os.environ.get().

Проаналізувавши вимоги до проекту, його було розділено на складові та реалізовано за допомогою наступних додатків: users, cinema, mylist, rating.

Розглянемо реалізацію кожного додатку.

Додаток users необхідний для забеспечення одного з основних функціональних можливостей та фундаментальних основ програмної системи. Розглянемо їх окремо.

Реєстрація користувачів. Фунціонал надає форму реєстрації через UserRegisterform та обробляє нові запити, зберігаючи дані у обліковому записі та відповідній таблиці баз даних.

Приклад коду UserRegisterform та обробки помилок показано нижче.

```
# users/forms.py
from django.contrib.auth.forms import UserCreationForm
from .models import Profile

class UserRegisterForm(UserCreationForm):
    email = forms.EmailField()

    class Meta:
        model = User
        fields = ['username', 'email', 'password1', 'password2']

#users/views.py
def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            messages.success(request, f'Account for {username} has been
created! Now you can login.')
            return redirect('login')
    else:
        form = UserRegisterForm()
    return render(request, 'users/register.html', {'form': form})
```

Сторінка реєстрації нового користувача зображена на рисунку 4.1.

Управління профілем користувача. Сторінка профілю дозволяє користувачам переглядати та оновлювати облікову інформацію. В нашому випадку це поля username та email. Для редагування та оновлення цих даних використовуються вбудовані методи UserUpdateForm та ProfileUpdateForm.

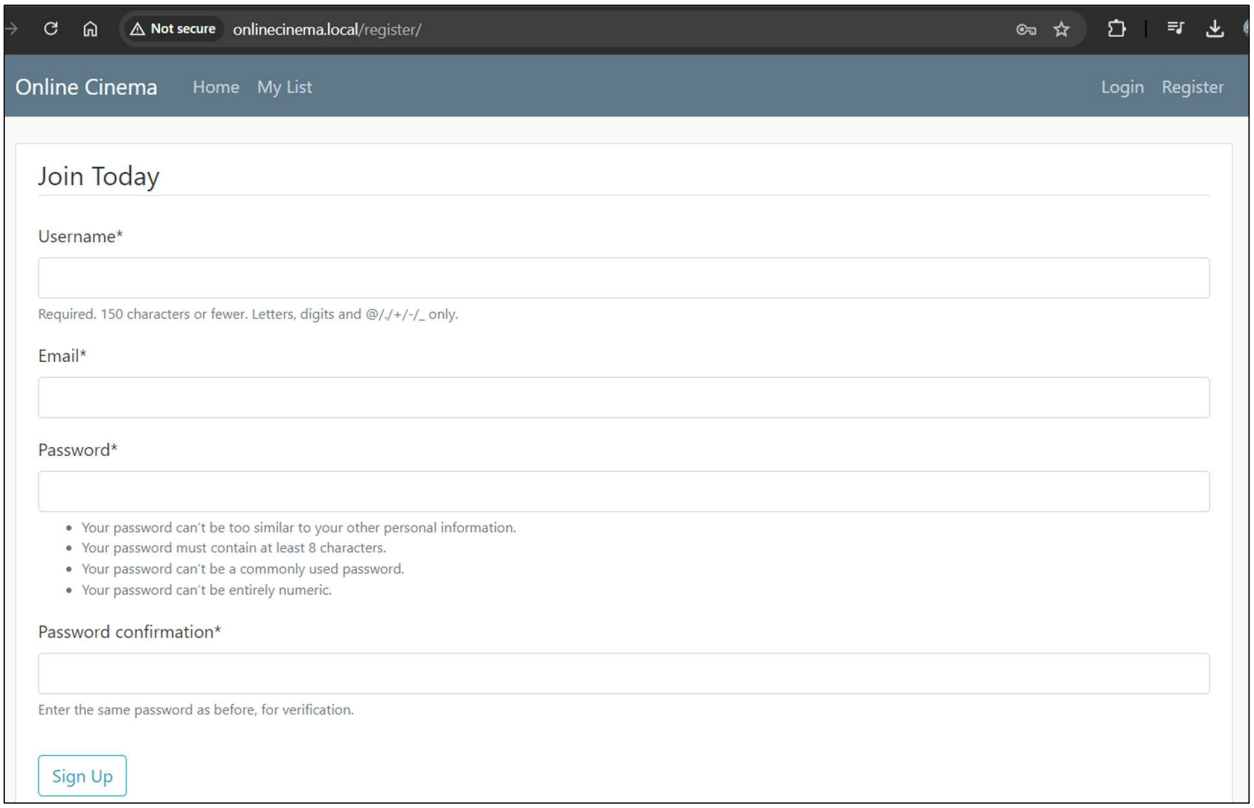


Рисунок 4.1 – Сторінка форми реєстрації нового користувача (створено самостійно)

Профіль також має фунціонал перегляду, та у разі необхідності поновлення підписки користувача.

```
# users/views.py
def profile(request):
    if request.method == 'POST':
        user_update_form    = UserUpdateForm(request.POST,
instance=request.user)
        profile_update_form = ProfileUpdateForm(request.POST,
request.FILES, instance=request.user.profile)
        if user_update_form.is_valid() and profile_update_form.is_valid():
            user_update_form.save()
            profile_update_form.save()
            messages.success(request, f'Profile details have been
updated.')
            return redirect('profile')
    else:
        user_update_form    = UserUpdateForm(instance=request.user)
        profile_update_form =
ProfileUpdateForm(instance=request.user.profile)

        exp_date = request.user.profile.exp_date
        is_expired = exp_date < date.today()
```

```
    context = {
        'u_form': user_update_form,
        'p_form': profile_update_form,
        'is_expired': is_expired,
        'exp_date': exp_date
    }
    return render(request, 'users/profile.html', context)
```

Вихід з облікового запису користувача. Ця функція використовує вбудований метод Django.

```
def user_logout(request):
    logout(request)
    return render(request, 'users/logout.html')
```

Сторінка управління профілем зображена на рисунку 4.2.

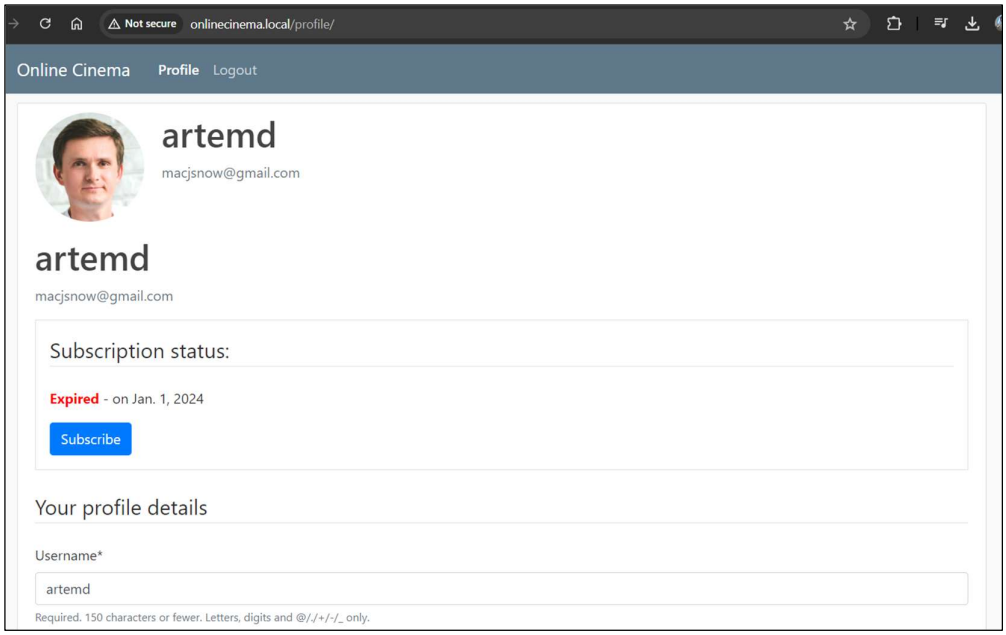


Рисунок 4.2 – Сторінка управління профілем (створено самостійно)

Управління підпискою. Перегляд статусу підписки, та в разі необхідності вибору нової та оновлення. Наявна реалізація дозволяє переглянути статус, дату завершення підписки. Логіка налаштована таким чином, що кнопка поновлення підписки стає активною тільки тільки якщо підписка закінчилась. Додатково є перевірка та обмеження прямого доступу до сторінки оновлення.

```
№ users/views.py
def subscription(request):
    if request.user.profile.exp_date > date.today():
        return redirect('profile')
    return render(request, 'users/subscription.html')

def process_subscription(request):
    plan = request.POST['plan']
    user_profile = request.user.profile
    user_profile.exp_date = date.today() + timedelta(days=int(plan))
    user_profile.save()
    return redirect('cinema-home')

def update_profile(sender, user, request, **kwargs):
    profile = user.profile
    profile.is_expired = profile.exp_date < date.today()
    profile.save()

# users/templates/users/profile.html
<div class="border p-3 mb-4">
    <legend class="border-bottom mb-4">Subscription status:</legend>
    {% if is_expired %}
    <p><span style="color: red;"><b>Expired</b></span> - on {{ exp_date
}}</p>
    <a href="{% url 'subscription' %}" class="btn btn-
primary">Subscribe</a>
    {% else %}
    <p><span style="color: green;"><b>Active</b></span> - expires on {{
exp_date }}</p>
    {% endif %}
</div>
```

Сторінка вибору тарифного плану зображена на рисунку 4.3.

