

**Пояснювальна записка  
до курсової роботи**

на тему: Система підтримки діяльності онлайн кінотеатру

КП.ІІ-0107.045450.01.81

Київ – 2023

## ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	4
ВСТУП.....	5
1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ .....	7
1.1 Загальні положення .....	7
1.2 Змістовний опис і аналіз предметної області .....	9
1.3 Аналіз існуючих технологій та успішних ІТ-проектів .....	10
1.3.1 Аналіз відомих алгоритмічних та технічних рішень .....	11
1.3.2 Аналіз допоміжних програмних засобів та засобів розробки.....	16
1.3.3 Аналіз відомих програмних продуктів.....	17
1.4 Аналіз вимог до програмного забезпечення .....	21
1.4.1 Розроблення функціональних вимог .....	30
1.4.2 Розроблення нефункціональних вимог .....	36
1.5 Постановка задачі .....	37
Висновки до розділу .....	38
2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	40
2.1 Моделювання та аналіз програмного забезпечення.....	40
2.2 Архітектура програмного забезпечення.....	42
2.3 Конструювання програмного забезпечення.....	44
2.4 Аналіз безпеки даних .....	50
Висновки до розділу .....	50
3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	52
3.1 Аналіз якості ПЗ.....	52
3.2 Опис процесів тестування.....	54
3.3 Опис контрольного прикладу .....	60
Висновки до розділу .....	73
4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.	75
4.1 Розгортання програмного забезпечення.....	75
4.2 Підтримка програмного забезпечення.....	77

Висновки до розділу .....	77
ВИСНОВКИ.....	79
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	81

## **ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ**

БД	– База даних.
TMDB	– The Movie Database
2FA	– Two-factor authentication
API	– Application Programming Interface
MFA	– Multi-factor authentication
JS	– JavaScript
DOM	– Document Object Model
IDE	– Integrated development environment
ПЗ	– Програмне забезпечення

## ВСТУП

В сучасному онлайн суспільстві, зросла необхідність у зручних та функціональних інструментах для взаємодії з медіа контентом. Онлайн кінотеатри виявилися однією з найпопулярніших форм розваг та розвитку в сучасному світі. Разом із зростанням цієї популярності, з'явилася потреба у вдосконаленні сервісів для забезпечення якісного користувацького досвіду.

Онлайн кінотеатри стають зручним та доступним засобом для споживачів з метою отримання доступу до широкого асортименту відомих та нових фільмів, мультфільмів, серіалів тощо. Проте, багато існуючих платформ мають певні обмеженнями та неефективно вирішувати завдання, пов'язані зі зручністю використання, безпекою та взаємодією зі споживачами.

Мета даного курсового проекту — розробити та впровадити веб-застосунок, який максимально задовольнить потреби користувачів онлайн кінотеатрів, забезпечуючи високий рівень зручності, ефективності та безпеки використання. Основні завдання включають у себе:

- Аналіз вимог до програмного забезпечення для вивчення функціоналу та особливостей існуючих онлайн кінотеатрів, а також визначення та формулювання вимог до нового застосунку;
- Моделювання та конструювання програмного забезпечення через визначення та реалізацію основних компонентів;
- Аналіз якості та тестування програмного забезпечення через проведення різних видів тестів для перевірки функціональності та надійності застосунку;
- Розгортання системи у реальному середовищі, надання доступу користувачам онлайн та визначення методів підтримки роботи програмного забезпечення.

Щодо сучасних тенденцій у сфері онлайн кінотеатрів визначають важливі аспекти, які слід враховувати при розробці нового продукту. Це включає в себе підвищення вимог до безпеки, зручності та доступності для різних категорій

користувачів та використання сучасних хмарних технологій для підвищення швидкодії та надійності.

Застосунок Mediatoria може бути використаний в різних сферах та галузях, де важливим є доступ користувачів до якісного та різноманітного відеоконтенту. Це може включати в себе широкий спектр груп людей-любителів кіно.

Враховуючи вищезазначені фактори, курсовий проект присвячений розробці веб-застосунку, який відповідає сучасним вимогам та тенденціям у сфері онлайн розваг та розвитку.

# 1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 1.1 Загальні положення

Онлайн-кінотеатри стали невід'ємною частиною сучасного життя пропонуючи користувачам можливість дивитися фільми і телепрограми з будь-якого пристрою з доступом до інтернету. Онлайн-кінотеатри використовують різні технології для забезпечення інтерактивного перегляду. Загалом, такі системи можуть пропонувати різноманітні функції, серед яких можна виділити наступні:

- 1) Перегляд фільмів за категоріями або жанрами;
- 2) Оцінка фільмів;
- 3) Пошук конкретного фільму для перегляду;
- 4) Коментування фільмів;
- 5) Формування персональних списків фільмів

Отже, наша система підтримки діяльності онлайн кінотеатру має забезпечити реалізацію цих функцій.

Оскільки система має надавати користувачам можливість переглядати фільми, а також утримувати інформацію про користувачів, їх збережені фільми та іншу додаткову інформацію, то доцільним буде використання БД. Для зберігання даних буде використана база даних Firestore.

Firestore – це NoSQL-база даних, яка надає високодоступні та масштабовані можливості зберігання даних[1]. Firestore використовує хмарну інфраструктуру, що дозволяє легко масштабувати та забезпечувати високу доступність даних. Дані організовані у вигляді документів у колекціях, що дозволяє ефективно взаємодіяти з ними та отримувати швидкий доступ до необхідної інформації.

Також Firestore підтримує режим реального часу, що дозволяє миттєво отримувати зміни у даних без необхідності оновлення сторінок чи додатків. База даних дозволяє додавати нові, видаляти старі та редагувати поля до існуючих документів без перерви в роботі системи через запити.

Звичайно, застосунок, що має можливість аутентифікації користувачів має надавати надійний функціонал, що забезпечить безпеку даних. Отже, для створення особистих облікових записів у системі підтримки діяльності онлайн кінотеатру буде використана система Firebase Auth.

Firebase Auth — це система аутентифікації, яка надає різні методи аутентифікації[2]. В нашому випадку оберемо аутентифікацію через електронну пошту та пароль. Цей метод є найпоширенішим і забезпечує достатній рівень безпеки.

Firebase Auth забезпечує безпеку даних користувачів за допомогою наступних заходів:

- 1) Хешування паролів: паролі користувачів хешуються перед їхнім зберіганням у базі даних[3]. Це ускладнює відновлення паролів у разі їхнього втрати;
- 2) Двофакторна аутентифікація (2FA): 2FA надає додатковий рівень безпеки, вимагаючи від користувача ввести код, який надсилається на його пристрій під час входу в систему[4];
- 3) Безпека даних на рівні сервера: Firebase Auth реалізує ряд заходів безпеки на рівні сервера, таких як фільтрація вхідних даних та перевірка безпеки API[5].

Для розробки веб-застосунку системи підтримки діяльності онлайн кінотеатру буде використана технологія React.

React - це бібліотека для створення інтерфейсів користувача, яка була розроблена Facebook[6]. Вона дозволяє розробникам створювати великі веб-додатки, які використовують дані, які можуть змінюватися з часом, без перезавантаження сторінки. Це робить його ідеальним для створення сучасних односторінкових веб-додатків[7].

React має ряд переваг, які роблять його придатним для розробки веб-застосунків для онлайн кінотеатру:



- Компонентність: React дозволяє розробляти інтерфейс користувача з використанням компонентів. Це спрощує розробку та обслуговування коду;
- Декларативність: React-код є декларативним, що означає, що розробник описує, що він хоче побачити на екрані, а не те, як це досягти. Це робить код більш зрозумілим і читабельним;
- Ефективність: React використовує техніку, відому як Virtual DOM, для ефективного рендерингу інтерфейсу користувача. Це дозволяє React швидко та плавно оновлювати інтерфейс користувача при зміні даних.

Для реалізації функціональності системи підтримки діяльності онлайн кінотеатру буде використана React-бібліотека React Router. Вона дозволяє розробляти веб-застосунки з динамічними маршрутами[8]. React Router використовує техніку, відому як компоненти маршрутизації, для визначення маршрутів у веб-застосунку. Компоненти маршрутизації відповідають за відображення вмісту сторінки, яка відповідає певному маршруту.

## 1.2 Змістовний опис і аналіз предметної області

На сучасному етапі розвитку ІТ-технологій індустрія розваг, зокрема сфера онлайн кінотеатрів, є однією з перспективних та швидко розвиваючих галузей. Вона відзначається стрімким зростанням популярності та поширенням сервісів для перегляду відео контенту в Інтернеті.

Існує багато платформ для перегляду та прокату фільмів, які пропонують користувачам широкий спектр контенту, зокрема фільми, серіали, документальні фільми та телешоу. Однак ці платформи також мають низку недоліків, які можуть негативно вплинути на користувацький досвід.

На сьогоднішній день, використання ІТ-технологій у сфері онлайн кінотеатрів стало нормою. Програмне забезпечення, яке використовується для підтримки діяльності онлайн кінотеатру, включає в себе різні компоненти, такі

як системи управління контентом, аутентифікація користувачів, зберігання та обробка даних.

Однак, незважаючи на широке використання цих технологій, вони мають й певні недоліки. Наприклад, багато систем не забезпечують достатнього рівня безпеки даних, що може призвести до витоку приватної інформації користувачів. Крім того, деякі системи можуть бути складними у використанні, що знижує задоволеність користувачів.

Також, однією з ключових вимог сучасних глядачів є забезпечення зручного та комфортного середовища для перегляду фільмів. Недостатня функціональність та низька інтерактивність деяких існуючих онлайн кінотеатрів може призводити до негативного досвіду використання.

Інтеграція функцій взаємодії користувачів, таких як коментування, оцінювання та обмін рекомендаціями, допоможе створити сприятливе комунікаційне середовище серед глядачів.

Розробка системи персоналізації контенту та зручного управління списку улюблених фільмів дозволить кожному користувачеві створити унікальний досвід перегляду.

В рамках цієї курсової роботи, ми обрали шлях розробки веб-застосунку для онлайн кінотеатру “Mediatoria”, який не тільки дозволяє користувачам переглядати фільми онлайн, але й взаємодіяти з системою через коментарі та оцінки. Наша мета - створити зручне середовище та інтерфейс для користувачів, де вони зможуть переглядати, оцінювати, коментувати та зберігати фільми з різних пристроїв. Ми використовуємо React для розробки фронтенду, Firebase та Firestore для аутентифікації та утримання даних, а також TMDb для завантаження фільмів. Це дозволяє нам вдосконалити процес використання знань предметної області в програмному забезпеченні.

### 1.3 Аналіз існуючих технологій та успішних IT-проектів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації веб-застосунку для

підтримки діяльності онлайн-кінотеатрів «Mediatoria». Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

### 1.3.1 Аналіз відомих алгоритмічних та технічних рішень

Реєстрація та аутентифікація користувачів – ключовий етап створення будь-якої системи, і вибір правильного методу грає критичну роль у забезпеченні безпеки, ефективності та зручності для користувача застосунка. В ході створення системи необхідно визначити методи для реєстрації та аутентифікації. Отже, спробуємо визначити підходити для вирішення цієї задачі:

- Використання локальної БД, що є простим рішенням у реалізації, проте воно не дозволяє масштабувати застосунок через перевантаженість системи та сама система може бути вразливою до атак;
- Використання сторонньої служби БД (Google, Facebook), що дозволяє масштабувати застосунок, оскільки навантаження розподіляється між окремими серверами. Надійність гарантується службою.
- Використання власної сторонньої служби. Вона забезпечує повний контроль над процесом підтвердження особи, але потребує найбільших зусиль у реалізації та підтримці.

Як видно з переліку найбільш гарантованим та комфортним у використанні є метод отримання послуг від сторонніх служб. Одною з таких є Firebase Auth через зручну інтеграцію за допомогою API.

Firebase Auth - це сторонній сервіс автентифікації, що надається компанією Google[9]. Firebase Auth пропонує широкий спектр функцій, які дозволяють легко реалізувати реєстрацію користувачів у веб-додатках. До переваг Firebase Auth можна віднести:

- Економія часу на розробку методів аутентифікації;
- Отримання детальної аналітики та демографічної інформації про користувачів;
- Інтеграція з іншими сервісами Firebase, що полегшує спільне використання сервісів;

- Використання таких стандартів, як OAuth 2.0 та OpenID Connect, які можна легко інтегрувати з власним бекендом;
- Захист даних користувачів шляхом надання безпечних методів аутентифікації;
- Налаштовування різноманітних методів аутентифікації, включаючи адресу електронної пошти та пароль, телефонні номери, а також зовнішні провайдери, такі як Google, Facebook, або Twitter;
- Автоматична обробка усіх аспектів безпеки;
- Використання мультифакторної аутентифікації, забезпечуючи додатковий шар безпеки через використання додаткового фактору, такого як SMS-код чи аутентифікація на пристрої[10].
- Управління правами доступу з визначенням рівня доступу для кожного типу користувачів з наданням їм управління даними[11]. Оскільки маємо 3 можливих стани користувачів (неавторизований, звичайний користувач та адміністратор), то даний функціонал стане у нагоді.

Задача зберігання даних користувача є однією з найголовніших у веб-додатках. У нашому випадку нам потрібно зберігати такі дані користувача, як: ім'я, електронна пошта, дата реєстрації, список збережених фільмів, що дозволяє зберігати інформацію про користувачів, таку як їхні імена, паролі, адреси електронної пошти, коментарі та інші дані.

Існує кілька різних методів розв'язання задачі збереження даних користувачів. Найпоширенішими з них є:

- Використання локальної бази даних;
- Використання сторонньої бази даних;
- Використання хмарної бази даних.

Локальна база даних надає можливість зберігати дані користувачів у локальній базі даних додатку. Цей метод простий в роботі, але має ряд недоліків. Зокрема, він не дозволяє масштабувати додаток, оскільки база даних може бути перевантажена великою кількістю користувачів.

При використанні методу сторонньої бази даних дані користувачів розміщуються в сторонній базі даних (MySQL[12], PostgreSQL[13] або MongoDB[14]). Цей метод вже забезпечує масштабованість, оскільки навантаження розподіляється між різними незалежними серверами баз даних.

Хмарні бази даних надають можливість зберігати всі необхідні дані в хмарі (Google Cloud Platform[15], Amazon Web Services[16] або Microsoft Azure[17]). Вони використовуються для структурованого зберігання інформації та надають потужні можливості запитів для пошуку та маніпулювання даними. Завдяки своїй зручності, стабільності та функціональності вони мають перевагу над іншими методами, що використовуються.

Firestore - це хмарна база даних NoSQL від Google. Вона надає гнучке, масштабоване сховище даних для мобільних, веб- та серверних додатків. Серед найважливіших переваг інструменту можна відзначити наступні[18]:

- Асинхронні запити;
- Взаємодія з базою даних за допомогою простих запитів, які можна використовувати для виконання складних NoSQL запитів;
- Забезпечення синхронізації в режимі офлайн, що допомагає користувачам взаємодіяти з застосунком за відсутності мережевого з'єднання;
- Автоматичне горизонтальне масштабування відповідно до навантаження[19];
- Використання протоколу HTTPS для шифрування передачі даних та вбудованих механізмів безпеки, які визначають правила захисту від спроб несанкціонованого доступу.

Ще однією проблемою є наповнення застосунку контентом. Оскільки фільмотека має бути різноманітною та актуальною, потрібно вирішити, як отримувати дані про фільми.

Одним з методів є власноручне додавання фільмів. В цьому випадку, адміністратори будуть вручну додавати та оновлювати інформацію про фільми. Цей метод часом використовується в невеликих проектах, проте він потребує

багато часу та є неефективний для великих обсягів даних. Отже, необхідно отримувати інформацію з окремих публічних джерел.

Наш застосунок буде використовувати TMDb API[20]. Отримання даних з БД, у такому випадку, здійснюється через використання API. API TMDb є вирішенням вищеописаної проблеми та також має додаткові переваги[21]:

- Велика кількість даних;
- Надання простого у використанні інтерфейсу, що дозволяє розробникам легко отримувати дані про фільми;
- Розроблений для масштабування, що дозволяє йому обробляти великі обсяги запитів;
- Розширені функції пошуку фільмів за різними критеріями (жанр, рік виходу фільмів).

HTTP-запити - це основний спосіб взаємодії веб-додатків із зовнішніми ресурсами, такими як сервери API, веб-сайти та бази даних. Існує багато методів та технологій здійснення HTTP запитів у JavaScript, проте ми зазначимо найактуальніші[22].

Першим зазначимо доволі старий метод – XMLHttpRequest. Цей інтерфейс є вбудованим та відомим багатьом розробникам. Однак його API є менш зручним та потребує багато коду для реалізації навіть простих запитів.

Іншим сучасним методом є Fetch API, що був введений у стандарті HTML 2015. Він підтримує проміси та забезпечує гнучкий API для виконання запитів до серверів та роботи з відповідями[23]. Можна було б спокійно обрати Fetch API, проте він все ж має більш обмежені функції в порівнянні з Axios та не є доступним у всіх браузерах старих версій[24].

Axios – це бібліотека JavaScript для виконання HTTP-запитів з браузера або Node.js. Вона надає простий у використанні API і підтримує різні функції, включаючи запити потокового передавання. Має наступні переваги[25]:

- Використовує Promise API, що дозволяє використовувати async/await, щоб зробити код здійснення запитів більш зрозумілим та синхронним[26];

- Працює на різних браузерах, забезпечуючи однакову поведінку;
- Автоматичне перетворення даних JSON;
- Інтерцептори HTTP – здатність перехоплювати HTTP-запити для дослідження або зміни HTTP-запитів від вашого додатку до сервера або навпаки (реєстрація, аутентифікація, повторний запит);
- Підтримка запитів потокового передавання, що дозволяє обробляти дані по мірі їх надходження, а не чекати отримання всієї відповіді[27].

Розробка фронтенду на JavaScript включає в себе різні інструменти та бібліотеки, які розробники використовують для створення привабливих та ефективних веб-застосунків. Звичайно, стандартний JS вже надає інструментарій для створення інтерфейсів[28]. Проте, він не пропонує функціонал, що зробить процес розробки легшим. У цьому контексті, були створені альтернативи до використання стандартного JS якими є:

- Бібліотека jQuery, що має більший функціонал ніж стандартна версія JS[29]. Вона була популярна в минулому для спрощення роботи з DOM та взаємодії зі стороною клієнта. Проте, із розвитком сучасних бібліотек та фреймворків, використання jQuery визнається менш ефективним.
- Angular, React та Vue – фреймворки JavaScript, які надають повний набір інструментів для розробки веб-додатків[30]. Angular та Vue схожі у використанні, проте саме Vue визначається своїм прогресивним підходом та гнучкістю. У свою чергу, React відомий за свій декларативний синтаксис та компонентний підхід.

Звичайно, що для розробки фронтенду необхідно обрати актуальний варіант з повним набором інструментів. В нашому випадку оберемо React, що додатково має наступні переваги:

- 1) Використання компонентної архітектури, дозволяє розробникам створювати повторно використовувані компоненти для інтерфейсу користувача;
- 2) Використання віртуального DOM, який дозволяє додатку ефективно оновлювати та відображати компоненти;

- 3) Використання моделі рендерингу на стороні сервера (SSR), яка дозволяє генерувати HTML-код на сервері. Це робить додатки швидшими за додатки, створені на інших фреймворках.

### 1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

IDE (Integrated Development Environment) - це комплексне ПЗ, що поєднує в собі текстовий редактор, компілятор, відладчик та інші інструменти, необхідні для розробки програмного забезпечення.