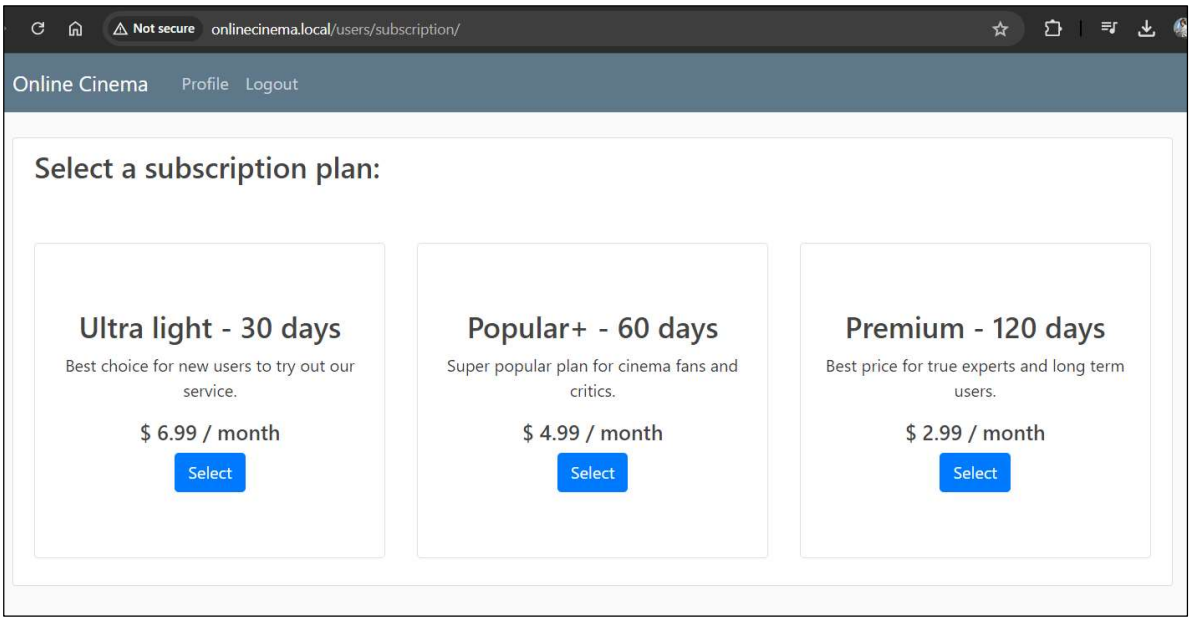


Рисунок 4.3 – Сторінка вибору тарифного плану (створено самостійно)

Перевірка терміну дії підписки. Функція update_profile це сигнал, який викликається, коли користувач входить. Він перевіряє, чи закінчується підписка користувача, і оновлює поле профілю користувача.

Приклад виводу інформації про активну підписку, що закінчилась, зображено на рисунку 4.4.

Рисунок 4.4 – Вивід інформації про активну підписку (створено самостійно)

Приклад виводу інформації про підписку, що закінчилась, зображено на рисунку 4.5.

Рисунок 4.5 – Вивід інформації про підписку, що закінчилась (створено самостійно)

Модель профілю розширює вбудовану модель користувача Django з додатковими полями: зображення, дата закінчення підписки та булеве поле.

```
# users/models.py
class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    image = models.ImageField(default='default.jpg',
upload_to='profile_pics')
    exp_date = models.DateField(default=date(2024, 1, 1))
    is_expired = models.BooleanField(default=True)
```

Інтерфейс адміністратора для перегляду профілей користувачів та управління ними реалізовано через реєстрацію моделі користувача та клас ProfileAdmin.

```
# users/admin.py
class ProfileAdmin(admin.ModelAdmin):
    list_display = ('user', 'exp_date')

admin.site.register(Profile, ProfileAdmin)
```

Панель адміністратора з відкритим розділом редагування профілей користувачів зображено на рисунку 4.6.

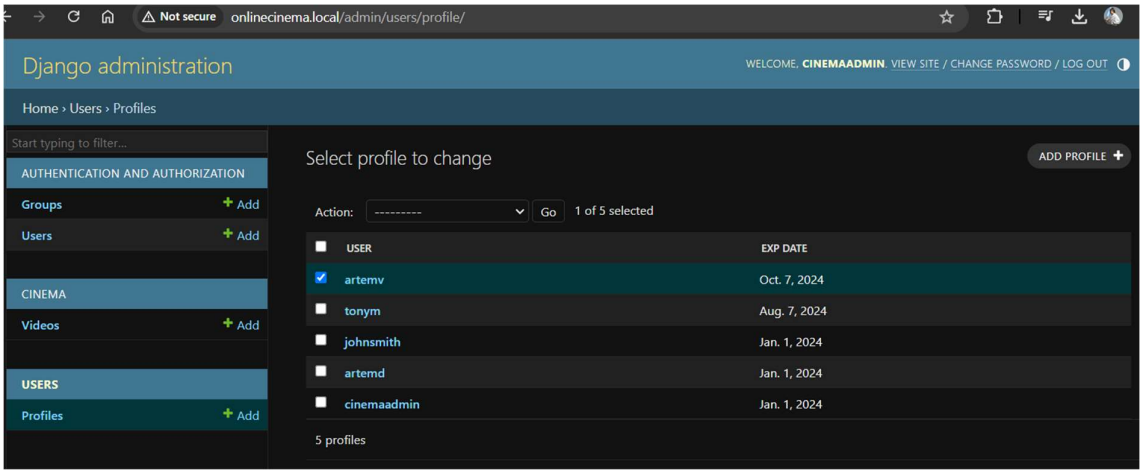


Рисунок 4.6 – Панель адміністратора, редарування профілей користувачів (створено самостійно)

Додаток cinema це ще одна складова основного функціоналу програмної системи. У попередньо розглянутому дотатку users ми забезпечили процес реєстрації та авторизації, поновлення та відстежування підписки користувача. Саме умова активного стану підписки забезпечує доступ до функціоналу перегляду відео контенту сайту та роботи з ним. Приклади серверної та клієнської частини було розібрано у розділі цікавих рішень, та показано на рисунках 3.8 та 3.9. Основна задача цього додатку – показ контенту, забезпечення фільтрації та пошуку.

Додаток mylist надає можливість зберегти обраний контент у власний розділ – список перегляду, що забезпечить швидкий доступ до відео, обраного попередньо.

Для реалізації динамічної складової, а саме заміни малюнку «хрестика» перед додаванням, на «галочку» після додавання у власний список, було додано

JavaScript код у вигляді рішення з використанням бібліотеки jQuery, що значно пришвидшує розробку. Код працює також і після подвійного натискання, повертаючи малюнок «хрестика». У наведеному прикладі коду також реалізовано надсилання обраного id елементу через fetch(url) та надсилання HTTP запиту до серверу з метою подальшого додавання у власний список разом з візуальною трансформацією.

```
# cinema/templates/cinema/home.html
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></sc
ript>
<script>
    var urlTemplate = '{% url "mylist:toggle-watch-list" video_id=0 %}';

    $(document).ready(function() {
    $('.toggle-watch-list').click(function(event) {
        event.preventDefault();
        var link = $(this);
        var videoId = link.data('video-id');
        var url = urlTemplate.replace('0', videoId);
        fetch(url).then(response => response.json()).then(data => {
            var icon = link.find('i');
            if (data.added) {
                icon.removeClass('fa-plus').addClass('fa-check');
            } else {
                icon.removeClass('fa-check').addClass('fa-plus');
            }
        });
    });
});
</script>
```

Для реалізації перегляду та видалення елементів зі сторінки mylist реалізовано функції.

```
# mylist/views.py
def toggle_watch_list(request, video_id):
    video = get_object_or_404(Video, pk=video_id)
    watch_list, created =
WatchList.objects.get_or_create(user=request.user)
    if video in watch_list.videos.all():
        watch_list.videos.remove(video)
        added = False
    else:
        watch_list.videos.add(video)
        added = True
    return JsonResponse({'added': added})
```

```
@login_required
def mylist(request):
    user = request.user
    try:
        watch_list = WatchList.objects.get(user=user)
        videos = watch_list.videos.all()
    except WatchList.DoesNotExist:
        videos = []
    return render(request, 'mylist/mylist.html', {'watch_list': videos})
```

Приклад елементу що додано до власного списку, та елементу за замовчуванням, тобто такого, що не додано до власного списку зображено на рисунку 4.7.