Для розробки на JavaScript існують декілька різних IDE, які можна використовувати:

- 1) Visual Studio Code - це безкоштовний та відкритий код IDE, який підтримує широкий спектр мов програмування, включаючи JavaScript[31]. Visual Studio Code пропонує широкий спектр функцій, включаючи інтелектуальне автозаповнення, відладку, рефакторинг та управління пакетами;
- 2) IntelliJ IDEA - це платний IDE, який розробляється компанією JetBrains. IntelliJ IDEA пропонує широкий спектр функцій, включаючи інтелектуальне автозаповнення, відладку, рефакторинг, управління пакетами та підтримку веб-фреймворків, таких як React, Angular та Vue.js[32];
- 3) WebStorm - це платний IDE, який розробляється компанією JetBrains[33]. Він спеціалізується на розробці веб-додатків на JavaScript. WebStorm пропонує широкий спектр функцій, включаючи інтелектуальне автозаповнення, відладку, рефакторинг, управління пакетами, підтримку веб-фреймворків, таких як React, Angular та Vue.js, а також підтримку хмарних платформ, таких як GitHub та Bitbucket.

В свій час, я мала досвід використання всіх вищеописаних IDE. На мою думку, WebStorm є ряд переваг перед іншими[34], такі як:

- 1) Повноцінна підтримка популярних фреймворків JavaScript, таких як React, Angular, та Vue.js;



- 2) Наявність інструментів автодоповнення коду та перевірки помилок, що робить процес написання коду ефективним, швидким та надійним;
- 3) Можливість бачити зміни у реальному часі за допомогою функції Live Edit;

### 1.3.3 Аналіз відомих програмних продуктів

Розглянемо два програмних продукти, які частково чи повністю реалізують функціонал, описаний у технічному завданні. Такими застосунками є Netflix та IMDB, які є відомими онлайн-платформами, присвячені кіноіндустрії та сфері розваг.

Netflix - це глобальний стрімінговий сервіс, який надає своїм підписникам доступ до великого каталогу фільмів, серіалів, телешоу та іншого розважального контенту (Рисунок 1.1). Netflix доступний у більш ніж 190 країнах світу. Платформа володіє великою кількістю оригінального контенту та функціоналом для зручного перегляду[35].

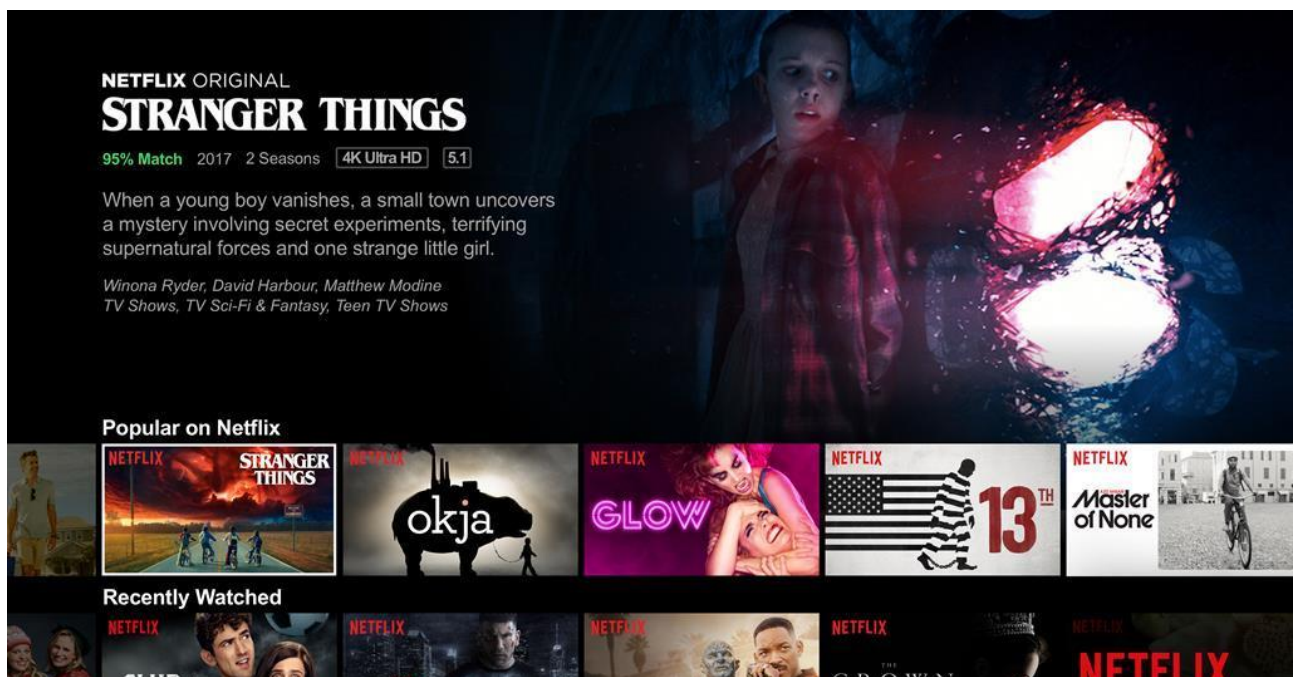


Рисунок 1.1 – Головна сторінка Netflix

Як видно з рисунку 1.1 майже увесь екран заповнений контентом. При вході на головну сторінку вмикається трейлер зі звуком на головному банері. При спробі перевести курсор в інше місце вмикається відповідний трейлер у

вікні обраного фільму. І це є головним мінусом даного сервісу. Він не надає користувачу можливості спокійно обрати контент для перегляду. Натомість, він постійно відволікається на картинки, що постійно змінюються. І в кінці кінців, з більшою вірогідністю користувач обирає фільм з першої половини сайту, де картинка була привабливіша, а не якість.

Ще одним недоліком платформи є те, що користувачі не можуть залишити власний відгук чи коментар до фільму, що обмежує інтерактивність та обмін думками. До цього додамо також те, що користувачі не знають нічого про рейтинг обраного фільму (Рисунок 1.2).

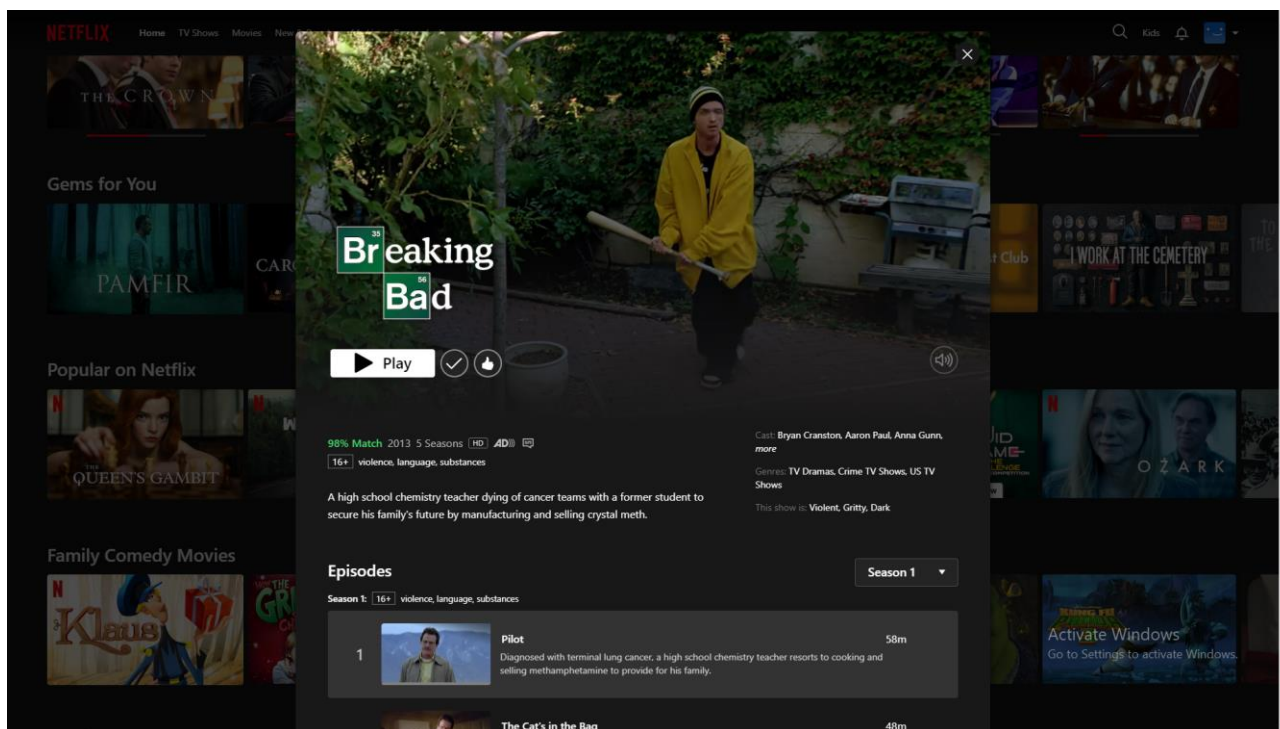


Рисунок 1.2 – Інформаційне вікно фільму

IMDB (Internet Movie Database) - це велика онлайн-база даних про фільми, телесеріали, акторів та інших учасників кіноіндустрії. IMDB надає інформацію про фільми, рейтинги, популярність, новини та інші елементи, що стосуються світу кіно (Рисунок 1.3).

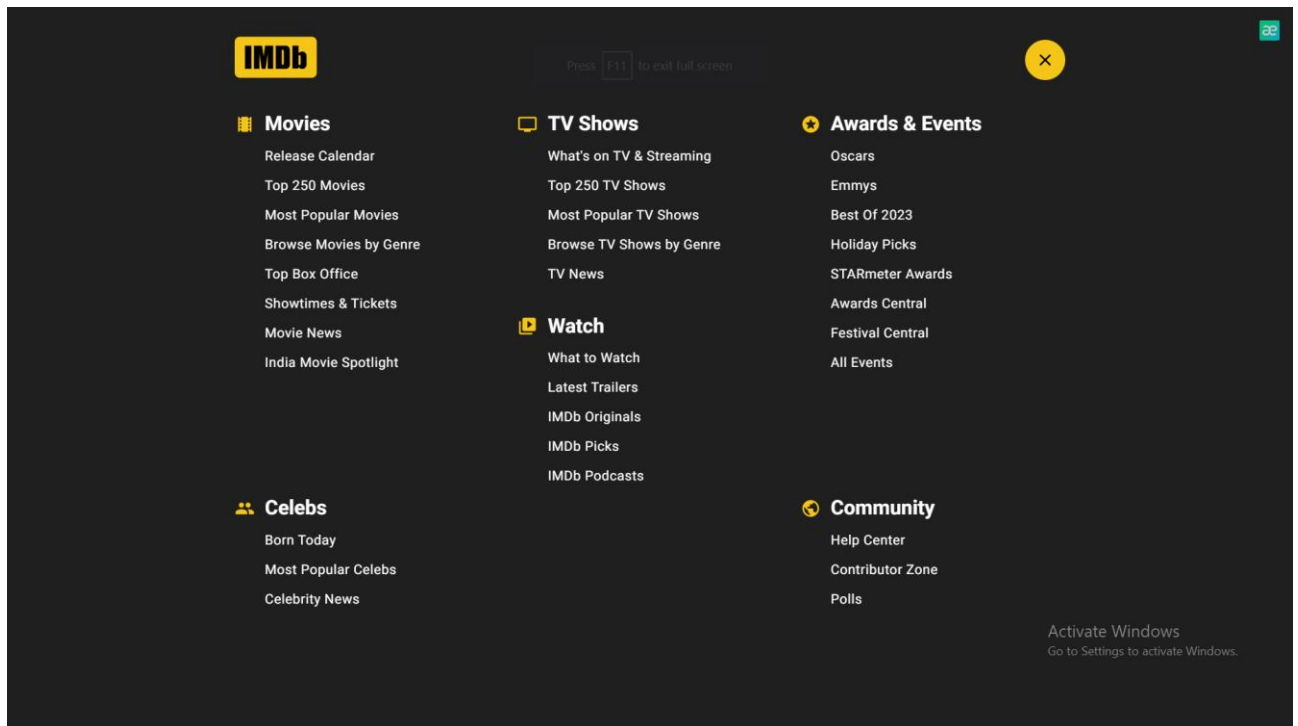


Рисунок 1.3 – Навігація між сторінками IMDb

Порівняно з Netflix на IMDb користувач має можливість писати коментарі до фільмів та залишати їм оцінку (Рисунок 1.4).

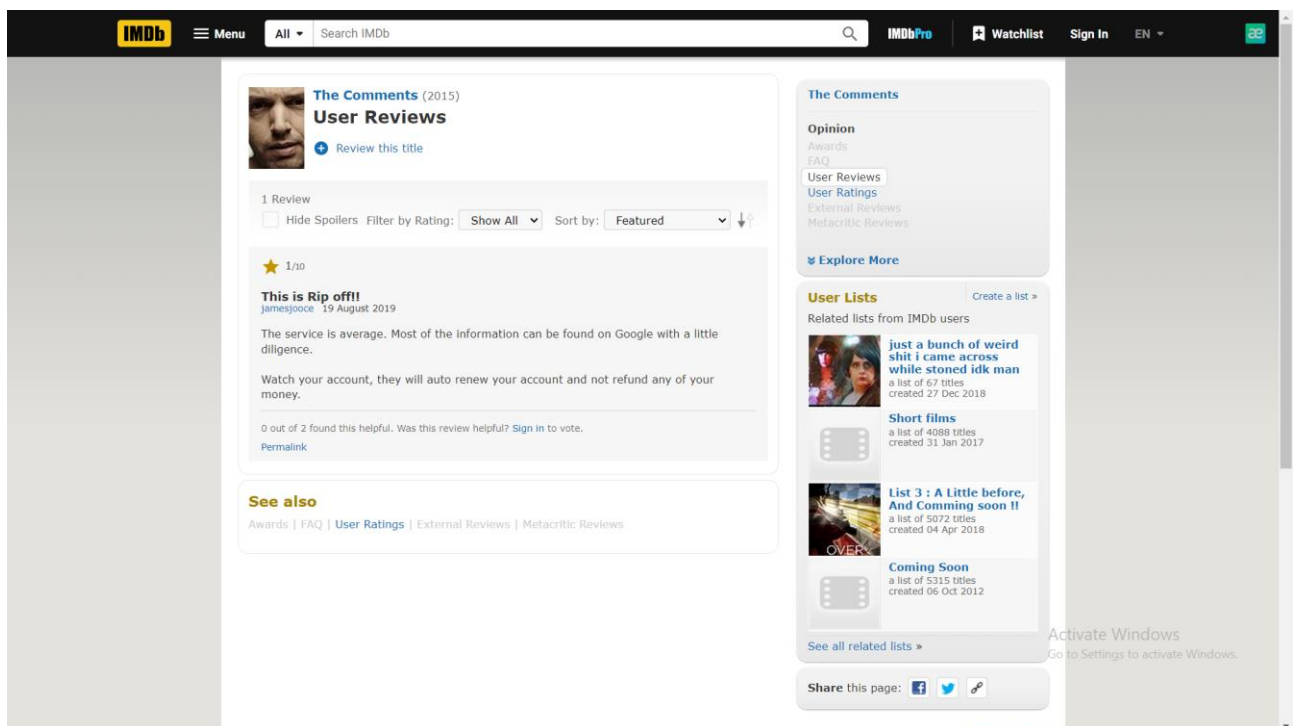


Рисунок 1.4 – Вікно відгуків користувачів з оцінками

IMDb пропонує ряд способів пошуку фільмів, телешоу та інших видах розваг. Користувачі можуть шукати за назвою, режисером, актором, жанром або будь-якою іншою інформацією. Також, ще одною перевагою платформи є те, що

користувачі можуть створювати та ділитися списками фільмів та телешоу, які вони хочуть переглянути.

З недоліків IMDB можна зазначити те, що сервіс не надає можливість перегляду фільмів, обмежуючи користувачів лише отриманням інформації про фільми. Також інформація на сайті може бути неактуальною. IMDB не завжди швидко оновлює інформацію про фільми та телешоу. Наприклад, на сайті може бути інформація про фільм, який уже вийшов з прокату або який був перероблений.

Для порівняння курсової роботи з аналогом можна скористатись таблицею 1.1.

Таблиця 1.1 – Порівняння з аналогом

Функціонал	Mediatoria	Netflix	IMDB	Пояснення
Можливість перегляду фільму	+	+	-	IMDB не надає можливості дивитися фільми.
Фільтрація за визначеними критеріями	+	+-	+	Netflix має систему тегів за якою відбувається фільтрація результату. Проте для звичайного користувача, не є очевидним як, наприклад, знайти фільми певного року.
Можливість написання відгуків	+	-	+	Netflix не надає ніякого функціоналу для написання коментарів чи спілкування користувачів.

Продовження таблиці 1.1

Функціонал	Mediatoria	Netflix	IMDB	Пояснення
Можливість ставити оцінку фільму	+	+ -	+	Netflix дає можливість поставити «Like» фільму, проте не надає можливості перегляду користувацьких рейтингів.
Фільтрація за визначеними точними критеріями	+	+ -	+	Netflix має систему тегів за якою відбувається фільтрація результату. Проте для звичайного користувача, не є очевидним як, наприклад, знайти фільми певного року.
Можливість збереження фільму	+	+	+	
Якісні UI та UX	+	+ -	+	Netflix має забагато відволікаючих факторів, такі як: звуки, автоматичні включення трейлерів, постійна динаміка та відкриваючі вікна

## 1.4 Аналіз вимог до програмного забезпечення

Головною функцією програмного забезпечення є створення середовища для перегляду та оцінки фільмів користувачами та забезпечення простору для обговорення контенту через коментарі. Більше функцій можна побачити на рисунку 1.5.