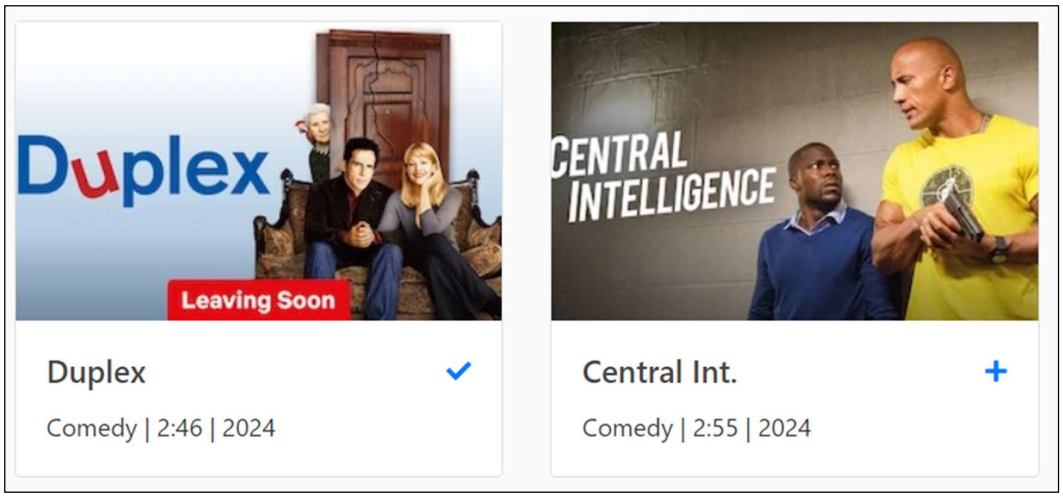


Рисунок 4.7 – Елемент, що додано до власного списку (зліва), та елемент за замовчуванням (справа) (створено самостійно)

Додаток rating відображає загальний рейтинг кожного окремого відео, загальну кількість голосів та форму на сторінці деталей відео, а також зберігає ці дані у таблиці бази даних.

4.2Програмне рішення для створення клієнтської частини

Код для клієнської частини реалізовано в рамках Django за допомогою шаблонів з динамічними елементами фреймворку. Шаблони дозволяють перевикористовувати блоки розмітки, що повторюються, вставляти елементи у розмітку безпоседедньо з бази, або динамічні елементи використовуючи спеціальні блоки.

Приклад перевикористання меню навігації для сторінок показано в коді нижче.

```
# cinema/templates/home.html
{% extends "cinema/base.html" %}

{% block content %}

<div class="container">
  <form method="get" class="row mb-3">
…
```

Приклад використання динамічних елементів шаблону та їх вставки показано в наступному коді.

```
# cinema/templates/home.html
{% for video in videos %}
<div class="col-md-6">
    <div class="card mb-4">
        <div class="card-img-top">
            <img src="{{ video.placeholder_image.url }}" alt="{{
video.title }}" style="width: 100%; height: 192px; object-fit: cover;">
            </div>
            <div class="card-body">
                <div class="row">
                    <div class="col-md-8">
                        <a href="{% url 'cinema-detail' video.id %}">
                            <h5 class="card-title">{{ video.title }}</h5>
                        </a>
                        <p class="card-text">{{ video.genre }} | {{
video.duration }} | {{ video.year }}</p>
                    </div>
                    <div class="col-md-4 text-right">
                            <a href="#" class="toggle-watch-list" data-
video-id="{{ video.id }}">
                    <i class="fas fa-plus"></i>
                     </a>
                    </div>
             </div>
        </div>
    </div>
</div>
{% if forloop.counter|divisibleby:2 and not forloop.last %}
```

Як бачимо, блок {% for video in videos %} відповідає за обробку та вставку елементів обьекту за допомогою циклу.

Блок із структурою {{ video.title }} відповідає за вставку елементу обьекта, в тому числі в межах циклу.

Можливості динамічних блоків навіть посилені додатковими умовами, як наприклад {% if forloop.counter|divisibleby:2 and not forloop.last %}. В даному випадку ми хочемо розбивати стрічку так, щоб в одному рядку було не більше 2 елементів з циклу.

Для надання системі можливості динамічної взаємодії розмітки, було застосовано деякі можливості мови JavaScript.

Приклад виводу динамічних елементів зображено на рисунку 4.7 вище.

Ми широко використали можлівості CSS фреймворку Bootstrap 5 для надання кліенській частині однорідного та охайного вигляду. Також показано використання спеціальних класів, що відповідають за розмітку сторінки, наприклад class="row" забеспечує групування елементів та їх вивід у одній стрічці, а класс class="col-md-8" дозволяє елементу займати до 8 умовних блоків ширини сторінки з 12 для середнього розміру екрану. Класс class="card-img-top" застосовує до елемента набір CSS правил для форматування (в нашому випадку картинки). Загальний вигдял сторінки з використанням фреймворку для розмітки зображено на рисунку 4.8.

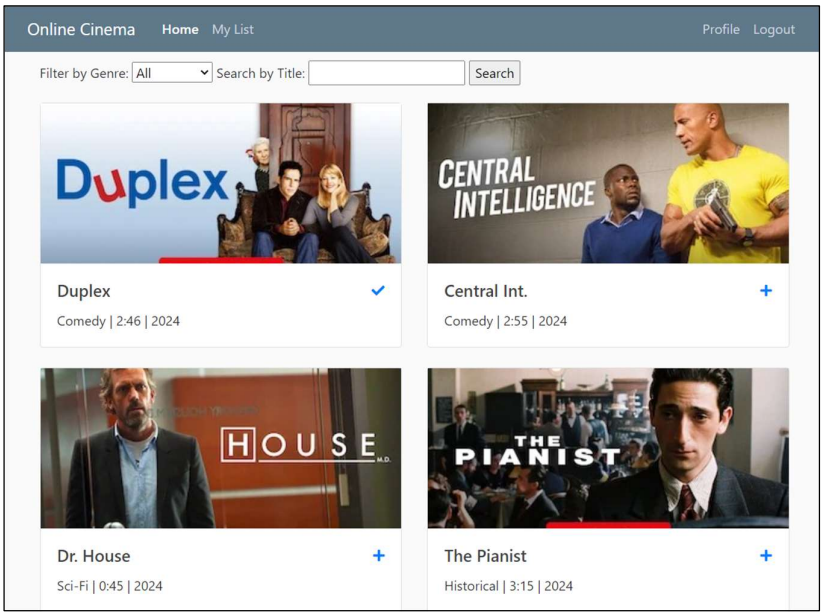


Рисунок 4.8 – Розмітка за допомогою CSS фреймворку Bootstrap (створено самостійно)

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1Тестування серверної частини за допомогою Postman

Postman – це професійний інструмент, що використовується для тестування API. Він дозволяє розробникам відправляти HTTP-запити до сервера та перевіряти відповіді. Цей інструмент підтримує всі типи HTTP-запитів, включаючи GET, POST, PUT, DELETE тощо інші. Зручність його використання полягаю у функціональності, адже він вміє перевіряти відповідей від сервера, автоматизувати тестування, використовувати змінні та середовища, імпортувати та експортувати тести.

Використовуючи Postman, виконаємо запит з метою отримати відповіді враховуючи параметри пошуку та фільтрації.

Оскільки наш проект має систему аутентифікації, спершу використаємо браузер для авторизації у системі, далі через консоль розробника скопіюємо параметри ключ-значення для cookies, а саме crsftoken, sessionid. Далі відкриємо Postman та створимо workspace. У параметри cookies задамо наш домен (onlinecinema.local) та окремо кожен з параметрів ключ-значення cookies.

Тепер виконаємо запит для перевірки роботи API.

Запит має відповісти з HTML сторінки, що містить відео елементи з урахуванням жанру – Historical. Виконаємо запит та перевіримо результати. Як бачимо на рисунку 5.1, код відвовіді 200, тобто запит виконано успішно, а повернений HTML код містить 1 блок з відео.

Далі перевіримо правильність відповіді за допомогою веб браузеру й відвовідних налаштувань фільтрації на сторінці, як бачимо результат співпадає (див. рис. 5.2).

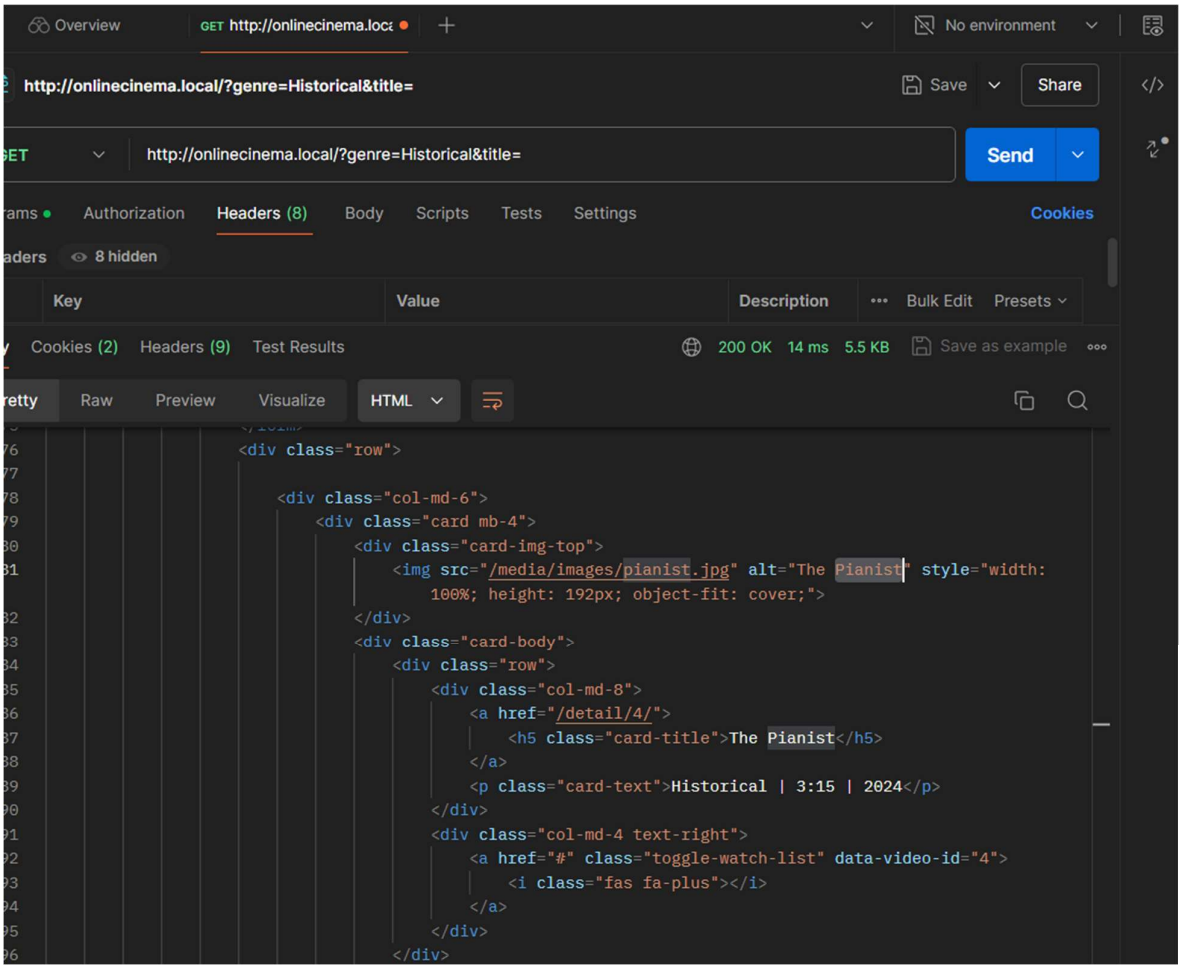


Рисунок 5.1 – Відповідь запиту з фільтром Historical у Postman (створено самостійно)

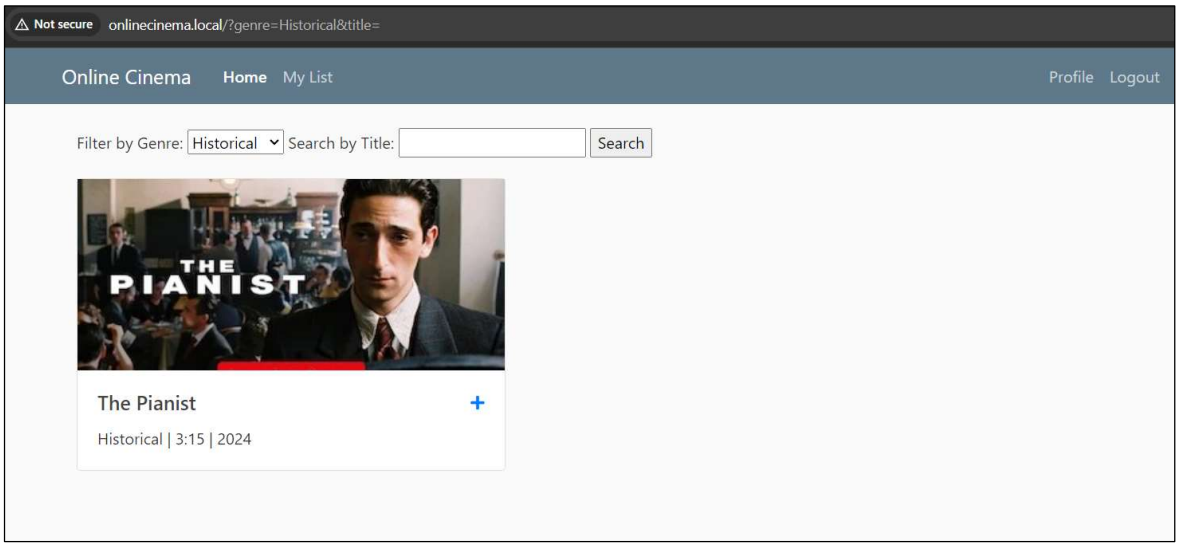


Рисунок 5.2 – Відповідь запиту з фільтром Historical у Chrome (перевірка) (створено самостійно)

5.2Manual та нефункціональне тестування клієнтської частини

В рамках мануального тестування було обрано процес додавання обраних відео до списку перегляду активного користувача і далі видалення з цього списку.

Нефункціональне тестування включатиме перевірку правильності відображення сторінки з відео після додавання та після видалення.

Для реалізації перевірки виконаємо авторизацію у системі, та перейдемо до сторінки з відео. Далі відкриємо консоль розробника у нашому браузері (Google Chrome). Це потрібно для відстежування можливих програмних помилок під час тестування.

Оберемо декілька відео, та додамо у власний список перегляду, натиснувши «хрестик». В результаті бачимо, що обрані відео помічаються як додані, про що свідчить відповідна «галочка» (рисунок 5.3).

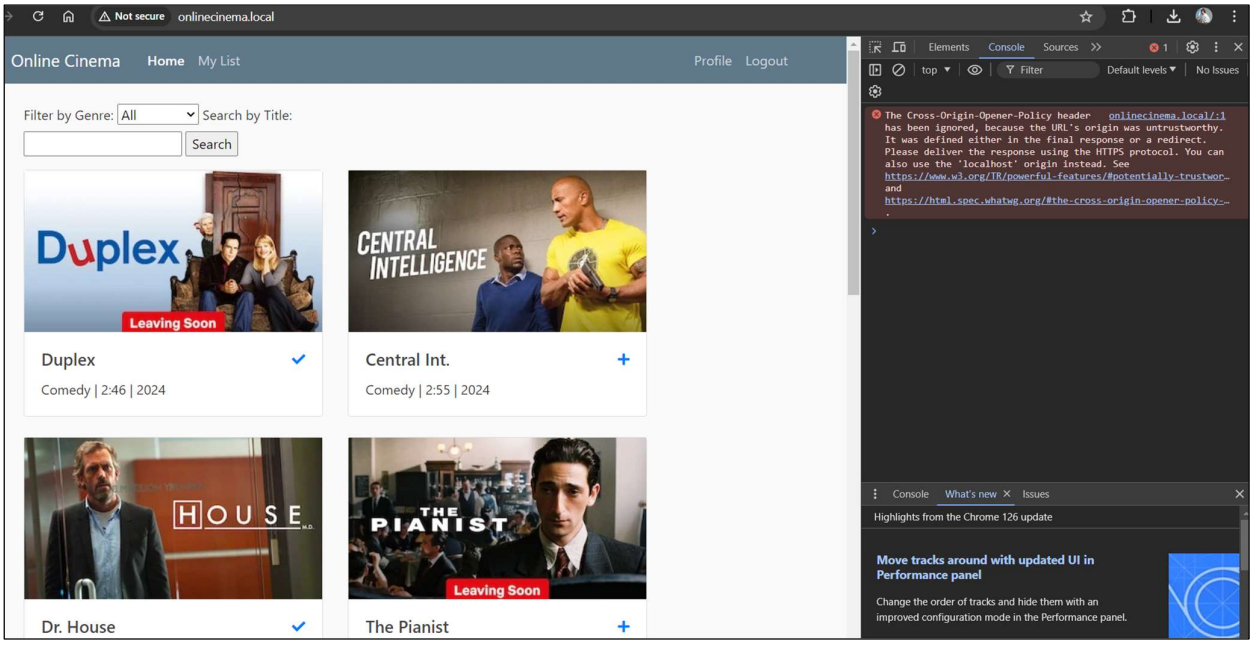


Рисунок 5.3 – Додавання відео до власного списку перегляду (створено самостійно)

Як бачимо, відео додано і жодних помилок у консолі не відображено (окрім попередження про CORS policy).

Тепер перейдемо до сторінки My List та перевіримо наявність доданих елементів. Як бачимо на рисунку 5.4, відео відображаються, жодних помилок у консолі.

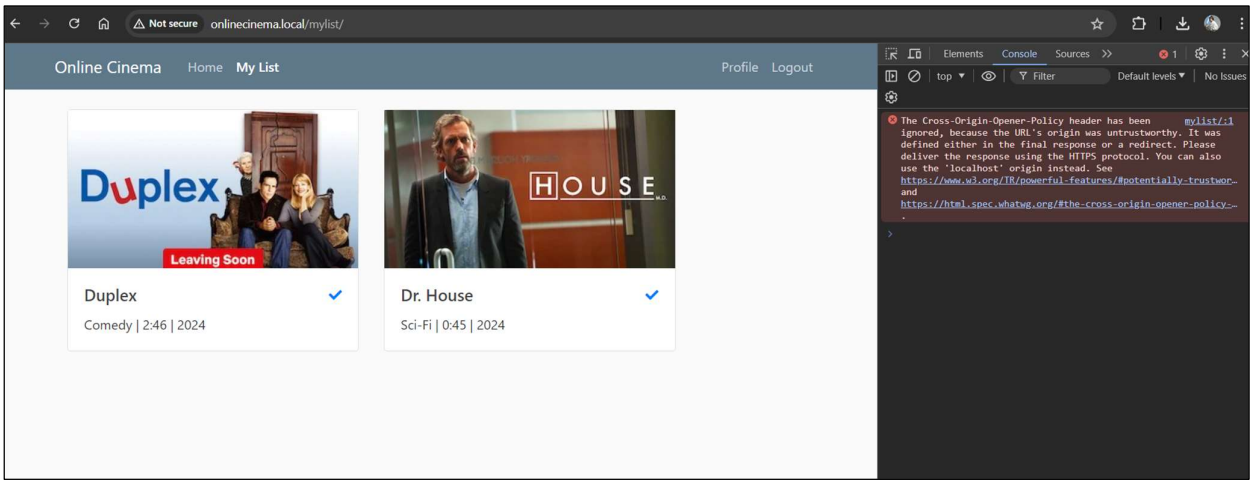


Рисунок 5.4 – Додавання відео до власного списку, сторінка My List (створено самостійно)

Тепер протестуємо видалення відео із власного списку, та перевіримо результат видалення візуально через браузер та через консоль бази даних.