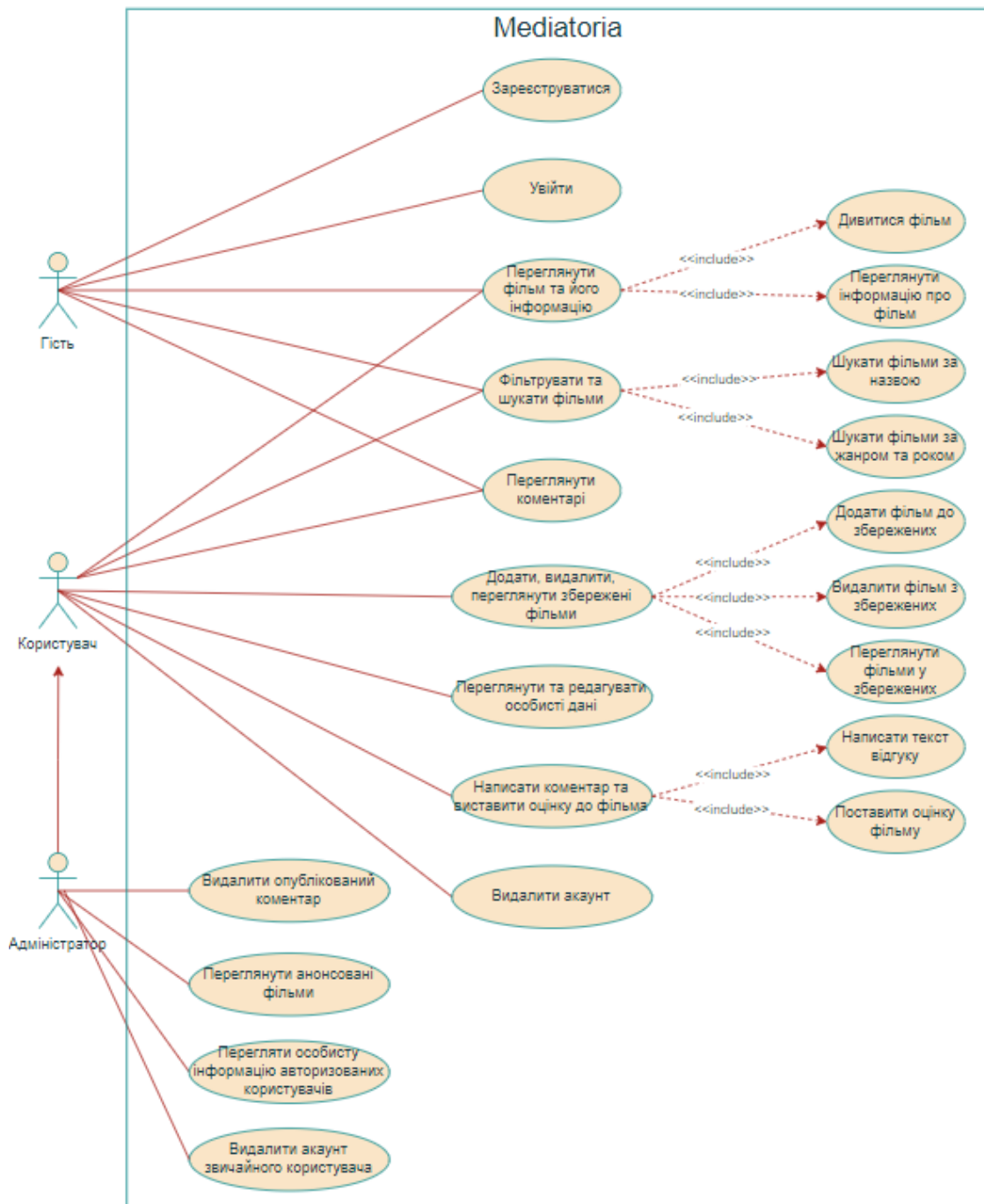


Рисунок 1.5 – Діаграма варіантів використання

В таблицях 1.2 - 1.15 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 - Варіант використання UC-1

Use case name	Реєстрація користувача
Use case ID	UC-01
Goals	Реєстрація нового користувача в системі
Actors	Гість (неzareєстрований користувач)
Trigger	Користувач бажає zareєструватися
Pre-conditions	-
Flow of Events	1. Користувач відвідує сторінку реєстрації; 2. Користувач заповнює поля: ім'я, пошта, пароль, повторний пароль; 3. Користувач натискає кнопку "Зареєструватися"; 4. Система перевіряє введені дані та при успішній реєстрації переводить користувача на головну сторінку
Extension	Якщо користувач ввів некоректні дані, то повідомляємо про помилковий від певного поля.
Post-Condition	Створення сторінки користувача, перехід на головну сторінку

Таблиця 1.3 - Варіант використання UC-2

Use Case Name	Вхід
Use Case ID	UC-02
Goals	Здійснити вхід в профіль
Actors	Користувач (zareєстрований)
Trigger	Користувач хоче увійти в систему як аутентифікований
Pre-conditions	Користувач zareєстрований у системі
Flow of Events	Користувач переходить на сторінку входу. На сторінці він бачить форму з полями "Email" та "Password", а також кнопку "Login". Користувач вводить свою електронну пошту та пароль відповідно. Після введення даних, користувач натискає кнопку "Login". Система перевіряє введені дані. Якщо введені дані

Продовження таблиці 1.3

	коректні, користувач автоматично переходить на головну сторінку застосунку. Якщо введені дані некоректні, система повідомляє користувача про помилку входу. Користувач виправляє помилки та повторює процес входу.
Extension	Якщо користувач ввів некоректні дані, то повідомляємо про помилковий від певного поля.
Post-Condition	Користувач авторизований і переходить на головну сторінку

Таблиця 1.4 - Варіант використання UC-3

Use Case Name	Пошук фільму за назвою
Use Case ID	UC-03
Goals	Здійснити пошук фільму за назвою
Actors	Користувач (любий)
Trigger	Користувач вводить назву фільму у поле пошуку, після чого нажимає «Enter» чи на кнопку пошуку
Pre-conditions	-
Flow of Events	Користувач вводить назву фільму у поле пошуку. Користувач натискає кнопку або ентер для початку пошуку. Система шукає фільми з введеною назвою. Користувача перенаправляють на сторінку фільмів з підборкою фільмів з введеною назвою.
Extension	Якщо користувач ввів пусте поле, то нічого не відбувається
Post-Condition	Користувач переглядає сторінку фільмів з результатами пошуку.

Таблиця 1.5 - Варіант використання UC-4

Use Case Name	Пошук фільму через фільтрацію
Use Case ID	UC-04
Goals	Здійснити пошук фільму через фільтрацію
Actors	Користувач (любий)
Trigger	Користувач обирає опції фільтрації (жанр та рік) на сторінці фільмів



Продовження таблиці 1.5

Pre-conditions	Користувач перебуває на сторінці фільмів
Flow of Events	Користувач обирає жанр та рік з випадального списку опцій на сторінці фільмів. Користувач натискає кнопку для застосування обраних опцій. Сторінка фільтрує існуючу підборку фільмів, що відповідають обраним критеріям.
Extension	Якщо користувач нічого не обрав, проте запустив пошук, то отримаємо фільми усіх жанрів та років
Post-Condition	Користувач переглядає сторінку фільмів з результатами фільтрації

Таблиця 1.6 - Варіант використання UC-5

Use Case Name	Перегляд фільму
Use Case ID	UC-05
Goals	Здійснити перегляд фільму
Actors	Користувач (любий)
Trigger	Користувач натискає на пуск відеоплеєра для запуску фільму
Pre-conditions	Користувач перебуває на сторінці перегляду фільму
Flow of Events	Користувач натискає на кнопку пуску відеоплеєра для запуску фільма. Користувач має можливість зупиняти фільм, змінювати гучність та розмір екрану перегляду.
Extension	-
Post-Condition	Користувач переглядає фільм з можливістю управління відтворенням.

Таблиця 1.7 - Варіант використання UC-6

Use Case Name	Перегляд інформації про фільм та його відгуків
Use Case ID	UC-06
Goals	Отримати інформацію про фільм
Actors	Користувач (любий)
Trigger	Користувач натискає на постер фільму для переходу на сторінку обраного фільму
Pre-conditions	Користувач перебуває на сторінці фільмів

Продовження таблиці 1.7

Flow of Events	Користувач натискає на постер фільму. Система перенаправляє користувача на сторінку обраного фільму. На сторінці фільму надається інформація: назва фільму, рік, опис, мова, жанри. На сторінці є блок відгуків, кнопки для збереження фільму та переходу до його перегляду.
Extension	-
Post-Condition	Користувач переглядає інформацію про обраний фільм.

Таблиця 1.8 - Варіант використання UC-7

Use Case Name	Додавання та видалення фільму з збережених
Use Case ID	UC-07
Goals	Додати або видалити фільм із збережених
Actors	Користувач (авторизований)
Trigger	Користувач натискає на кнопку для збереження фільму
Pre-conditions	Користувач натискає на кнопку «Like»
Flow of Events	Користувач натискає на кнопку для збереження фільма. Система перевіряє, чи фільм вже збережений у користувача. Якщо фільм не збережений, він додається до збережених. Якщо фільм вже збережений, він видаляється зі списку збережених.
Extension	-
Post-Condition	Статус збереження фільма змінюється відповідно до дії користувача.

Таблиця 1.9 - Варіант використання UC-8

Use Case Name	Перегляд фільмів у збережених
Use Case ID	UC-08
Goals	Перегляд фільмів у збережених
Actors	Користувач (авторизований)
Trigger	Користувач переходить на сторінку збережених фільмів
Pre-conditions	Користувач перебуває на сторінці збережених фільмів

Продовження таблиці 1.9

Flow of Events	Користувач переходить на сторінку збережених фільмів. Система відображає фото до фільма, дату релізу, жанри, назву та мову у вигляді таблиці. Користувач може натискати на фото чи назву фільма та переходить на його сторінку.
Extension	-
Post-Condition	Користувач переглядає фільми у своїх збережених.

Таблиця 1.10 - Варіант використання UC-9

Use Case Name	Написання відгуку до фільму
Use Case ID	UC-09
Goals	Авторизований користувач хоче написати відгук та поставити оцінку фільму
Actors	Авторизований користувач
Trigger	Користувач перебуває на сторінці фільму у секторі коментарів
Pre-conditions	Користувач увійшов у свій обліковий запис та перебуває на сторінці фільму
Flow of Events	Система відображає форму для написання відгуку та встановлення оцінки. Користувач вводить текст відгуку та встановлює оцінку. Користувач натискає кнопку "Опублікувати". Система перевіряє, чи всі обов'язкові поля заповнені. Якщо усі поля заповнені, відгук додається до списку опублікованих відгуків.
Extension	Якщо не всі поля заповнені, користувач інформується про необхідність заповнення усіх полів.
Post-Condition	Відгук доданий до списку опублікованих.

Таблиця 1.11 - Варіант використання UC-10

Use Case Name	Перегляд та редагування особистих даних
Use Case ID	UC-10
Goals	Перегляд та редагування особистих даних користувача

Продовження таблиці 1.11

Actors	Авторизований користувач
Trigger	Користувач переходить на сторінку свого профілю та натискає кнопку
Pre-conditions	Користувач увійшов у свій обліковий запис на сторінку профіля
Flow of Events	Користувач переходить на сторінку свого профілю. Система відображає дані користувача: ім'я, пошта, кількість збережених фільмів, дата реєстрації. Користувач натискає кнопку редагування. Система відображає форму для редагування ім'я та пароля. Користувач вносить зміни та натискає кнопку "Зберегти". Система перевіряє, чи нові дані проходять валідацію. Якщо валідація пройшла успішно, дані зберігаються.
Extension	Якщо нові дані не проходять валідацію, користувач інформується про помилку. Якщо користувач не зробив ніяких змін чи надав нові валідні дані він інформується про успішне збереження.
Post-Condition	Користувач переглядає актуальні особисті дані на сторінці профіля.

Таблиця 1.12 - Варіант використання UC-11

Use Case Name	Видалення акаунта
Use Case ID	UC-11
Goals	Видалення облікового запису
Actors	Авторизований користувач
Trigger	Користувач натискає кнопку видалення акаунта на своїй сторінці профілю
Pre-conditions	Користувач увійшов у свій обліковий запис
Flow of Events	Користувач натискає кнопку видалення акаунта на своїй сторінці профілю. Дані користувача видаляються з бази даних. Користувач автоматично виходить із системи. Користувач перенаправляється на головну сторінку.
Extension	-
Post-Condition	Акаунт користувача видалено, користувач неавторизований.

Таблиця 1.13 - Варіант використання UC-12

Use Case Name	Перегляд списку користувачів та видалення акаунтів
Use Case ID	UC-12
Goals	Перегляд та видалення користувачів
Actors	Адміністратор
Trigger	Адміністратор переходить на сторінку списку користувачів
Pre-conditions	Користувач переходить на сторінку списку користувачів
Flow of Events	Адміністратор переходить на сторінку списку користувачів. Система відображає дані всіх зареєстрованих користувачів: ім'я, пошта, дата реєстрації, роль. Для кожного користувача, який не є адміністратором, адміністратор бачить кнопку для видалення акаунту. Адміністратор натискає кнопку видалення акаунту для обраного користувача. Система підтверджує дію адміністратора та видаляє дані користувача з таблиці та бази даних.
Extension	-
Post-Condition	Користувач переглядає актуальну таблицю зареєстрованих користувачів.

Таблиця 1.14 - Варіант використання UC-13

Use Case Name	Видалення коментарів
Use Case ID	UC-13
Goals	Видалення відгуку до фільму
Actors	Адміністратор
Trigger	Адміністратор натискає кнопку видалення коментаря на сторінці фільму в секторі опублікованих відгуків
Pre-conditions	Адміністратор авторизований, знаходиться на сторінці фільму
Flow of Events	Адміністратор натискає кнопку видалення коментаря на сторінці фільму. Система підтверджує намір користувача або адміністратора видалити коментар. Дані коментаря видаляються з бази даних. Коментар зникає зі списку опублікованих відгуків.
Extension	-
Post-Condition	Коментар видалено з бази даних та не відображається на сторінці фільму

Таблиця 1.15 - Варіант використання UC-14

Use Case Name	Перегляд списку анонсованих фільмів
Use Case ID	UC-14
Goals	Перегляд анонсів
Actors	Адміністратор
Trigger	Адміністратор переходить на сторінку списку фільмів
Pre-conditions	Користувач увійшов у свій обліковий запис з правами адміністратора
Flow of Events	Адміністратор переходить на сторінку списку фільмів. Система відображає анонси актуальних фільмів у вигляді таблиці. Таблиця містить фото фільму, назву, дату релізу, жанри та інші деталі.
Extension	-
Post-Condition	Адміністратор переглядає анонси фільмів

#### 1.4.1 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. На рисунку 1.6 наведено загальну модель вимог, а в таблицях 1.16 – 1.33 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 1.7

ID	Description	Name	Priority	Status
1	Реєстрація користувача	FR_1	1	Approved
2	Вхід користувача	FR_2	1	Approved
3	Перегляд інформації про фільм	FR_3	1	Approved
4	Перегляд відгуків	FR_4	2	Approved
5	Перегляд фільму	FR_5	3	Approved
6	Пошук фільмів по назві	FR_6	2	Approved
7	Фільтрування фільмів за жанром та роком	FR_7	2	Approved
8	Додавання фільма до збережених	FR_8	1	Approved
9	Видалення фільма зі збережених	FR_9	1	Approved
10	Перегляд збережених фільмів	FR_10	1	Approved
11	Перегляд особистих даних	FR_11	1	Approved
12	Редагування особистих даних	FR_12	2	Approved
13	Написання відгуку	FR_13	2	Approved
14	Видалення акаунта	FR_14	3	Approved
15	Видалення відгуків адміністратором	FR_15	4	Approved
16	Перегляд анонсованих фільмів адміністратором	FR_16	4	Approved
17	Перегляд особистих даних зареєстрованих користувачів адміністратором	FR_17	1	Approved
18	Видалення акаунту зареєстрованого користувача	FR_18	1	Approved

Рисунок 1.6 – Модель вимог у загальному вигляді

Таблиця 1.16 – Функціональна вимога FR-1

Назва	Реєстрація користувача
Опис	<p>Система повинна надавати можливість реєстрації неавторизованому користувачеві. Під час реєстрації користувач повинен заповнити обов'язкові поля, такі як ім'я, електронна пошта, пароль та підтвердження паролю.</p> <p>Ім'я має мінімальну довжину 3 символи та може складатися з літер, цифр, нижнього підкреслення та додатково може містити пробіли.</p> <p>Електронна пошта повинна відповідати шаблону «*@*.*», де «*» означає будь-які символи, окрім пробілу та знаку «@» у кількості від одного.</p> <p>Пароль повинен бути більше 8 символів та містити хоча б одну цифру, велику та малу літеру.</p>

Таблиця 1.17 – Функціональна вимога FR-2

Назва	Вхід користувача
Опис	<p>Система повинна надавати можливість не авторизованому користувачеві входити у свій обліковий запис. Користувач повинен ввести свою електронну пошту та пароль, які відповідають його реєстраційним даним. Після введення коректних облікових даних система дозволяє користувачеві отримати доступ до особистого облікового запису та використовувати функціонал додатку.</p>

Таблиця 1.18 – Функціональна вимога FR-3

Назва	Перегляд інформації про фільм
Опис	<p>Система повинна дозволяти любому користувачу переглядати інформацію про конкретний фільм. На сторінці фільму буде відображатися назва фільму, рік випуску, опис, мова, жанри та інші важливі характеристики.</p>

Таблиця 1.19 – Функціональна вимога FR-4

Назва	Перегляд відгуків
Опис	Система повинна надавати можливість любому користувачеві переглядати опубліковані відгуки до конкретного фільму. На сторінці фільму користувач може бачити відгуки інших користувачів, включаючи текстовий опис та оцінку.

Таблиця 1.20 – Функціональна вимога FR-5

Назва	Перегляд фільму
Опис	Система повинна дозволяти користувачеві переглядати фільм. На сторінці фільму буде розміщено кнопку для переходу на сторінку перегляду фільму. На ній буде розміщений відеоплеєр для запуску фільму, а також інші функції, такі як зупинка, регулювання гучності та розмір екрану перегляду.

Таблиця 1.21 – Функціональна вимога FR-6

Назва	Пошук фільмів по назві
Опис	Система дозволяє любому користувачеві здійснювати пошук фільмів за їх назвою з будь-якої сторінки через блок пошуку, де користувач має ввести назву фільму у відповідне поле та отримати результати пошуку, які включають фільми, назви яких містять введену користувачем фразу. Результати пошуку повинні бути відсортовані за релевантністю.

Таблиця 1.22 – Функціональна вимога FR-7

Назва	Фільтрування фільмів за жанром та роком
Опис	Система повинна надавати можливість користувачеві фільтрувати фільми за жанром та роком випуску через обрання певних критеріїв у випадаючих списках. Система повинна відобразити результати фільтрації, які відповідають обраним критеріям.



Таблиця 1.23 – Функціональна вимога FR-8

Назва	Додавання фільму до збережених
Опис	<p>Система повинна дозволяти зареєстрованому користувачеві додавати фільми до свого списку збережених.</p> <p>На сторінці кожного фільму та в додаткових локаціях повинна бути кнопка або інший елемент інтерфейсу, який користувач може натискати для додавання фільму до свого списку.</p> <p>Після натискання цієї кнопки фільм повинен бути доданий до списку збережених користувача.</p>

Таблиця 1.24 – Функціональна вимога FR-9

Назва	Видалення фільму зі збережених
Опис	<p>Система повинна надавати можливість зареєстрованому користувачеві видаляти фільми зі свого списку збережених.</p> <p>На сторінці кожного фільму в списку збережених повинна бути кнопка або інший елемент інтерфейсу, який користувач може натискати для видалення фільму зі свого списку.</p> <p>Після натискання цієї кнопки фільм повинен бути видалений зі списку збережених користувача.</p>

Таблиця 1.25 – Функціональна вимога FR-10

Назва	Перегляд збережених фільмів
Опис	<p>Система повинна надавати можливість зареєстрованому користувачеві переглядати свій список збережених фільмів.</p> <p>Інформація про кожен фільм може включати фото, назву, рік релізу, жанри та інші характеристики у вигляді таблиці.</p> <p>Натискання на об'єкт таблиці має переводити користувача до сторінки обраного фільму.</p>

Таблиця 1.26 – Функціональна вимога FR-11

Назва	Перегляд особистих даних
Опис	Система повинна надавати можливість зареєстрованому користувачеві переглядати свої особисті дані, такі як ім'я, електронна пошта, кількість збережених фільмів, дата реєстрації та інші.

Таблиця 1.27 – Функціональна вимога FR-12

Назва	Редагування особистих даних
Опис	Система повинна надавати можливість авторизованому користувачеві редагувати своє ім'я або пароль. Якщо користувач успішно вносить зміни, система повинна їх зберегти та повідомити користувача про успішну операцію. Валідація даних відповідна до етапу реєстрації.

Таблиця 1.28 – Функціональна вимога FR-13

Назва	Написання відгуку
Опис	Система повинна надавати можливість зареєстрованому юзеру залишати відгуки до фільмів. На сторінці фільму користувач може написати текстовий відгук та встановити оцінку. Заповнення усіх полів є обов'язковим. Відгуки можна опублікувати, щоб вони були відображені на відповідних сторінках фільмів.

Таблиця 1.29 – Функціональна вимога FR-14

Назва	Видалення акаунта
Опис	Система повинна надавати можливість авторизованому користувачеві видаляти свій обліковий запис. На сторінці профілю користувач повинен мати кнопку "Видалити акаунт", при натисканні на яку обліковий запис користувача буде видалений з системи, і він буде переадресований на головну сторінку як неавторизований. Усі особисті дані користувача повинні бути вилучені з бази даних.

Таблиця 1.30 – Функціональна вимога FR-15

Назва	Видалення відгуків адміністратором
Опис	<p>Система повинна надавати можливість адміністраторові видаляти коментарі до фільмів.</p> <p>На сторінці фільму в секторі опублікованих коментарів адміністратор може вибрати конкретний коментар та видалити його.</p> <p>Після видалення, коментар повинен бути видалений із списку опублікованих коментарів та з бази даних.</p>

Таблиця 1.31 – Функціональна вимога FR-16

Назва	Перегляд анонсованих фільмів адміністратором
Опис	<p>Система повинна надавати можливість адміністраторові переглядати список фільмів.</p> <p>На сторінці з анонсами фільмів адміністратор може бачити анонси актуальних фільмів у вигляді таблиці.</p> <p>Кожен елемент списку повинен включати основну інформацію про фільм, таку як назва, жанр, рік випуску, та інші важливі характеристики.</p>

Таблиця 1.32 – Функціональна вимога FR-17

Назва	Перегляд особистих даних зареєстрованих користувачів адміністратором
Опис	Система повинна надавати можливість адміністраторові переглядати список всіх зареєстрованих користувачів разом з їхніми особистими даними, такими як ім'я, електронна пошта, дата реєстрації, роль та інші.

Таблиця 1.33 – Функціональна вимога FR-18

Назва	Видалення акаунту зареєстрованого користувача
Опис	Система повинна надавати можливість адміністраторові видаляти облікові записи зареєстрованих користувачів. В адміністративному розділі системи адміністратор може переглядати список користувачів та обирає конкретного користувача для видалення. Після підтвердження видалення, обліковий запис користувача повинен бути повністю вилучений з системи, а усі його особисті дані повинні бути видалені з бази даних.

	UC_1	UC_2	UC_3	UC_4	UC_5	UC_6	UC_7	UC_8	UC_9	UC_10	UC_11	UC_12	UC_13	UC_14
FR_1	x													
FR_2	x	x												
FR_3						x								
FR_4						x								
FR_5					x									
FR_6			x											
FR_7				x										
FR_8							x							
FR_9							x							
FR_10								x						
FR_11										x				
FR_12										x				
FR_13									x					
FR_14											x			
FR_15													x	
FR_16														x
FR_17												x		
FR_18												x		

Рисунок 1.7 – Матриця трасування вимог

#### 1.4.2 Розроблення нефункціональних вимог

Серед нефункціональних вимог зазначимо наступні:

- Вихідний код програми має бути наведений у репозиторію GitHub;
- Застосунок має бути зручним для використання та інтуїтивним;
- Застосунок має мати адаптивний дизайн для різних пристроїв (комп'ютерна та телефонна версії);
- Застосунок має працювати в браузерах Google Chrome та Opera;

– Розробка ПЗ має бути виконана у WebStorm IDE.

### 1.5 Постановка задачі

Метою курсової роботи є розробка веб-додатку для підтримки діяльності онлайн-кінотеатру Mediatoria. Додаток розрахований на людей, які люблять дивитися фільми онлайн з різних девайсів.

Основним завданням курсової роботи є забезпечення користувачам можливості перегляду фільмів онлайн, а також взаємодії з системою за допомогою функціоналу, який включає в себе формування персонального списку улюблених фільмів, написання коментарів та виставлення рейтингів фільмів, а також ефективну та зручну систему пошуку.

Для реалізації основної функції онлайн-кінотеатру необхідно надати користувачам систему відтворення відео, яка підтримує основні формати відеофайлів і дозволяє переглядати фільми з різних пристроїв, таких як комп'ютери та телефони.

Для залучення користувачів до активної взаємодії з системою було впроваджено функцію рецензування. Кожен користувач може висловлювати свої враження від фільмів, ділитися думками та створювати дискусії навколо конкретних картин.

Однією з найголовніших особливостей є функція збереження фільмів. Система дозволяє користувачам більше взаємодіяти з медіа-контентом, зберігаючи фільми, які їм сподобалися, для подальшого перегляду або написання коментарів.

Крім того, система пропонує інструменти для швидкого пошуку фільмів. Зручна пошукова система дозволяє знаходити кінострічки за назвою або за допомогою фільтрації за різними параметрами, наприклад, за жанром або роком випуску. Це полегшує користувачам пошук контенту, який найбільше їх цікавить, серед широкого асортименту фільмів.

Система враховує різні ролі користувачів. Гості можуть лише переглядати фільми та коментарі до них. Зареєстровані користувачі отримують додаткові

можливості, такі як додавання коментарів, оцінювання фільмів та керування списком збережених фільмів. Адміністратори, крім того, мають повноваження керувати контентом, видаляти коментарі або акаунти зареєстрованих користувачів, а також переглядати дані.

Додаток повинен бути простим у використанні, мати зручний інтерфейс і стабільно працювати. Оскільки в сучасному світі користувачі можуть отримувати доступ до веб-сторінок з різних пристроїв, додаток повинен мати адаптивний дизайн.

### Висновки до розділу

У рамках цього розділу було виконано комплексний аналіз предметної області та визначено напрямки розвитку системи підтримки діяльності онлайн кінотеатру "Mediatoria". Розглянуті аспекти включають загальний огляд області, аналіз існуючих технологій та програмних продуктів, вимоги до розроблюваного ПЗ, а також постановку завдань для подальшої реалізації.

В розділі "Загальні положення" було проведено загальний огляд предметної області, визначено цілі та завдання розробки системи підтримки діяльності онлайн кінотеатру "Mediatoria". Описано основні функціональні можливості застосунку та його призначення для любителів фільмів.

Далі був проведений аналіз використання знань предметної області в ІТ-технологіях, визначено недоліки існуючих рішень у сфері онлайн кінотеатрів та вказано напрямки для подальших покращень.

Розділ "Аналіз існуючих технологій та успішних ІТ-проектів" включає в себе аналіз алгоритмічних та технічних рішень, що використовуються у подібних проектах. Розділ дозволив визначити оптимальні підходи до реалізації веб-застосунку, такі як React, Firebase Auth, Firestore та інші. Обрані рішення дозволяють вирішити задачі пов'язані із збереженням даних, аутентифікацією, визначення необхідних бібліотек та інші.

Для реалізації застосунку Mediatoria необхідно було визначити необхідний функціонал для користувачів в порівнянні з аналогічними сервісами підтримки

кінотеатрів. Для прикладу були обрані веб-сервіси Netflix та IMDB. Основна увага була приділена їх порівняльному аналізу, були визначені недоліки та переваги для кожного з них. Саме це дало змогу виокремити проблеми, які "Mediatoria" спрямована вирішити.

Наступним кроком стала розробка функціональних та нефункціональних вимог, які визначають необхідний функціонал застосунку. Був створений перелік функцій для різних типів користувачів, детально описано кожен елемент вимог та його призначення. Результати було продемонстровано у вигляді таблиць моделі вимог у загальному вигляді та її детальному опису, що в кінці кінців, дозволило сформулювати матрицю трасування вимог.

На основі розроблених функціональних та нефункціональних вимог була сформульована постановка задачі. Вона охоплює узагальнений опис функціоналу та системи для різних категорій користувачів.

У цілому, проведений аналіз та постановка завдань дозволяють чітко визначити шлях розробки системи підтримки діяльності онлайн кінотеатру "Mediatoria" для забезпечення максимальної користувацької зручності та функціональності.

## 2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 2.1 Моделювання та аналіз програмного забезпечення

Для опису бізнес процесу взаємодії користувача із сторінкою фільму використовується BPMN модель (рисунок 2.1).

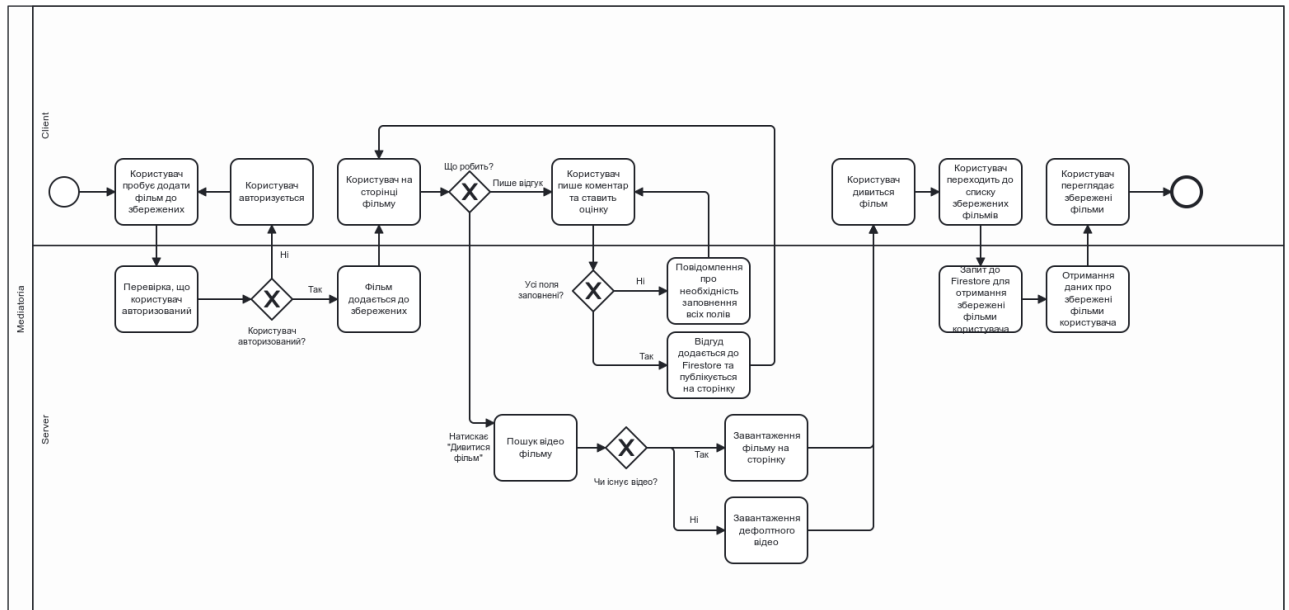


Рисунок 2.1 – Бізнес процес взаємодії користувача із сторінкою фільму

Опис послідовної взаємодії користувача із сторінкою фільму:

- 1) користувач пробує додати фільм до збережених;
- 2) сервер перевіряє чи користувач авторизований;
- 3) якщо користувач неавторизований, він переходить на сторінку «Log In», де авторизується, п.1;
- 4) якщо користувач авторизований, сервер додає фільм до збережених;
- 5) користувач знаходиться на сторінці фільму;
- 6) якщо користувач вирішив писати відгук він пише коментар та ставить оцінку;
- 7) сервер перевіряє чи усі поля заповнені;



- 8) якщо не усі поля заповнені сервер повідомляє клієнту про це, далі п.6;
- 9) якщо усі поля заповнені, сервер додає відгук до Firestore та публікує його на сторінці, далі п.5;
- 10) якщо користувач вирішив натиснути на кнопку «Дивитися фільм», сервер виконує пошук відео фільму;
- 11) якщо відео не існує сервер передає відео за замовчуванням, п.13;
- 12) якщо відео існує сервер передає його;
- 13) користувач переглядає передане відео від сервера на сторінці перегляду;
- 14) користувач переходить до списку збережених фільмів;
- 15) сервер виконує запит до Firestore для отримання даних збережених фільмів користувача;
- 16) сервер отримує дані збережений фільмів користувача;
- 17) користувач переглядає збережені фільми;

Для опису бізнес процесу взаємодії адміністратора із даними системи використовується BPMN модель (рисунок 2.2).

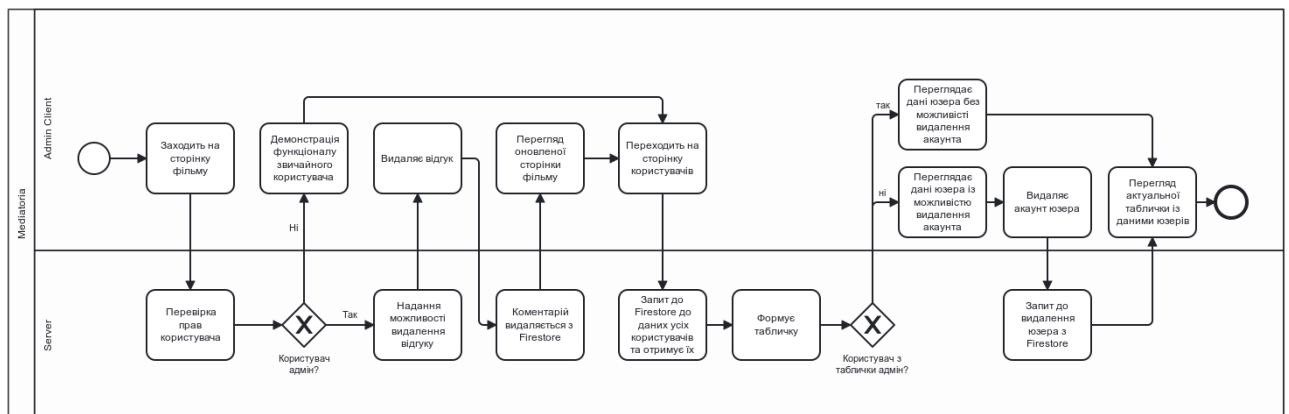


Рисунок 2.2 – Бізнес процес взаємодії адміністратора із даними системи

Опис послідовної взаємодії адміністратора із даними системи:

- 1) користувач заходить на сторінку фільму;
- 2) сервер перевіряє роль користувача;

- 3) якщо користувач не адміністратор, йому демонструється функціонал звичайного юзера, далі п.8;
- 4) якщо користувач адміністратор, йому надається функціонал видалення коментарів;
- 5) користувач видаляє відгук;
- 6) сервер видаляє відгук з Firestore;
- 7) користувач переглядає оновлену сторінку фільму;
- 8) користувач переходить на сторінку користувачів;
- 9) сервер виконує запит до Firestore до даних усіх користувачів та отримує їх;
- 10) сервер формує таблицю користувачів;
- 11) якщо користувач з таблиці є адміном, його акаунт неможливо видалити, далі п.15;
- 12) якщо користувач з таблиці не є адміном, його акаунт можливо видалити;
- 13) користувач видаляє акаунт іншого користувача;
- 14) сервер виконує запит до видалення юзера з Firestore;
- 15) адміністратор переглядає актуальну таблицю із даними користувачів.

## 2.2 Архітектура програмного забезпечення

Для створення веб-застосунку обрали клієнт-серверну архітектуру, яка визначає спосіб взаємодії між фронтендом та бекендом. Додаток "Mediatoria" використовує клієнт-серверну архітектуру, де клієнтська сторона загалом представлена React-компонентами, а серверна — інфраструктурою Firebase, що включає в себе автентифікацію через Firebase Auth та роботу з даними за допомогою Firestore (Рисунок 2.3). Перейдемо до більшого та конкретнішого опису обох частин цієї архітектури.

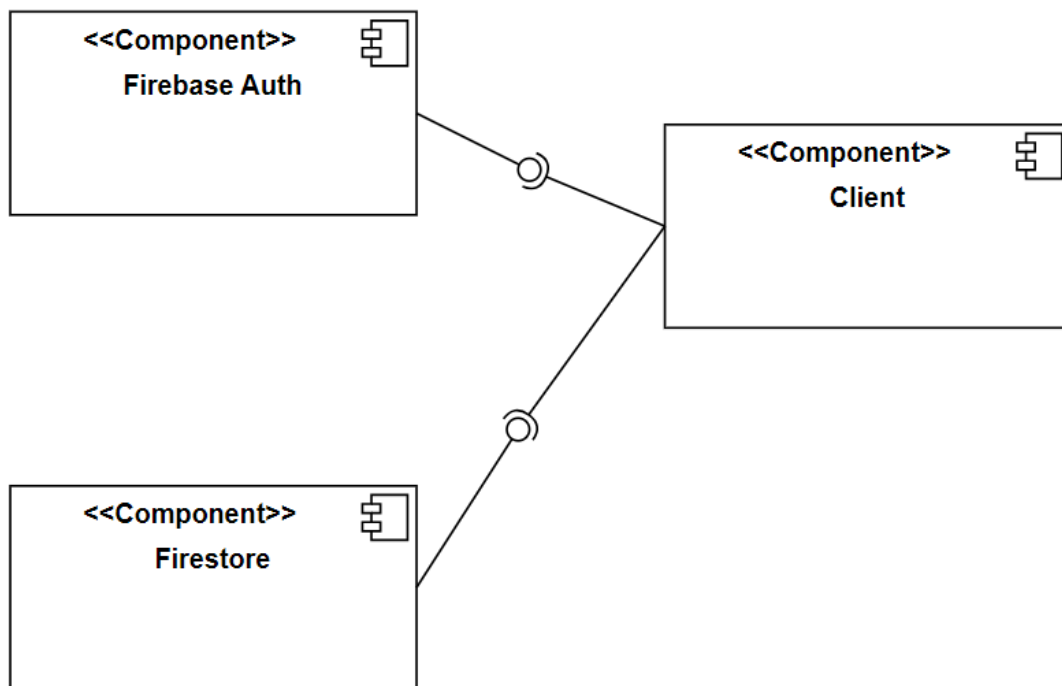


Рисунок 2.3 – Схема клієнт-серверної архітектури застосунку

Клієнтська частина додатку "Mediatoria" забезпечує інтерфейс взаємодії користувача з системою, визначаючи його вигляд, поведінку та функціонал. Вона побудована на основі бібліотеки React, який використовує різноманітні інструменти для оптимізації та полегшення розробки.

Основним будівельним блоком клієнтської частини є React-компоненти. Кожна сторінка та функціональна частина додатку реалізована як окремий компонент, що спрощує розробку. Використання контекстів надасть ефективний обмін даними між компонентами, а react-router-dom дозволить створювати односторінкові додатки з ефективною навігацією за допомогою компоненти Routes. Вона визначає доступні маршрути, а Route визначає, який компонент має бути відображений для кожного маршруту. Це створить послідовність візуальних сторінок для користувача.

Клієнтська частина також взаємодіє з сервером через HTTP-запити для отримання фільмів через API TMDb, які відображаються на відповідних сторінках, забезпечуючи користувачам доступ до актуальної інформації.

Щодо серверної частини, основним її фреймворком є Firebase, що забезпечує ряд інструментів для розробки та взаємодії з сервером. Зокрема, використання Auth, Firestore дозволяє ефективно управляти аутентифікацією, зберіганням даних.

Загалом, Firebase Auth надає швидкий та безпечний спосіб управління ідентифікацією користувачів, а БД Firestore використовується для зберігання та організації різноманітної інформації, такої як дані про користувачів, коментарі до фільмів та інші елементи. Спрощена структура документів та колекцій дозволяє ефективно обробляти дані в реальному часі.

### 2.3 Конструювання програмного забезпечення

Оскільки маємо постійно перевіряти дані пов'язані із користувачем через запити гарною ідеєю стане використання контексту. В даному випадку, для роботи із даними користувача визначимо AuthContext. Саме такий підхід полегшить відображення персоналізованої інформації в різних частинах додатку без необхідності передачі даних через пропси. Також в нас буде можливість розширення або зміни механізмів у майбутньому. Це забезпечує гнучкість програмному забезпеченню.

Для забезпечення функціоналу реєстрації своїх облікових записів реалізуємо функцію `signup` у `AuthContext`, яка використовує `createUserWithEmailAndPassword` з `Firebase Auth API`. У цьому ж методі необхідно зберегти нові дані від користувача після успішної реєстрації. Отже, на цьому етапі потребуємо створення запису в базі даних `Firestore` за допомогою метода `setDoc`.