Рисунок 5.5 показує стан таблиці mylist_watchlist_videos бази даних до видалення елементів.

Рисунок 5.5 – Стан таблиці mylist_watchlist_videos до видалення відео із власного списку (створено самостійно)

Як бачимо, наразі з watchlist_id 1 (тобто списком перегляду нашого користувача) асоціюється 2 записи з video_id 1 та 3.

Тепер спробуємо видалити відео через веб інтерфейс, натиснувши відповідну картинку «галочку» на блоках з відео.

На рисунку 5.6 бачимо, що відео зникли зі сторінки. Тепер перевіримо, чи було їх видалено з таблиці бази даних. На рисунку 5.7 ми бачимо результат повторного запиту до бази даних після видалення записів. Записи було видалено, отже тестування можна вважати успішним, помилок при даних умовах не знайдено.

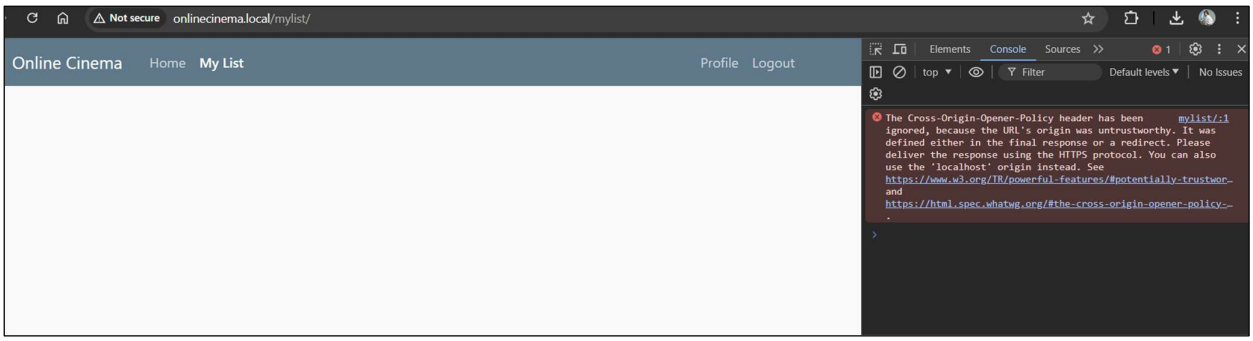


Рисунок 5.6 – Видалення відео з власного списку, сторінка My List (створено самостійно)

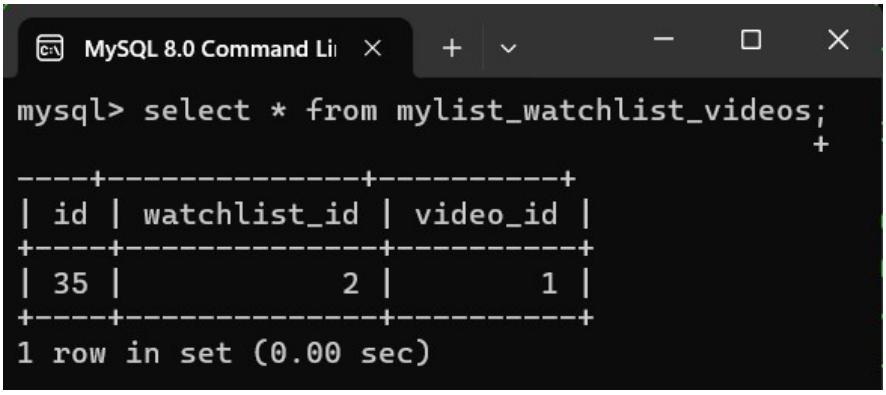


Рисунок 5.7 – Стан таблиці mylist_watchlist_videos після видалення відео із власного списку (створено самостійно)

ВИСНОВКИ

В процесі підготовки кваліфікаційної роботи бакалавра було зроблено аналіз предметної галузі, визначено та порівняно переваги та недоліки конкурентів, підготовлено, поставлено та виконано задачу з розробки програмної системи онлайн кінотеатру.

Було обґрунтовано конкурентні переваги та сформульовано загальне бачення розвитку програмної системи, виділено бачення цільової аудиторії та напрямків масштабування.

Створено архітектуру з демонстрацією та поясненням взаємодії окремих підсистем, зображено та пояснено взаємодію з користувачем, окреслено наявні вимоги до впровадження та технічні обмеження.

Застосувавши сучасні підходи та програмні засоби було створено захищену, але легку у використанні програмну систему, виділено наступні кроки з вдосконалення та запровадження у хмарній екосистемі.

Використовуючи результат програмної реалізації користувач має змогу створити обліковий запис пройшовши реєстрацію, авторизуватись у системі за допомогою логін форми, відредагувати власний профіль, обрати підписку за однією із 3 опцій, переглядати, фільтрувати відео з загального каталогу, а також використовувати швидкий пошук за ключовими словами. За допомогою функції збереження до власного списку, користувач має змогу відкласти перегляд, створивши список улюбленого контенту, а система рейтингу дає можливість поділитись враженнями та бачити загальну оцінку аудиторії.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Fatimah Muhd Shukri, Hidayat Hamid. The Impact of Online Streaming Platforms on the Film Industry and Film-Viewing Culture in Malaysia: Journal of Entrepreneurship and Business 12(1): 101-115, March 2024. - 115 с. URL: https://www.researchgate.net/publication/379444803_The_Impact_of_Online_Streaming_Platforms_on_the_Film_Industry_and_Film-Viewing_Culture_in_Malaysia (дата звернення: 20.06.2024).

2. Video Streaming Market (By Streaming Type: Live Video Streaming, Non-Linear Video Streaming; By Component (USD): Software, Content Delivery Services; By Solutions: Internet Protocol TV, Over-the-Top, Cable TV, Pay-TV; By Platform: Gaming Consoles, Laptops & Desktops, Smartphones & Tablets, Smart TV; By Service: Consulting, Managed Services, Training & Support; By Revenue Model; By Deployment Type; By User) - Global Industry Analysis, Size, Share, Growth, Trends, Regional Outlook, and Forecast 2022-2030: Report Code: 2061. URL: https://www.precedenceresearch.com/video-streaming-market (дата звернення: 20.06.2024).

3. The Django admin site. Django documentation. URL: https://docs.djangoproject.com/en/3.0/ref/contrib/admin/ (дата звернення 20.06.2024).

4. Bill Lubanovic. Introducing Python. Oreilly, 2016. 480 с.

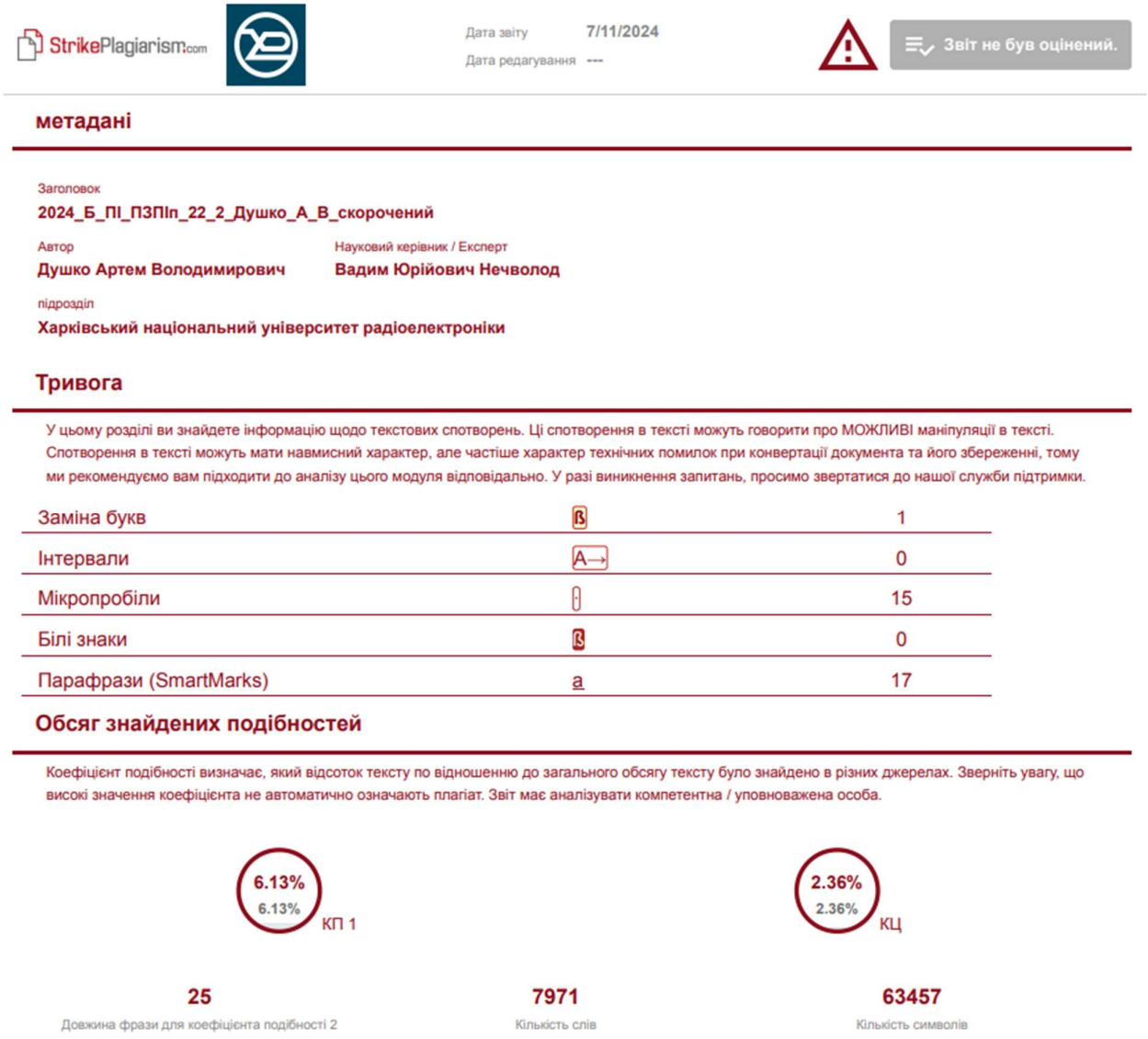
5. The Bootstrap CSS Framework. Official documentation. URL: https://getbootstrap.com/docs/4.1/getting-started/introduction/ (дата звернення 25.06.2024).

6. Russ Unger, Carolyn Chandler. UX design for user experience designers in the field or in the making. Peachpit Press, 2009. 289 с.

7. Writing Django views. Best practices and recommendations. URL: https://docs.djangoproject.com/en/4.2/topics/http/views/ (дата звернення 20.06.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



StrikePlagiarism.com

Дата звіту 7/11/2024
Дата редагування ---

Звіт не був оцінений.

метадані

Заголовок
2024_Б_ПІ_ПЗПІп_22_2_Душко_А_В_скорочений

Автор
Душко Артем Володимирович

Науковий керівник / Експерт
Вадим Юрійович Нечволод

підрозділ
Харківський національний університет радіоелектроніки

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	1
Інтервали	A→	0
Мікропробіли	·	15
Білі знаки	ß	0
Парафрази (SmartMarks)	a	17

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

6.13% 6.13% КП 1

2.36% 2.36% КЦ

25
Довжина фрази для коефіцієнта подібності 2

7971
Кількість слів

63457
Кількість символів

ДОДАТОК Б

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Кваліфікаційна робота бакалавра

Програмна система онлайн кінотеатру

Виконав :
студент гр. ПЗПІп -22-2
Душко А. В.

Керівник :
ст. викл. каф. ПІ
Олійник О.В.

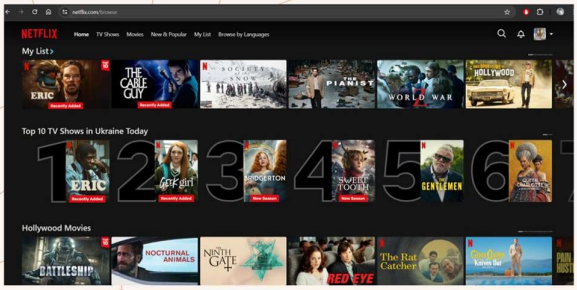
1

МЕТА РОБОТИ

Розробка системи онлайн кінотеатру

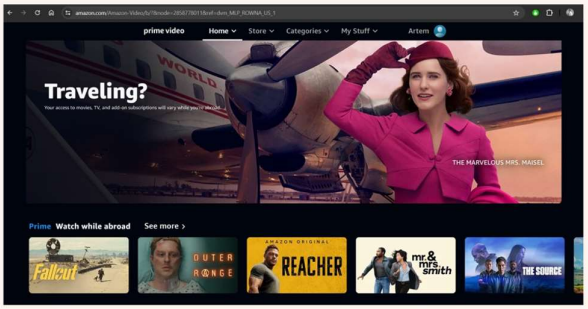
- реєстрація та аутентифікація користувачів
- редагування особистих даних та вибір тарифного плану
- фунціонал пошуку та фільтрації кінофільмів
- додавання кінофільмів у власний список перегляду
- оцінювання вражень та перегляд загальної статистики

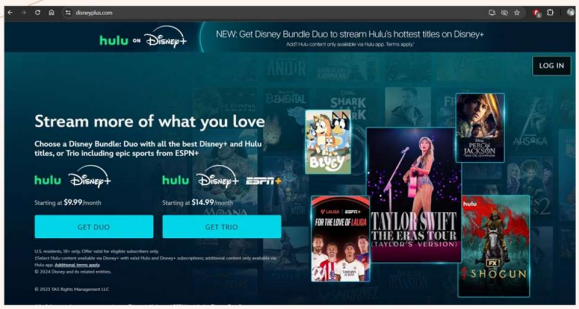
2



Netflix

Amazon Prime





Disney+

3

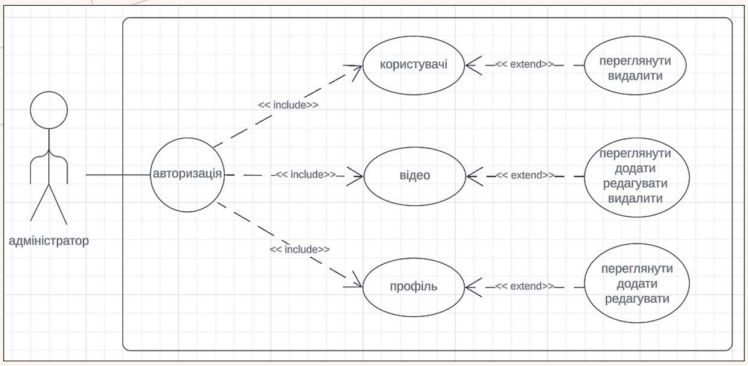
АНАЛІЗ АНАЛОГІВ

	Netflix	Amazon Prime	Disney+
Кількість контенту	~4,000 фільмів, ~2,400 шоу	~18,000 фільмів, ~2,100 шоу	~1,000 фільмів, ~500 шоу
Наявність ексклюзивів	Так	Так	Так
Кількість профілів	5	6	7
Батьківський контроль	Так	Так	Так
Міжнародне покриття	190+ країн	200+ країн	50+ країн
Тестовий період користування	Ні	30 днів	7 днів
Мобільні ігри	Так	Ні	Ні
Офлайн режим перегляду	Так	Так	Так
Підтримка Ultra HD	Так	Так	Так
Підтримка 3D Sound	Так	Так	Так
Додаткові послуги	Ні	Знижки в магазині	Ні
Вартість	$15.49/міс.	$14.99/міс.	$7.99/міс.

4

Постановка задачі та опис системи

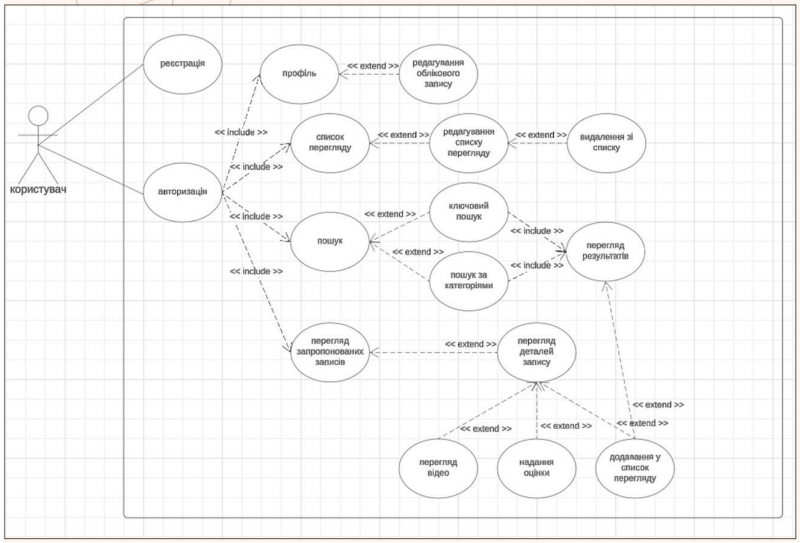
Спроектувати та реалізувати програмну систему онлайн кінотеатру



1. Адміністратор

- керує обліковими записами користувачів
- редагує та оновлює відео контент

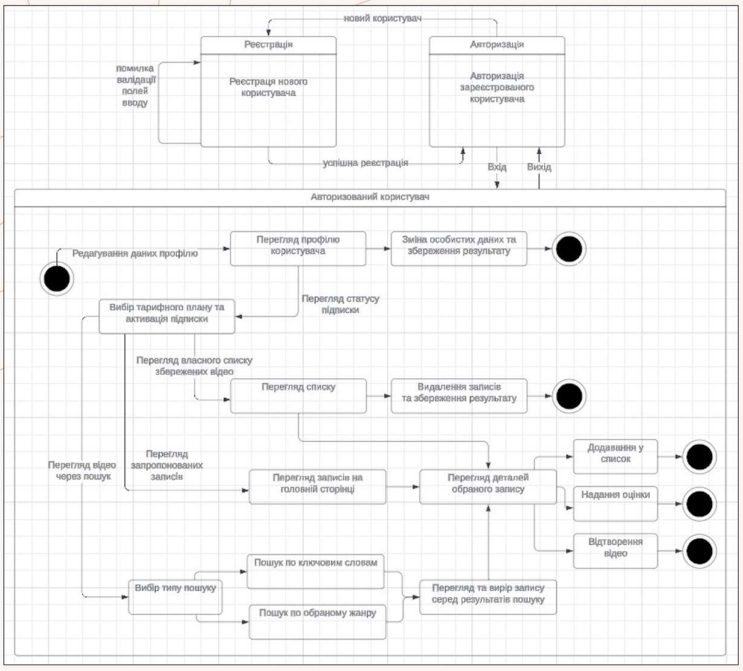
5



2. Користувач

- створює та оновлює обліковий запис
- обирає зручний тарифний план
- шукає та фільтрує контент
- переглядає рейтинг та додає контент у власний список перегляду
- переглядає контент у зручний час
- оцінює відео з власних вражень