Таким самим чином, маємо надати можливість входу в обліковий запис. Функція `login` буде використовувати `signInWithEmailAndPassword` з `Firebase Auth API` для аутентифікації користувача. Після вдалого входу користувач отримує доступ до ресурсів застосунку.

Механізм виходу з системи реалізуємо через метод `logout`, використовуючи `signOut` з `Firebase Auth` для безпечного завершення сеансу роботи в системі.

У даному контексті доцільним буде формування функцій визначення ролей користувачів, перевірки унікальності email, зміни паролю, імені через подібну взаємодію із `Firestore`, що зазначалися вище.

Оскільки також потребуємо постійної взаємодії із інформацією про фільми, доречним буде створення ще одного контексту `MovieContext`, що дозволить логічно групувати та надавати доступ до методів, які стосуються роботи з фільмами, забезпечуючи централізовану точку керування для цих операцій. Він буде взаємодіяти з `TMDB` та `Firestore`.

Взаємодія додатку з `TMDB` здійснюється за допомогою різних запитів до `TMDB API` для отримання інформації про фільми та жанри. Для утримання різних типів запитів можна сформувати файл зі статичними запитами та методами, що сформулюють працюючий запит за вхідними параметрами. Це стане у нагоді, наприклад, якщо користувач захочеті застосувати пошук за назвою. Тоді ми зможемо використати метод по типу:

```
requestTitle: (title) => `https://api.themoviedb.org/3/search/movie?api_key
    = ${API_KEY}&query = ${title}`.
```

Якщо ж необхідно зробити фільтрацію за роком та жанром, то у метода буде наступний вигляд:

```
requestMovies: (year, genre) => {
  let request =
`https://api.themoviedb.org/3/discover/movie?api_key=${API_KEY}&sort_by=popu
larity.desc`
  if (year !== "") request += `&primary_release_year=${year}`
  if (genre !== "") request += `&with_genres=${genre}`
  return request
}
```

Отримавши певне представлення роботи із TMDb, можемо визначити не тільки функціонал MovieContext, але й логіку роботи його методів.

Зручним у використанні буде метод, що дозволить отримувати фільми за запитом. Важливо зазначити, що один запит за замовчуванням повертає лише 20 одиниць контенту. Оскільки ми потребуємо усі варіанти до 1000 (умовний максимум), створимо асинхронний метод `GetMoviesByRequest` для вирішення цієї проблеми.

Також потребуємо у контексті MovieContext метод для отримання переліку жанрів та їх ID. Це є важливим, оскільки при надходженні фільмів їх жанри представлені у вигляді цифри і не несуть інформаційної цінності. Шляхом співставлення ID жанрів фільму до усіх у методі `GetMovieGenres` вирішуємо цю проблему. Сам метод має використовувати `axios` для виконання GET запиту до API TMDb.

До додаткового функціоналу MovieContext можемо також віднести роботу зі збереженими фільмами. Необхідно визначити методи для додавання, видалення та перевірки на збереженість.

Для цього, в свою чергу, треба визначити логіку для отримання переліку збережених фільмів з Firestore, яка має включати перевірку на авторизованість користувача, метод якого має бути реалізований у контексті AuthContext. За допомогою `getDoc` зможемо отримати документ користувача із Firestore за його електронною поштою.

Для збереження, видалення та перевірки чи є фільм у збережених спочатку маємо визначити відповідні назви методів у MovieContext – `SaveToFavorites`, `RemoveFromFavorites` та `IsInFavorites`. Так само маємо перевіряти чи користувач увійшов у систему чи ні. Далі, для взаємодії із документом користувача за його електронною поштою використовуємо метод `updateDoc`. Потім маємо різну логіку:

- в разі збереження маємо виконати `arrayUnion` для додавання нового об'єкта фільму до масиву `favoriteMovies`;

- в разі видалення об'єкта фільму з масиву `favoriteMovies` маємо застосувати `arrayRemove`;
- в разі перевірки на те, чи є фільм у збережених, використовуємо метод `some()`, щоб повернути булеве значення.

Подібним чином маємо визначити ще один контекст для утримання функціоналу для взаємодії із відгуками – `CommentContext`. Він має таку саму логіку дій відповідно до методів пов'язаних зі збереженими фільмами. Даний контекст має реалізовувати додавання інформації про відгуки до БД, видаляти їх та надавати.

Перерахувавши усю взаємодію клієнтської сторони із серверною, маємо змогу визначити кінцеву форму утримання даних про клієнтів, коментарі та фільми.

Колекція «users» має наступну вкладеність:

- Документ користувача:
  - 1) Поле `name`: ім'я користувача;
  - 2) Поле `email`: електронна пошта користувача, яка є унікальним ідентифікатором;
  - 3) Поле `role`: роль користувача (наприклад, "user" або "admin");
  - 4) Поле `createdAt`: дата та час створення облікового запису;
  - 5) Поле `updatedAt`: дата та час останнього оновлення даних користувача;
  - 6) Поле `favoriteMovies` – масив обраних фільмів:
    - a. Поле `id`: унікальний ідентифікатор фільму;
    - b. Поле `title`: назва фільму.

Коментарі будуть утримуватися в колекції «comments», що має наступну вкладеність:

- Документ фільму:
  - 1) Поле `movieId`: унікальний ідентифікатор фільму, пов'язаний з коментарями;
  - 2) Поле `comments`: масив коментарів;

- a. Поле user: пошта користувача-автора;
- b. Поле rating: оцінка фільму користувачем;
- c. Поле comment: текст відгуку.

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 2.22.

Таблиця 2.1 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	@headlessui/react	Бібліотека для створення доступних та гнучких інтерфейсів, наприклад для реалізації фільтра через Transition, Listbox.
2	@heroicons/react	Набір високоякісних SVG-ікон для використання в інтерфейсі.
3	@tailwindcss/line-clamp	Розширення для Tailwind CSS для зручного використання CSS-властивості line-clamp.
4	aos	Бібліотека анімацій для відслідковування прокрутки сторінки.
5	axios	Бібліотека для виконання HTTP-запитів.
6	dotenv	Забезпечує завантаження змінних середовища з файлу .env для безпечного збереження Firebase API.
7	firebase	Клієнтська бібліотека Firebase для роботи з послугами Firebase у веб-додатках.
8	react	Бібліотека для побудови інтерфейсу користувача.
9	react-dom	Додатковий пакет для роботи з DOM в React-додатках через метод createRoot.
10	react-dropzone	Компонент React для завантаження файлів методом "перетягни і відпусти".



Продовження таблиці 2.1

11	react-icons	Колекція іконок React для різноманітних потреб.
12	react-router-dom	Бібліотека для навігації між сторінками React-додатка.
13	swiper	Сучасна бібліотека для створення каруселей та слайдерів.
14	eslint	Інструмент для виявлення та автоматичної корекції помилок в коді.
15	tailwindcss	Утиліта для швидкого розробки адаптивних інтерфейсів з використанням класів.

Слід проаналізувати та визначити перелік системних вимог.

Програмне забезпечення повинно працювати під управлінням операційної систем Windows.

Мінімальна конфігурація технічних засобів:

- тип процесору: Intel Core i5;
- об'єм ОЗП: 4 Гб;
- підключення до мережі Інтернет зі швидкістю від 20 мегабіт;

Рекомендована конфігурація технічних засобів:

- тип процесору: Intel Core i7;
- об'єм ОЗП: 8 Гб;
- підключення до мережі Інтернет зі швидкістю від 100 мегабіт;

Обрані вимоги зумовлені мінімальними вимогами React версії 18.2.0, сучасних браузерів та операційної системи Windows. В свою чергу вони забезпечують оптимальну продуктивність, безпеку та зручність використання застосунку для максимальної кількості користувачів у різних умовах роботи.

## 2.4 Аналіз безпеки даних

Оскільки для взаємодії із TMDb маємо робити запити з використанням особистих API, доречним буде використання бібліотеки `dotenv` для захисту та управління конфіденційних змінних середовища. Сформуємо файл `.env`, де будемо зберігати ключі API Firebase, які не повинні викладатися у відкритий доступ.

Firebase Authentication надає ефективні засоби безпеки даних, що пов'язані із автентифікацією. Вони пересилаються зашифрованими по захищеному каналу. Це запобігає можливості перехоплення та зловживання конфіденційною інформацією. Firebase використовує промисловий стандарт шифрування TLS для захисту даних під час їх передачі між клієнтом та сервером.

Firebase Firestore забезпечує масштабовану та безпечну БД, доступ до якої Firestore контролюється правилами безпеки Firebase. Кожен запит на доступ до бази даних перевіряється відповідними правилами, що забезпечує вищий рівень безпеки.

### Висновки до розділу

В ході моделювання проекту були створені та детально розглянуті BPMN моделі, що визначають взаємодії користувача із сторінкою фільму та адміністратора із даними системи. Ці моделі надали чіткий уявлення про процеси та послідовності подій, що відбуваються під час користування системою.

У розділі "Архітектура програмного забезпечення" була акцентована увага на клієнт-серверній архітектурі. Детально висвітлено взаємодію клієнта з Firebase Auth та Firestore, а також представлено діаграму компонентів, яка ілюструє взаємозв'язки між компонентами системи. Описано їх взаємодію з точки зору React-компонентів, а також зазначено основні аспекти їх використання.

У наступному розділі "Конструювання програмного забезпечення" були детально розглянуті контексти `MovieContext`, `AuthContext` та `CommentsContext`. Визначено їх взаємодію з Firebase Auth, Firestore та The Movie Database (TMDb).

Надано опис колекцій Firestore із повною структурою даних. Розглянуто перелік використаних утиліт та здійснено аналіз системних вимог, що дозволяє отримати повну картину щодо вимог до середовища виконання.

Завершує розділ аналіз проекту зі стороною безпеки даних, яка є важливою складовою будь-якого програмного забезпечення. Тут можна визначити надійність зі сторони постачальника послуг.

В результаті всіх цих кроків визначено та детально описано архітектуру, контексти, взаємодії та засоби забезпечення безпеки для програмного забезпечення, що дозволяє забезпечити ефективну та безпечну роботу системи у повсякденному використанні.

### 3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

#### 3.1 Аналіз якості ПЗ

Процес розробки програмного забезпечення потребує не тільки створення працюючого продукту, але й має забезпечити його якість, ефективність та безпеку. Отже, маємо провести детальний аналіз якості програмного забезпечення веб-застосунку, зокрема через використання SonarCloud для перевірки коду.

За використанням SonarCloud, ми отримали наступні статистичні дані зображені на рисунку 3.1.



Рисунок 3.1 – Об’єм коду по категоріям

Загалом, код нашого проекту складає 3,189 рядків, з яких 2,9 тисяч є кодом JavaScript та 257 — CSS. Це значення є показником обсягу роботи, над якою необхідно провести аналіз якості.

Щодо структури програми у статистичному вигляді маємо перелік зображений на рисунку 3.2.

Size	▼
New Lines	0
Lines of Code	3,137
Lines	3,593
Statements	562
Functions	220
Classes	0
Files	64
Comment Lines	134
Comments (%)	4.1%

Рисунок 3.2 – Статистика по коду

Отже, визначивши об'єм та структуру програми, перейдемо до аналізу результатів складності коду зображених на рисунку 3.2.

Complexity ?	▼
Cyclomatic Complexity	348
Cognitive Complexity	190

Рисунок 3.3 – Результати щодо складності коду застосунка

Значення Cyclomatic Complexity та Cognitive Complexity свідчать про те, що код досить складний. Однак це може бути виправлено шляхом подальшого рефакторингу та оптимізації. Загалом найбільшу складність у програму вносить

використання контекстів, а саме методів визначених у них. Також маємо багато асинхронних методів, використання яких ускладнює програму.

Маємо оцінку Maintainability Rating – A, що вказує на те, наскільки легко можна підтримувати наш код. Рейтинг "A" свідчить про високий рівень підтримки кодової бази, що є дуже позитивним показником, оскільки A є найбільшою оцінкою. В свою чергу F – найгірша.

Маємо наступні показники якості безпеки:

- Security Review Rating on New Code: A
- Security Rating on New Code: A

Рейтинги безпеки "A" вказують на те, що код добре захищений від потенційних загроз і вразливостей.

Маємо показник надійності Reliability Rating on New Code: A, що вказує на високий рівень надійності коду. Це означає, що програма вміє ефективно виконувати свої функції та має низький ризик виникнення помилок чи аварій.

### 3.2 Опис процесів тестування

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 3.1 – 3.10

Таблиця 3.1 – Тест 1.1

Тест	Реєстрація користувача
Модуль	Авторизація
Номер тесту	1.1
Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні данні	Ім'я, електронна пошта, пароль, підтвердження паролю
Опис проведення тесту	У відповідні поля вводяться: ім'я, що повинно мати мінімальну довжину 3 символи та може складатися з літер, цифр, нижнього підкреслення та додатково може містити пробіли; електронна пошта повинна відповідати шаблону «*@*.*», де «*» означає будь-які символи, окрім

Продовження таблиці 3.1

	пробілу та знаку «@» у кількості від одного; пароль, що повинен бути більше 8 символів та містити хоча б одну цифру, велику та малу літеру.
Очікуваний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на головну сторінку як авторизований.
Фактичний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на головну сторінку як авторизований.

Таблиця 3.2 – Тест 1.2

Тест	Вхід користувача
Модуль	Авторизація
Номер тесту	1.2
Початковий стан системи	Користувач знаходиться на сторінці входу.
Вхідні данні	Пошта та пароль
Опис проведення тесту	У відповідні поля вводяться: електрона пошта та пароль, що були зазначені користувачем при успішній реєстрації. Далі користувач натискає кнопку для здійснення входу.
Очікуваний результат	Користувач успішно входить у систему як авторизований та перенаправляється на головну сторінку.
Фактичний результат	Користувач успішно входить у систему як авторизований та перенаправляється на головну сторінку.

Таблиця 3.3 – Тест 1.3

Тест	Перегляд таблиці збережених фільмів
Модуль	Збережені фільми

Продовження таблиці 3.2

Номер тесту	1.3
Початковий стан системи	Користувач авторизований та перебуває на сторінці "Favorites".
Вхідні данні	-
Опис проведення тесту	Користувач переглядає перелік інформації про усі додані фільми до збережених. Інформація має включати: назву фільму, постер, жанри, дата релізу. Користувач натискає на постер чи назву фільма і переходить на його сторінку.
Очікуваний результат	Відображення таблиці зі збереженими фільмами.
Фактичний результат	Відображення таблиці зі збереженими фільмами.

Таблиця 3.4 – Тест 1.4

Тест	Додавання фільму до збережених
Модуль	Збережені фільми
Номер тесту	1.4
Початковий стан системи	Користувач авторизований та знаходиться на сторінці фільму, який не доданий у збережені.
Вхідні данні	-
Опис проведення тесту	Користувач натискає на іконку «Like» для додавання фільму до збережених. Іконка змінює свій вигляд, інформуючи про успішне додавання контенту до збережених.
Очікуваний результат	Іконка «Like» змінила свій вигляд, сам фільм доданий до списку збережених фільмів.
Фактичний результат	Іконка «Like» змінила свій вигляд, сам фільм доданий до списку збережених фільмів.



Таблиця 3.5 – Тест 1.5

Тест	Додавання відгуку до фільму
Модуль	Відгуки
Номер тесту	1.5
Початковий стан системи	Користувач авторизований та перебуває на сторінці фільму у розділі коментарів.
Вхідні данні	Текст відгуку, оцінка фільму
Опис проведення тесту	Користувач вводить текст відгуку та обирає оцінку до фільму. Далі він натискає кнопку для опублікування відгуку.
Очікуваний результат	Відгук додається до переліку опублікованих, що демонструє ім'я автора, текст відгуку та оцінку.
Фактичний результат	Відгук додається до переліку опублікованих, що демонструє ім'я автора, текст відгуку та оцінку.

Таблиця 3.6 – Тест 1.6

Тест	Видалення відгуку до фільму
Модуль	Відгуки
Номер тесту	1.6
Початковий стан системи	Користувач є авторизованим адміністратором, перебуває на сторінці фільму із опублікованими відгуками.
Вхідні данні	-
Опис проведення тесту	Користувач обирає коментар, який хоче видалити з опублікованих. Він натискає іконку для видалення.
Очікуваний результат	Відгук користувача зникає з опублікованих коментарів.
Фактичний результат	Відгук користувача зникає з опублікованих коментарів.

Таблиця 3.7 – Тест 1.7

Тест	Пошук фільму за назвою
Модуль	Пошук фільмів
Номер тесту	1.7
Початковий стан системи	Користувач перебуває у веб-застосунку у верхній частині сторінки.
Вхідні данні	Поле назви фільму
Опис проведення тесту	Користувач вводить назву фільму зверху сторінки та натискає на кнопку початку пошуку чи «Enter».
Очікуваний результат	Користувач переходить на сторінку пошуку фільмів, де бачить фільми, що мають у своїй назві вхідний текст.
Фактичний результат	Користувач переходить на сторінку пошуку фільмів, де бачить фільми, що мають у своїй назві вхідний текст.

Таблиця 3.8 – Тест 1.8

Тест	Фільтрація фільмів
Модуль	Пошук фільмів
Номер тесту	1.8
Початковий стан системи	Користувач перебуває на сторінці фільмів.
Вхідні данні	Жанр та рік релізу фільму.
Опис проведення тесту	Користувач у верхній частині сторінки обирає з випадального списку опції жанру та року релізу фільмів. Далі він натискає кнопку «Пошук»
Очікуваний результат	Користувач переглядає відфільтровані фільми на сторінці за вказаними критеріями.
Фактичний результат	Користувач переглядає відфільтровані фільми на сторінці за вказаними критеріями.

Таблиця 3.9 – Тест 1.9

Тест	Видалення акаунту
Модуль	Користувачі
Номер тесту	1.9
Початковий стан системи	Користувач авторизований та перебуває на сторінці налаштувань акаунту.
Вхідні данні	-
Опис проведення тесту	Користувач натискає кнопку «Видалення акаунту».
Очікуваний результат	Користувач стає неавторизованим, дані видаляються, потрапляючи на головну сторінку. При спробі зайти у систему за старими даними, користувач не може увійти.
Фактичний результат	Користувач стає неавторизованим, дані видаляються, потрапляючи на головну сторінку. При спробі зайти у систему за старими даними, користувач не може увійти.

Таблиця 3.10 – Тест 1.10

Тест	Адміністративне видалення акаунту
Модуль	Користувачі
Номер тесту	1.10
Початковий стан системи	Користувач авторизований та є адміністратором, перебуває на сторінці списку усіх користувачів.
Вхідні данні	-
Опис проведення тесту	Користувач переглядає актуальну інформацію даних усіх зареєстрованих користувачів, обирає акаунт юзера (не адміністратора), що хоче видалити. Далі він натискає на іконку для видалення обраного користувача з системи.

### Продовження таблиці 3.10

Очікуваний результат	Поле користувача зникає з таблиці усіх зареєстрованих користувачів. При спробі видаленого користувача зайти за старими вхідними даними авторизацію зазнає невдачі.
Фактичний результат	Поле користувача зникає з таблиці усіх зареєстрованих користувачів. При спробі видаленого користувача зайти за старими вхідними даними авторизацію зазнає невдачі.

### 3.3 Опис контрольного прикладу

Продемонструємо основний функціонал застосунку. Почнемо з вигляду головної сторінки на рисунках 3.4-3.5, що демонструє актуальні фільми через банер та колекції фільмів. Для різних девайсів визначена відповідний варіант навігації. В широкоекраних – верх та низ сторінки, в малих – статична панель.