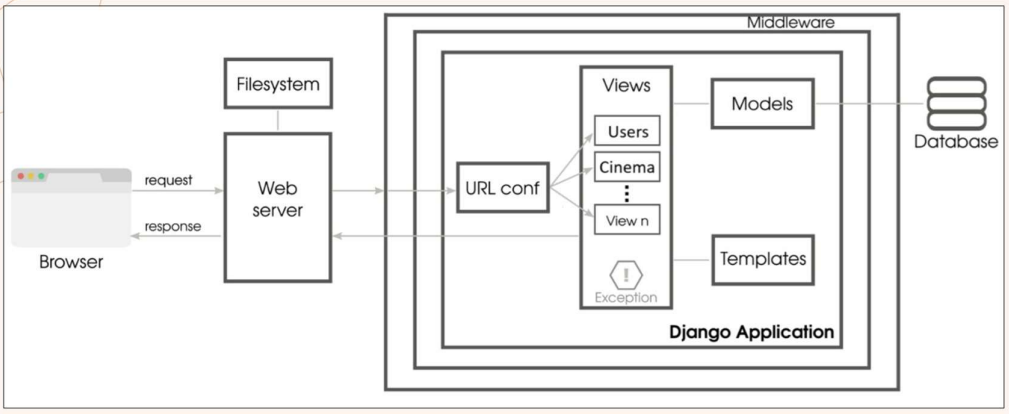
6



Діаграма станів

7

Діаграма розгортання



Django framework

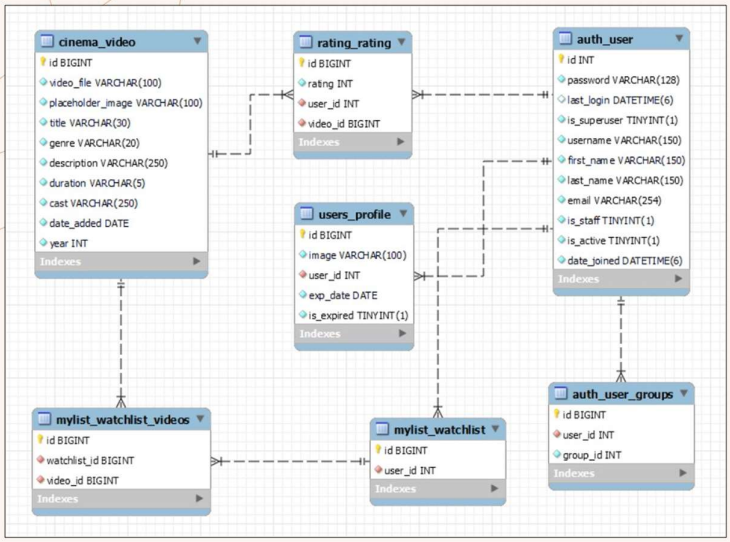
8

Вибір технологій розробки

- Django framework
- Основна мова програмування – Python
- Мова програмування для динамічного контенту – JavaScript
- CSS framework Bootstrap
- СУБД MySQL Workbench 8.0 CE

9

Логічна модель БД



ER діаграма

10

Приклад реалізації та поєднання фільтрації та ключового пошуку за назвою

```
# cinema/views.py
@login_required
def home(request):
    if request.user.profile.is_expired:
        return redirect('profile')
    selected_genre = request.GET.get('genre', '')
    title = request.GET.get('title', '')
    videos = Video.objects.all()

    if selected_genre:
        videos = videos.filter(genre=selected_genre)

    if title and len(title) >= 3:
        videos = videos.filter(title__icontains=title)

    genres = Video.objects.values_list('genre',
flat=True).distinct()
    return render(request, 'cinema/home.html', {'videos':
videos, 'genres': genres, 'selected_genre': selected_genre,
'title': title})
```

11

Приклад виводу результатів пошуку за назвою та фільтрацією за жанром

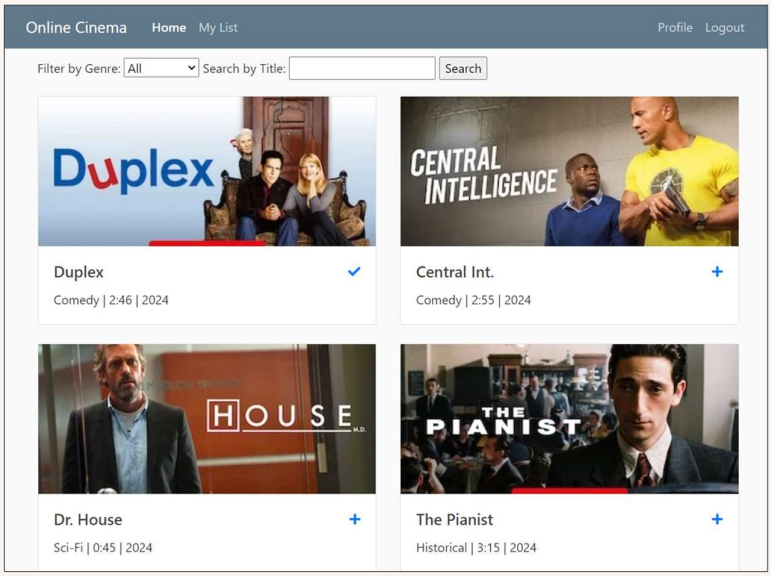
```
# cinema/templates/cinema/home.html
<div class="container">
  <form method="get" class="row mb-3">
    <div class="col-md-8">
        <label for="genre">Filter by Genre:</label>
        <select name="genre" id="genre" onchange="this.form.submit()">
            <option value="">All</option>
            {% for genre in genres %}
            <option value="{{ genre }}" {% if selected_genre == genre %}selected{% endif
%}>{{ genre }}</option>
            {% endfor %}
        </select>
        <label for="title">Search by Title:</label>
        <input type="text" id="title" name="title" value="{{ title }}">
        <input type="submit" value="Search">
    </div>
</form>
```

12

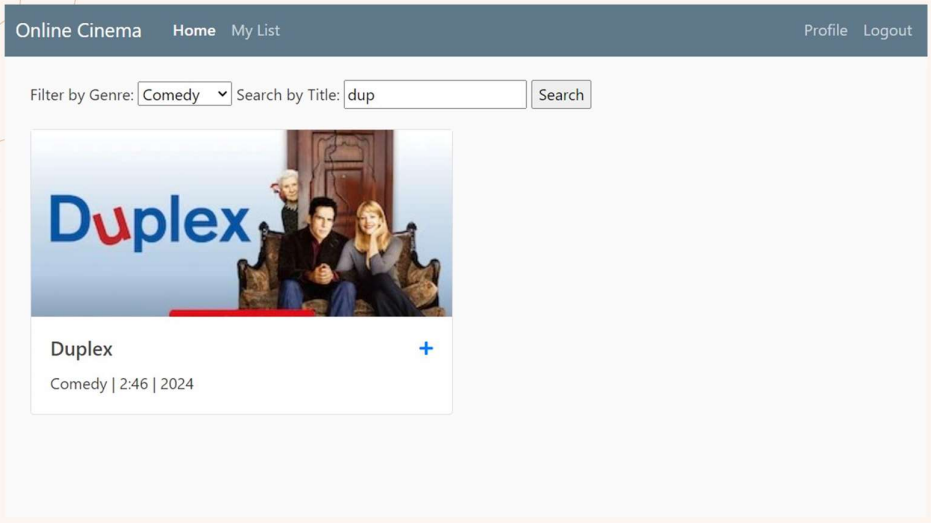
Приклад виводу результатів пошуку за назвою та фільтрацією за жанром

```
# cinema/templates/cinema/home.html
{% for video in videos %}
<div class="col-md-6">
    <div class="card mb-4">
        <div class="card-img-top">
            <img src="{{ video.placeholder_image.url }}" alt="{{ video.title }}" style="width: 100%; height:
192px; object-fit: cover;">
        </div>
        <div class="card-body">
            <div class="row">
                <div class="col-md-8">
                    <a href="{% url 'cinema-detail' video.id %}">
                        <h5 class="card-title">{{ video.title }}</h5>
                    </a>
                    <p class="card-text">{{ video.genre }} | {{ video.duration }} | {{ video.year }}</p>
                </div>
                <div class="col-md-4 text-right">
                    <a href="#" class="toggle-watch-list" data-video-id="{{ video.id }}">
                    <i class="fas fa-plus"></i>
                    </a>
                </div>
            </div>
        </div>
    </div>
</div>
{% if forloop.counter|divisibleby:2 and not forloop.last %}
```


Інтерфейс користувача Домашня сторінка (home)

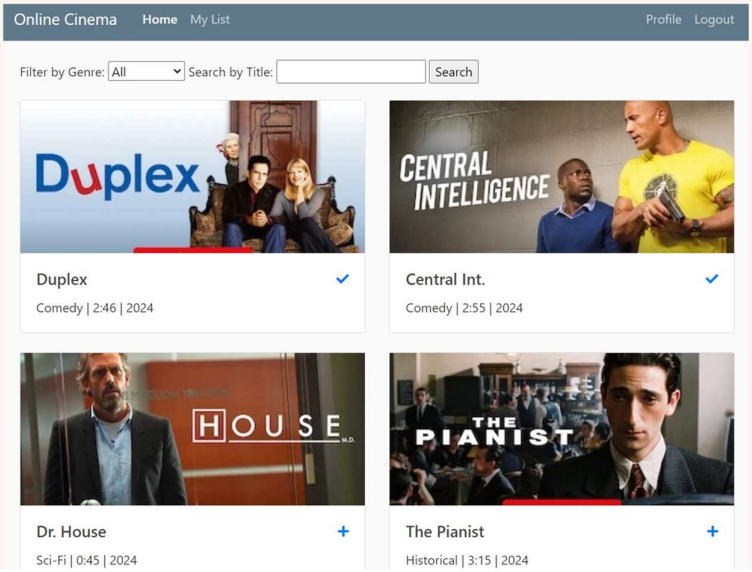


Інтерфейс користувача
Результат пошуку з фільтром



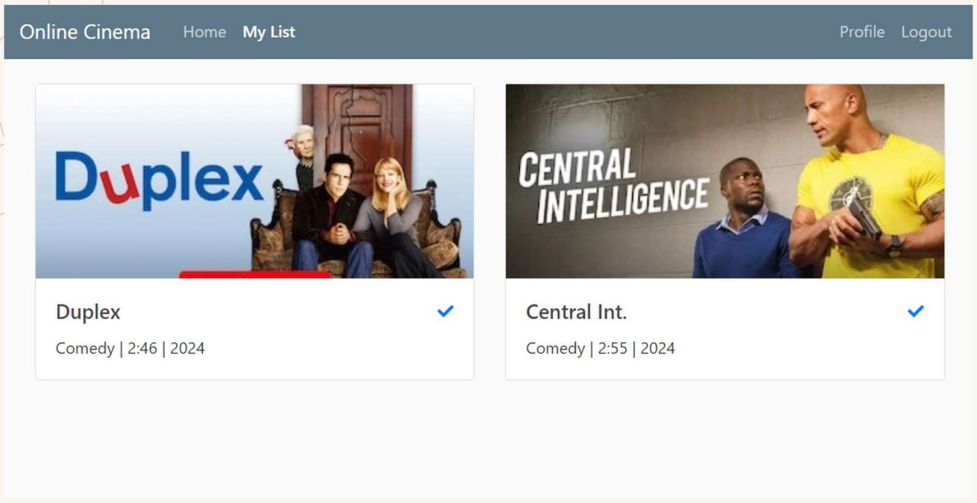
15

Інтерфейс користувача
Додавання у власний список перегляду



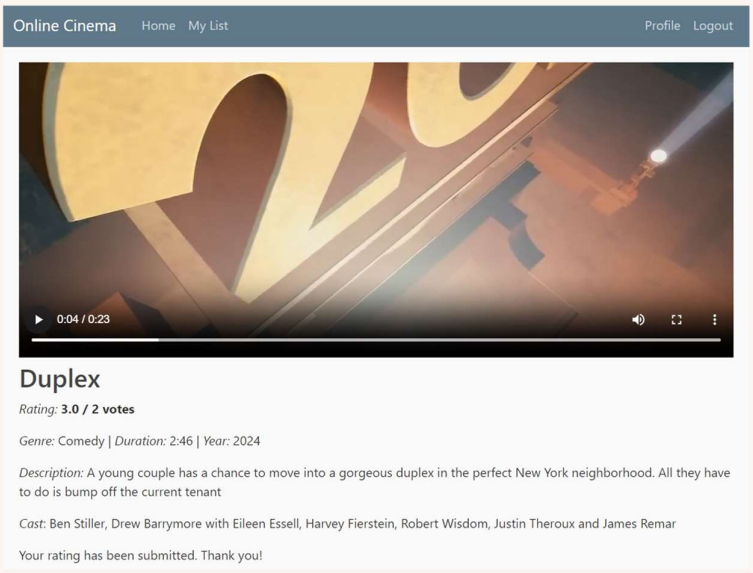
16

Інтерфейс користувача
Власний список перегляду



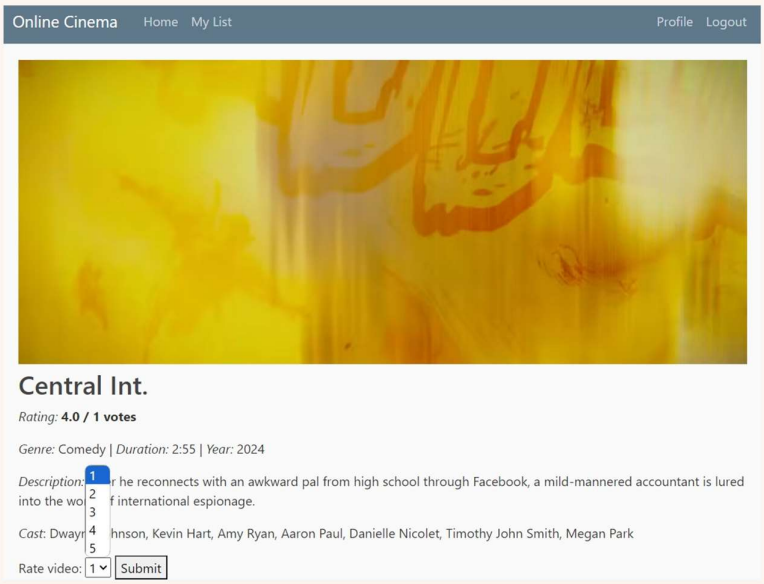
17

Інтерфейс користувача
Сторінка деталей відео



18

Інтерфейс користувача
Форма оцінювання вражень



Тестування
Postman

Перевірка правильності виборки за параметрами:

- http://onlinecinema.local/?genre=Historical&title=

Очікуваний результат:

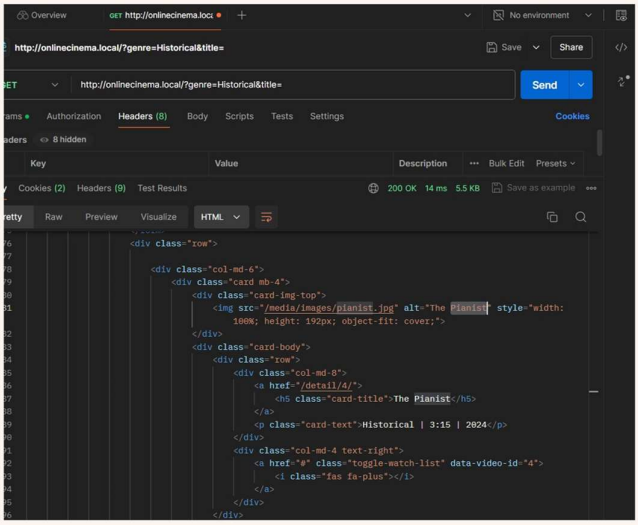
- вивід всіх відео за вказаним жанром

Результат перевірки Postman:

- Виведено всі відео за жанром «Historical»

Результат manual перевірки через веб браузер:

- Співпадає з результатами Postman



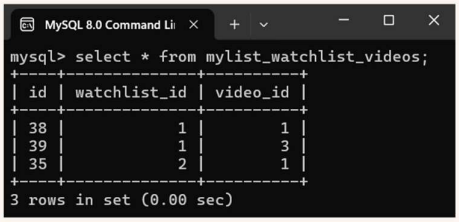
Manual тестування видалення відео з власного списку

Початкові умови:
- 2 відео у власному списку

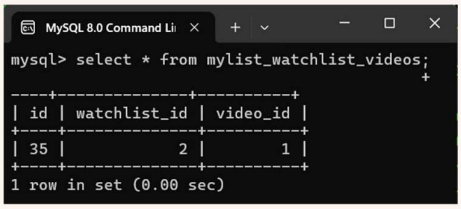
Очікуваний результат:
- 2 відео видалено зі списку
- відповідні записи видалено з таблиці «mylist_watchlist_videos»
- відсутні повідомлення про помилки

Результат тестування:
- відповідає очікуванням

Початковий стан



Результат



21

Практична користь

Самообслуговування
- користувач системи сам обирає тарифний план та відео для перегляду, не займаючи час працівника

Незалежність
- користувач обирає зручний час для перегляду та не залежить від жодних розкладів

Покриття
- доступ до системи без часових та територіальних обмежень

Перспективи розвитку
- розвиток україномовного контенту через залучення місцевих студій, в тому числі перекладу
- розвиток у регіонах та покриття зростаючих ринків
- фестивалі авторського кіно

22

Висновки

- проведено аналіз предметної галузі
- визначено та порівняно переваги та недоліки конкурентів
- обґрунтовано конкурентні переваги
- сформульовано загальне бачення розвитку програмної системи
- виділено бачення цільової аудиторії та напрямків масштабування
- спроектовано архітектуру з демонстрацією та поясненням взаємодії окремих підсистем
- реалізовано та протестовано програмну систему