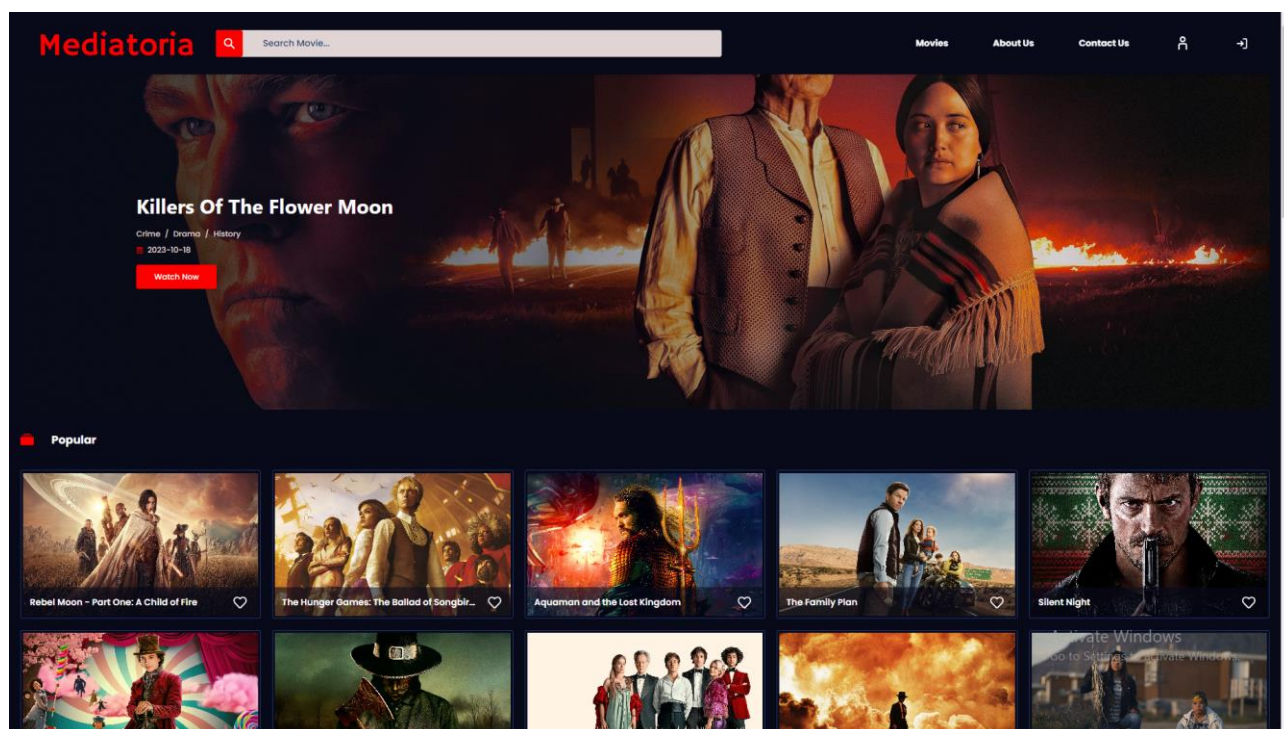


Рисунок 3.4 – Комп'ютерна версія головної сторінки

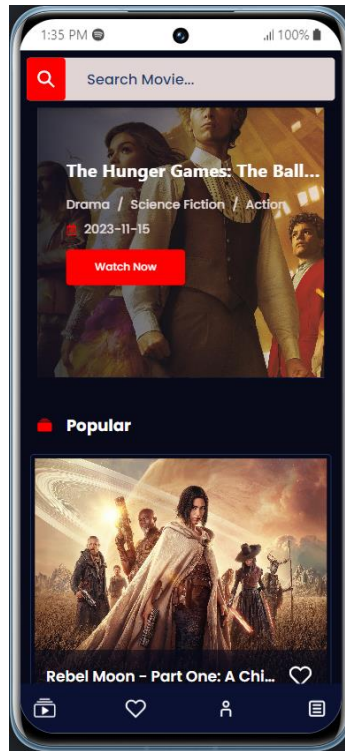


Рисунок 3.5 – Телефона версія головної сторінки

Також, натиснувши вкладку пошуку фільмів переходимо на її сторінку, що продемонстрована на рисунках 3.6-7.

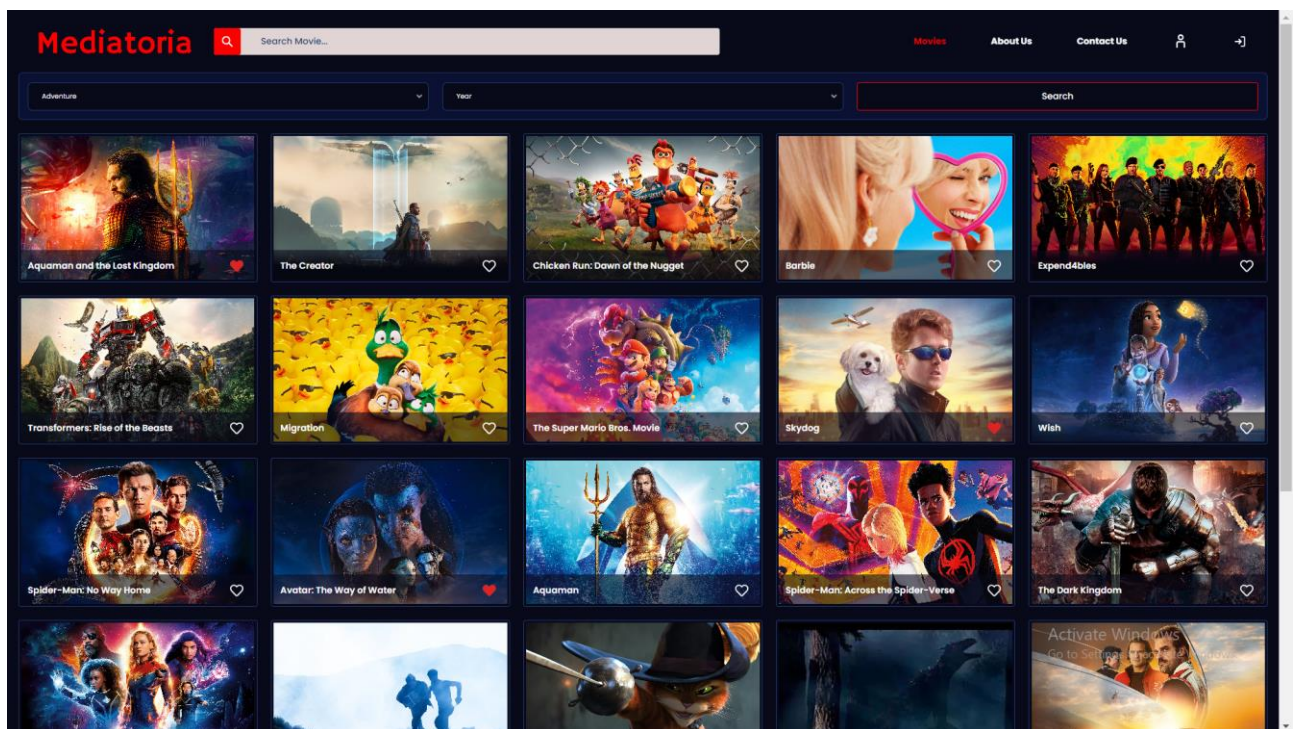


Рисунок 3.6 – Комп'ютерна версія сторінки пошуку фільмів



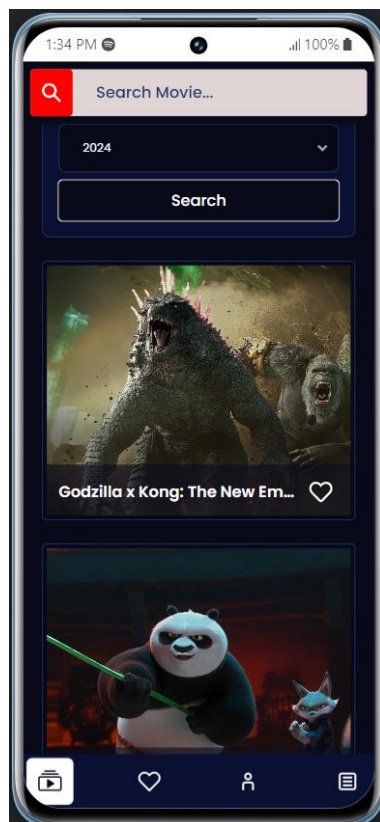


Рисунок 3.7 – Телефона версія сторінки пошуку фільмів

В разі, якщо користувач здійснює пошук по назві він переходить на цю ж сторінку з результатом його пошуку (Рисунки 3.8-9).

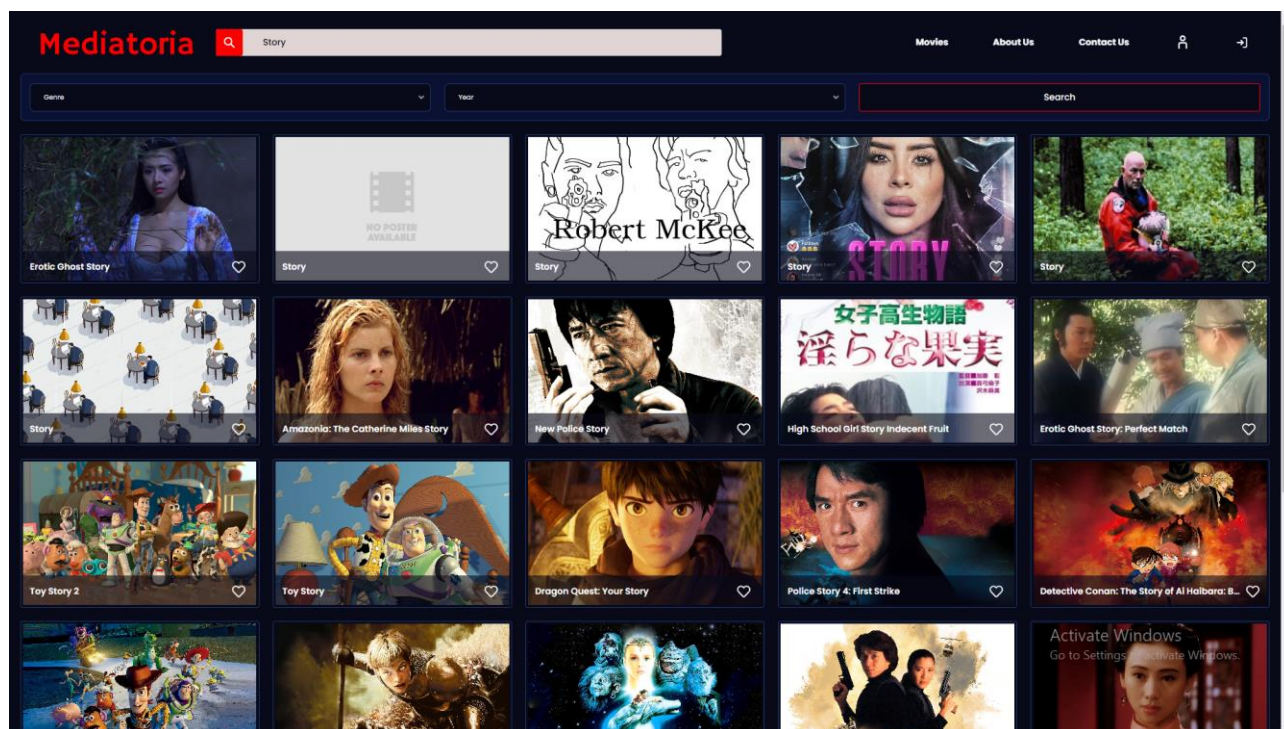


Рисунок 3.8 – Комп'ютерна версія сторінки пошуку фільмів з результатом пошуку за назвою «Story»

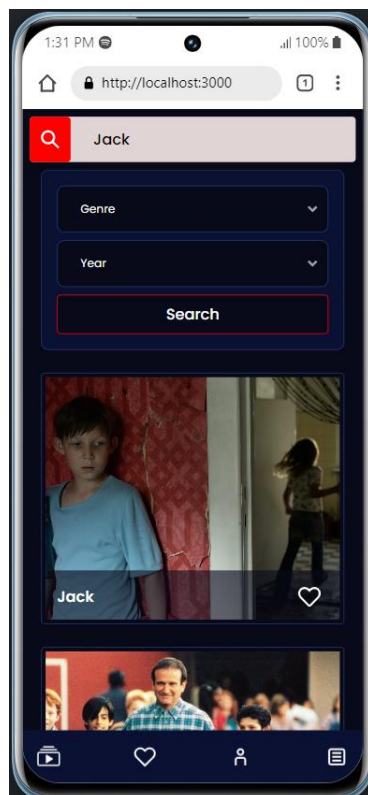


Рисунок 3.9 – Телефона версія сторінки пошуку фільмів з результатом пошуку за назвою «Jack»

Натиснувши на постери фільмів, користувач переходить до сторінки фільму, що містить інформацію про нього. Також є кнопки для збереження фільму та початку перегляду (Рисунки 3.10-11)

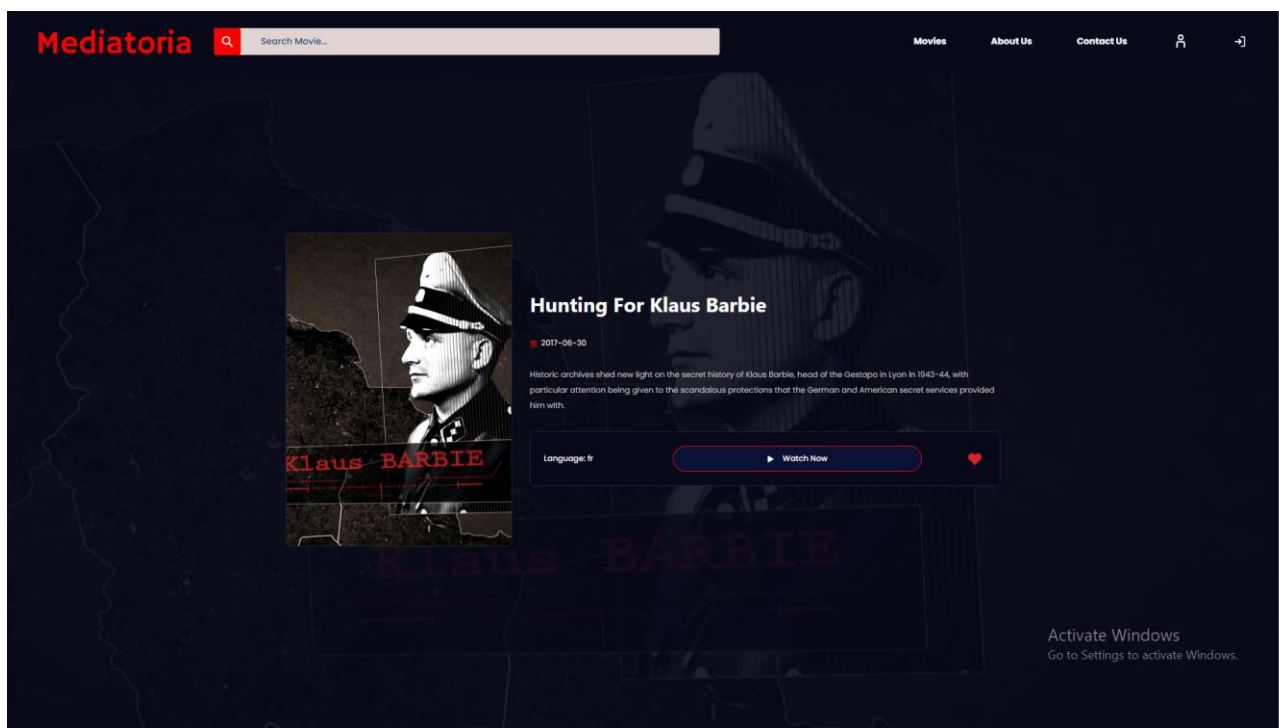


Рисунок 3.10 – Комп'ютерна версія сторінки фільму

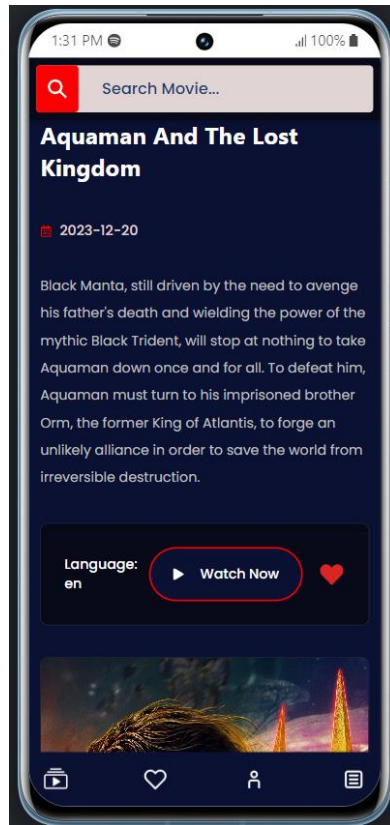


Рисунок 3.11 – Телефона версія сторінки фільму

На сторінці фільму користувач має змогу написати відгук у відповідному секторі зображеному на рисунках 3.12-13.

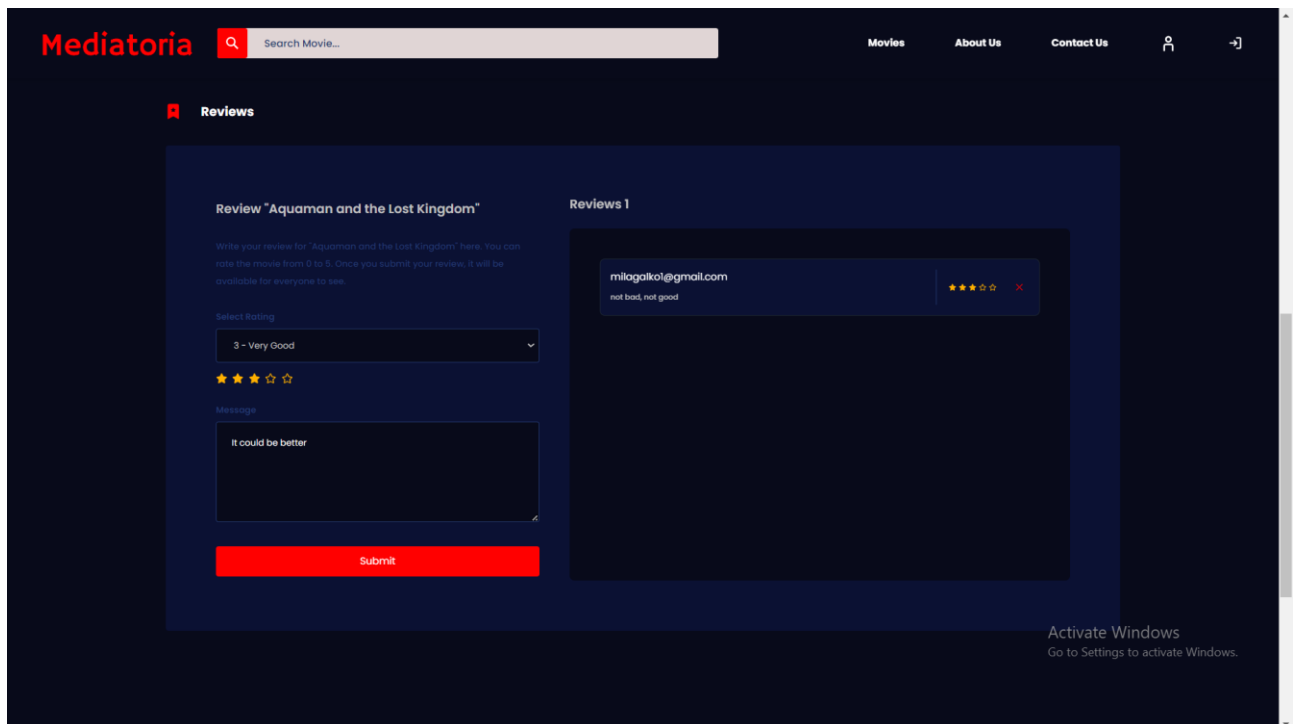


Рисунок 3.12 – Комп'ютерна версія сторінки фільму сектору написання відгуку



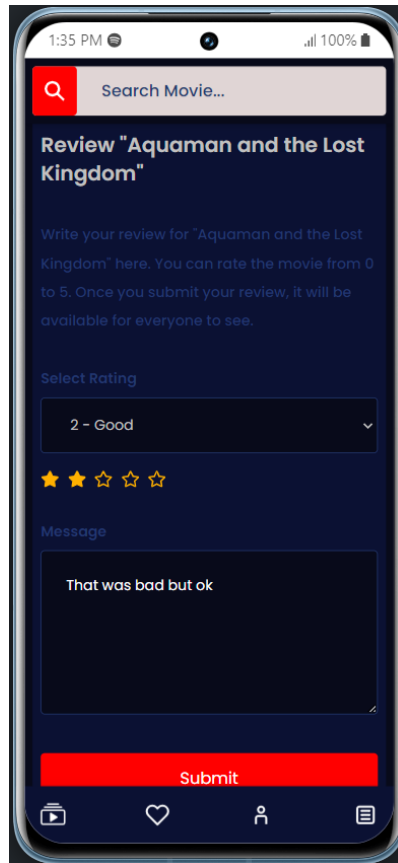


Рисунок 3.13 – Телефона версія сторінки фільму сектору написання відгуку

Якщо користувач натиснув кнопку «Watch Now» на сторінці обраного фільму, його переводить на сторінку перегляду фільму (Рисунки 3.14-15)

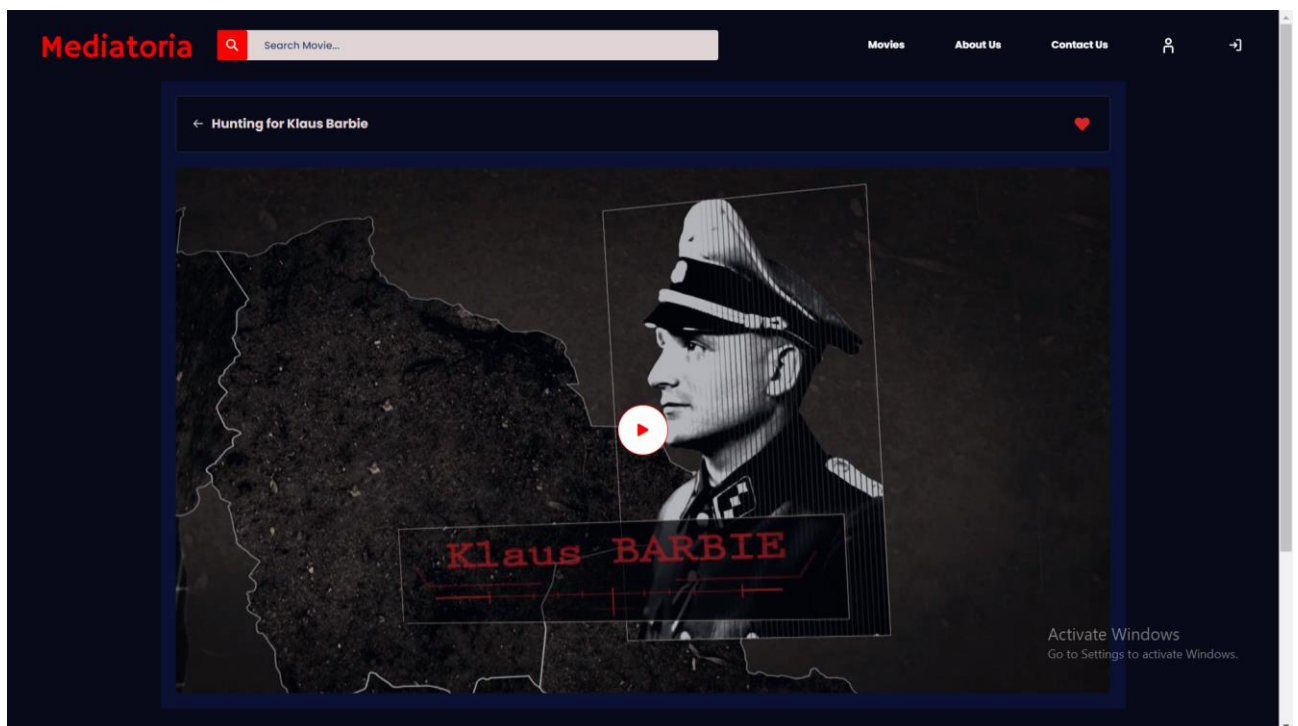


Рисунок 3.14 – Комп'ютерна версія сторінки перегляду фільму

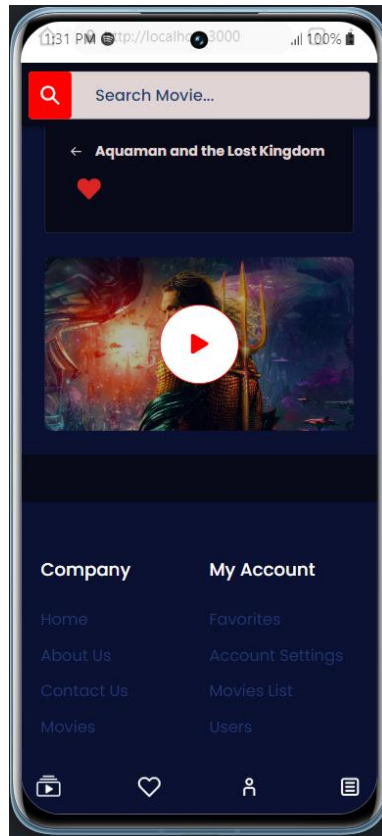


Рисунок 3.15 – Телефона версія сторінки перегляду фільму

Застосунок підтримує перегляд не тільки у форматі повного екрану, але й у «Picture in picture», що продемонстровано на рисунку 3.16.

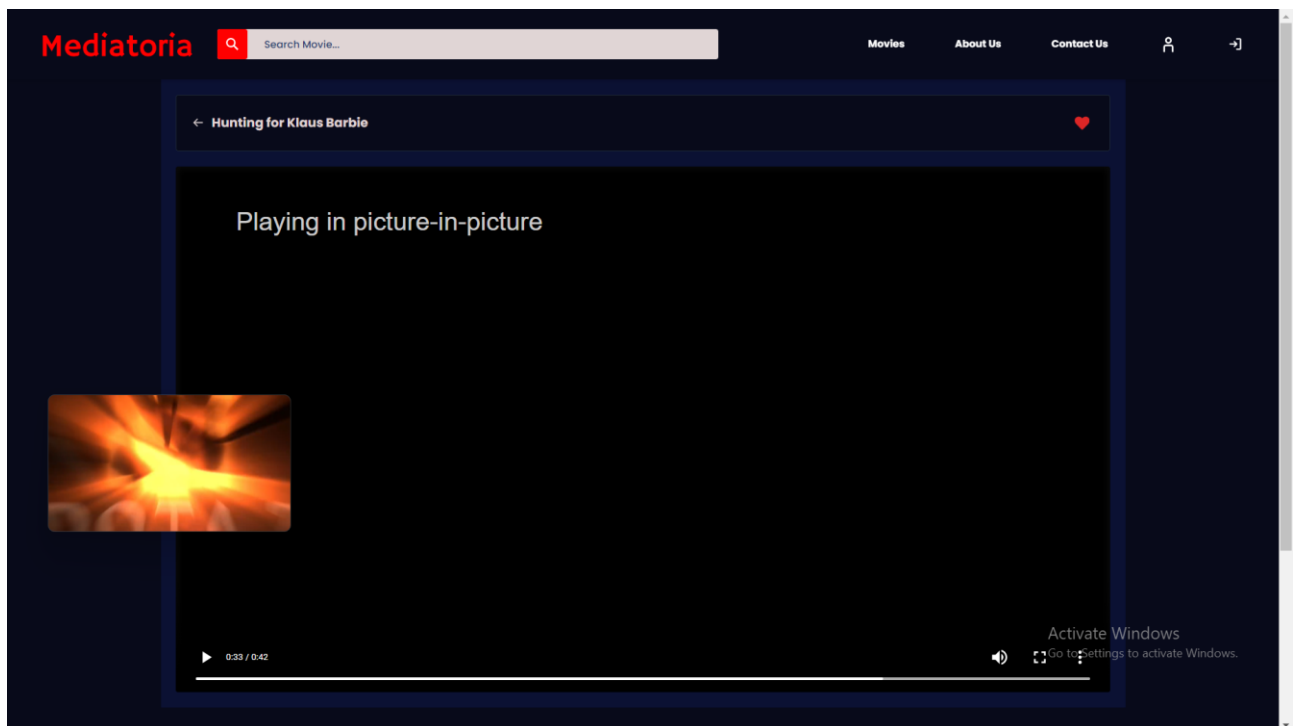


Рисунок 3.16 – Комп'ютерна версія сторінки перегляду фільму під час перегляду у форматі picture-in-picture

Для авторизації користувач натискає кнопку «Log In» і переходить на сторінку, що відображена на рисунках 3.17-18.

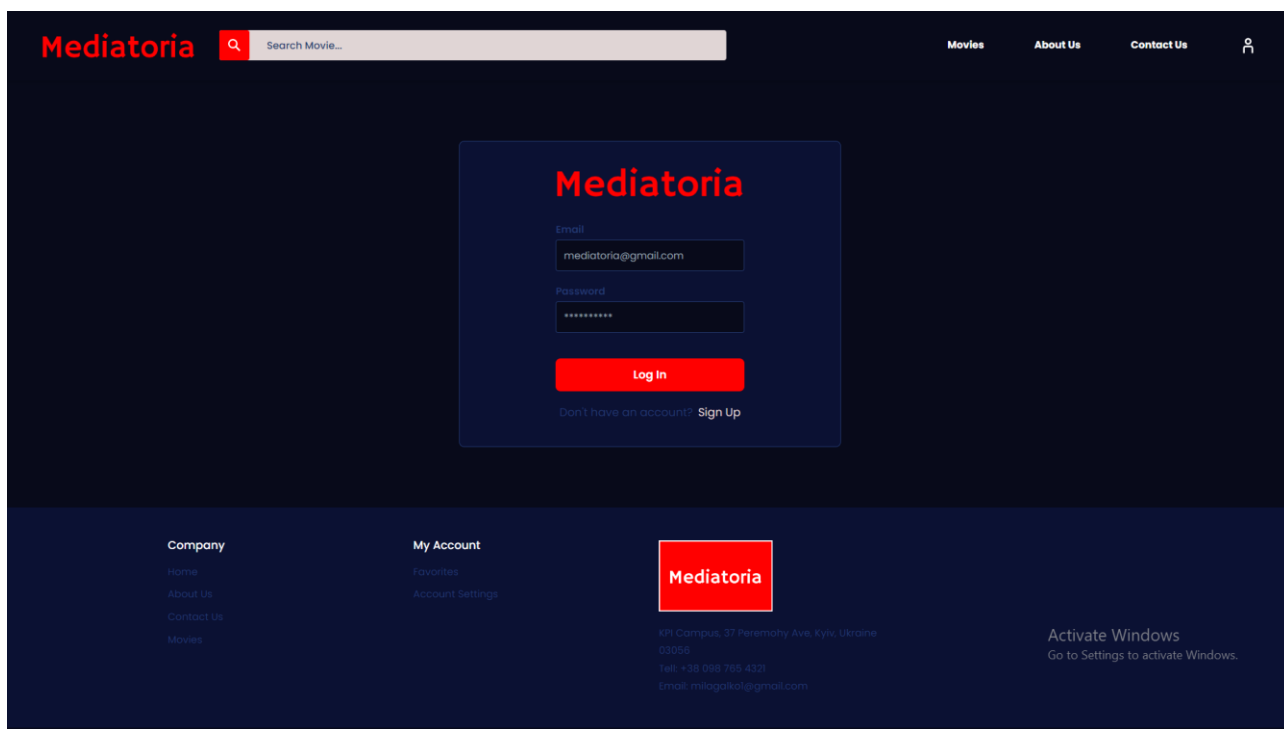


Рисунок 3.17 – Комп’ютерна версія сторінки входу

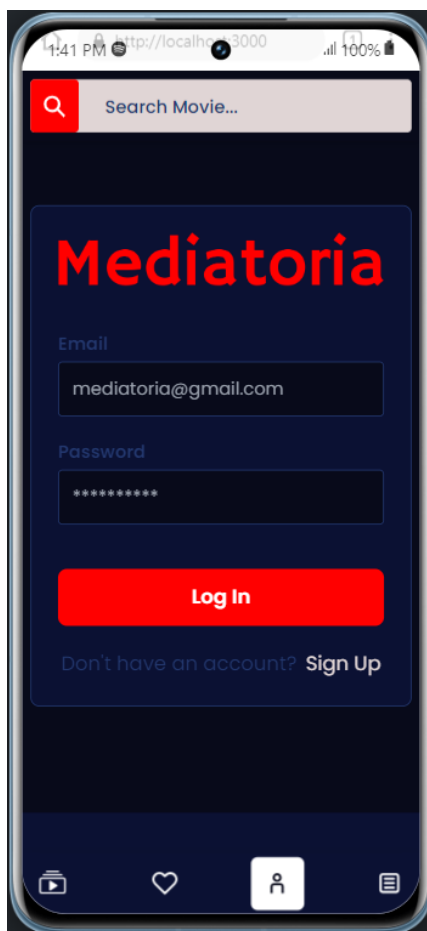
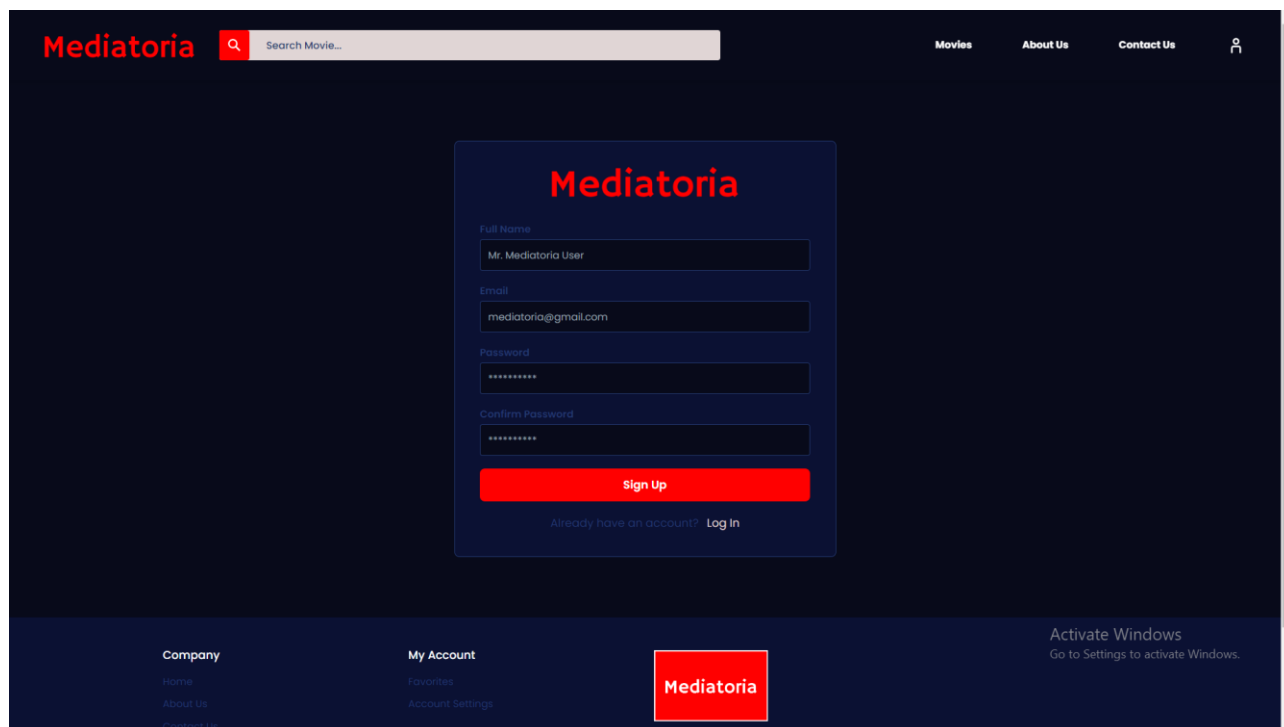


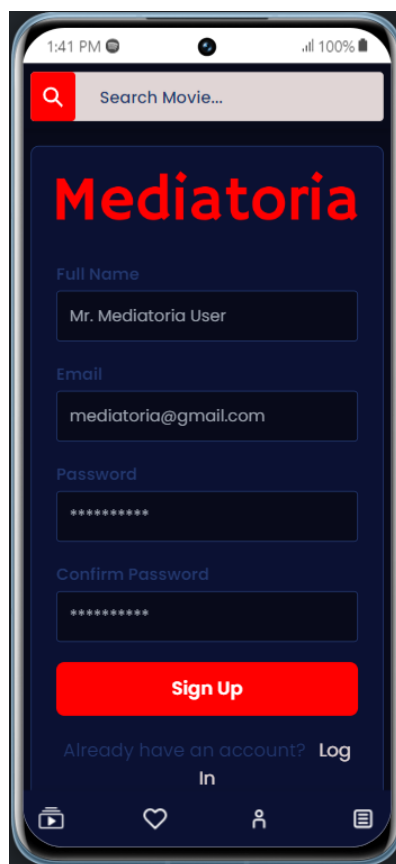
Рисунок 3.18 – Телефона версія сторінки входу

Якщо користувач не має акаунту, то натиснувши на посилання «Sign Up» на сторінці входу (Рисунки 3.17-18), переходить на сторінку реєстрації (Рисунки 3.19-20).



The screenshot shows a desktop browser window with the Mediatoria website. The header features the Mediatoria logo, a search bar, and navigation links for Movies, About Us, and Contact Us. The main content area displays a registration form with the following fields: Full Name (filled with 'Mr. Mediatoria User'), Email (filled with 'mediatoria@gmail.com'), Password (masked with dots), and Confirm Password (masked with dots). A red 'Sign Up' button is positioned below the form, and a link 'Already have an account? Log In' is located underneath. The footer contains links for Company (Home, About Us, Contact Us) and My Account (Favorites, Account Settings), along with a Windows activation notice.

Рисунок 3.19 – Комп’ютерна версія сторінки реєстрації



The screenshot shows the Mediatoria registration page on a mobile phone. The interface is adapted for a smaller screen, with the search bar and navigation links at the top. The registration form fields (Full Name, Email, Password, Confirm Password) and the 'Sign Up' button are clearly visible. A 'Log In' link is also present at the bottom of the form. The phone's status bar at the top shows the time as 1:41 PM and 100% battery. The bottom of the screen features a navigation bar with icons for home, favorites, account, and a menu.

Рисунок 3.20 – Телефона версія сторінки реєстрації

У телефонній версії також передбачене альтернативне представлення навігації між сторінками (Рисунок 3.21).

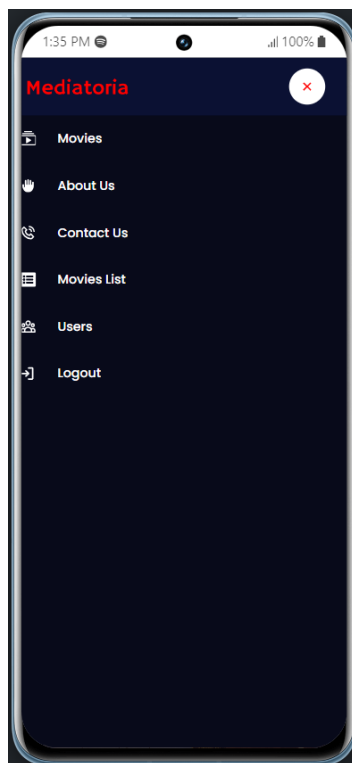


Рисунок 3.21 – Телефона версія навігації між сторінками

Натиснувши на іконку у формі користувача, юзер переходить на сторінку свого профілю, де демонструються його особисті дані (Рисунки 3.22-23).

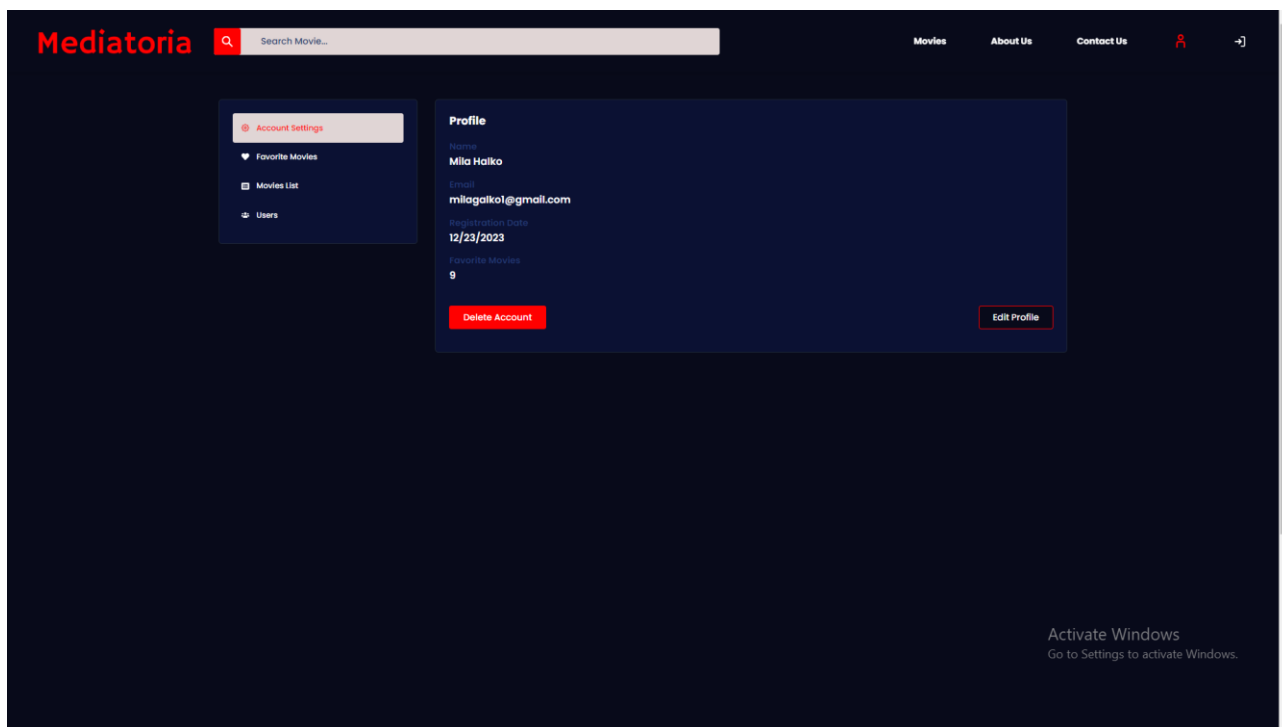


Рисунок 3.22 – Комп'ютерна версія сторінки профілю

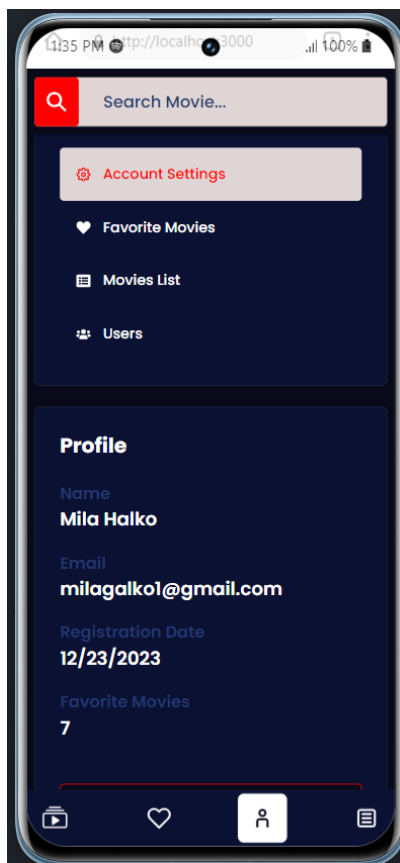


Рисунок 3.23 – Телефона версія сторінки профілю

Натиснувши на кнопку «Edit Profile», користувач має змогу змінити свої дані та зберегти їх (Рисунки 3.24-25).

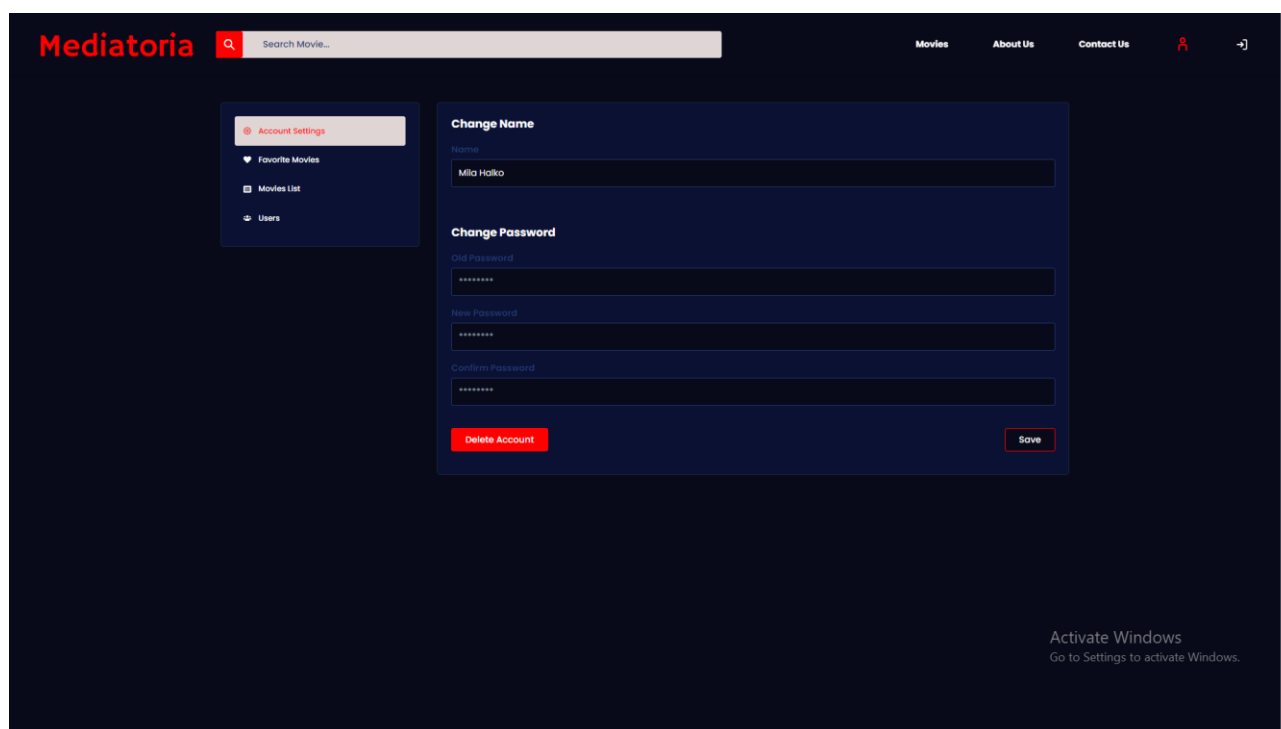


Рисунок 3.24 – Комп'ютерна версія сторінки профілю стан редагування

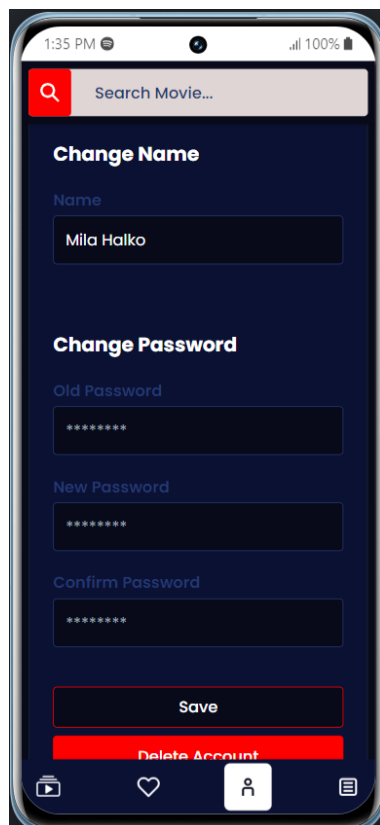


Рисунок 3.25 – Телефона версія сторінки профілю стан редагування

На сторінці «Favorite Movies» користувач може перегляди список своїх збережених фільмів та переходити на їх сторінки (Рисунки 3.26-27).

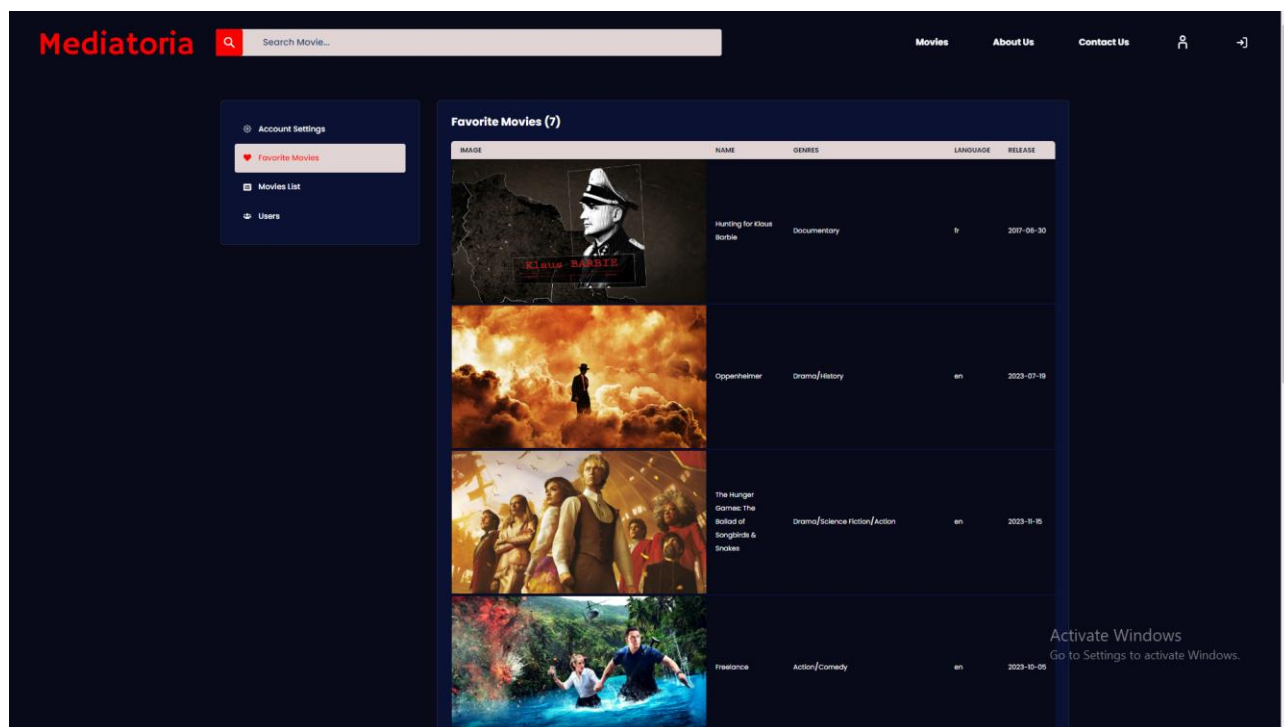


Рисунок 3.26 – Комп'ютерна версія сторінки збережених фільмів

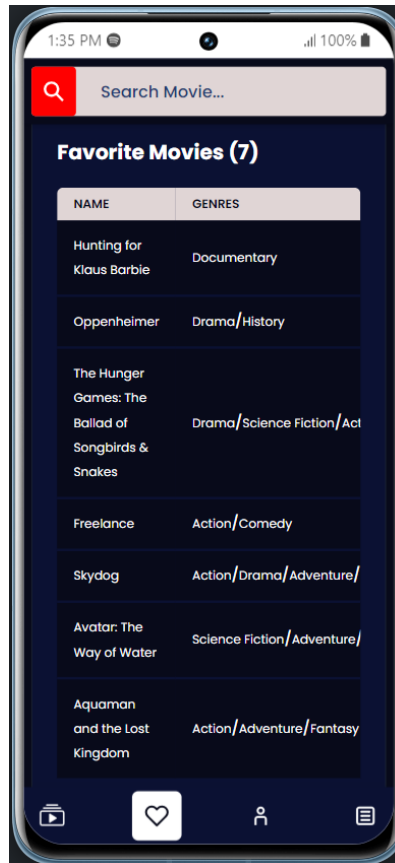


Рисунок 3.27 – Телефона версія сторінки збережених фільмів

У адміністратора також є можливість перегляду списку усіх зареєстрованих користувачів із можливістю видалення не адміністративних акаунтів (Рисунки 3.28-29).

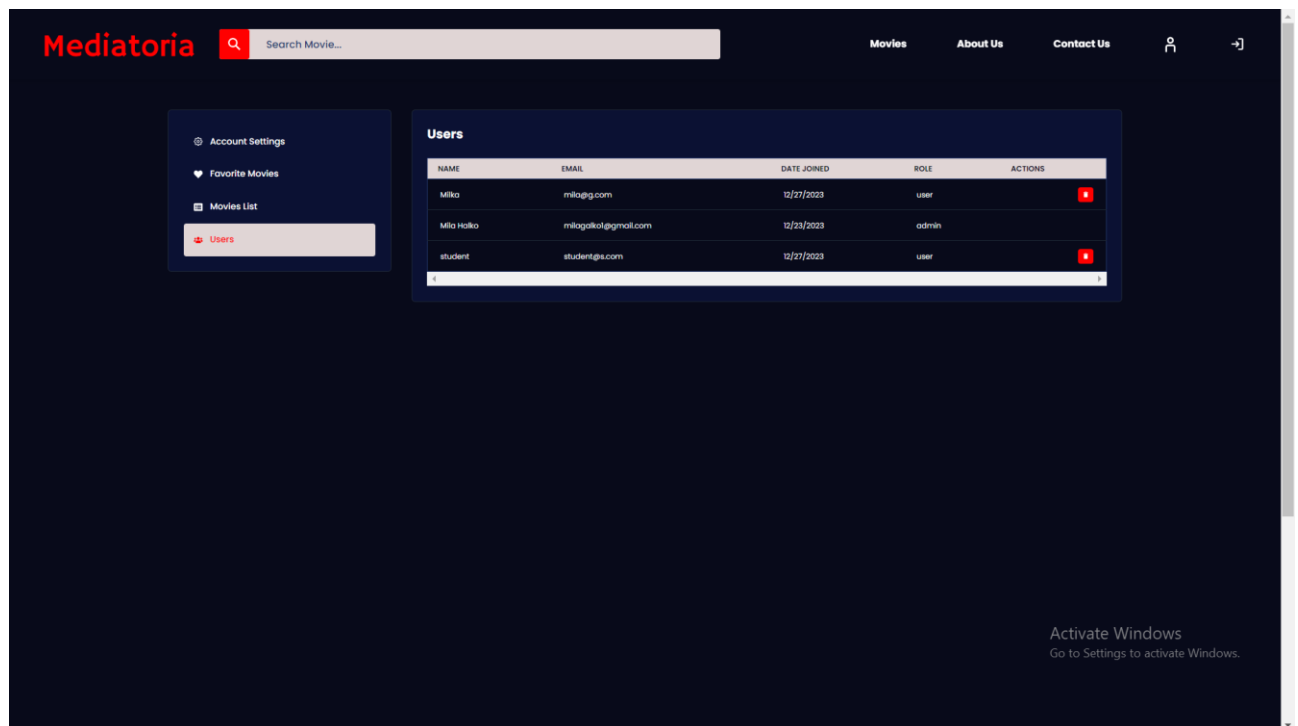


Рисунок 3.28 – Комп'ютерна версія сторінки зареєстрованих користувачів



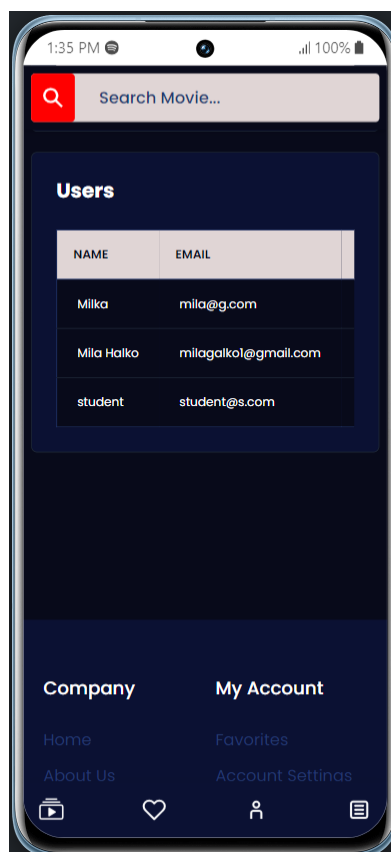


Рисунок 3.29 – Телефона версія сторінки зареєстрованих користувачів

### Висновки до розділу

Розділ допоміг вдосконалення веб-застосунку "Mediatoria" через аналіз якості коду за допомогою інструменту SonarCloud та розробка тестів. Це дозволило забезпечити високий рівень стабільності, безпеки та ефективності програмного продукту.

В процесі аналізу якості коду за допомогою інструменту SonarCloud було отримано важливі показники, які свідчать про високий ступінь стабільності та читабельності кодової бази. Рейтинги безпеки, надійності та збереження показують високу якість програмного забезпечення.

Для покриття основних функціональностей "Mediatoria" було розроблено 10 тестів, які успішно виконалися. Ці тести охоплюють різні аспекти, такі як аутентифікація користувача, робота з фільмами, управління обраною інформацією, коментарі. Усі тести пройшли успішно, підтверджуючи коректну роботу програмного продукту.

В кінці був продемонстрований основний функціонал веб-застосунку як на комп'ютері, так і на телефоні. Користувач може легко переглядати фільми, додавати їх до списку улюблених, залишати коментарі та насолоджуватися інтерактивним інтерфейсом.

## 4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 4.1 Розгортання програмного забезпечення

Для розгортання було використано сервіс Firebase, який надає можливості для постійної інтеграції і розгортання. Для використання функціоналу маємо завантажити інструменти як показано на рисунку 4.1.

```
PS C:\Users\milag\OneDrive\Documents\CodeZone\S7_CourseWork_Cinema\streaming-service-app> npm i -g firebase-tools

removed 32 packages, and changed 639 packages in 53s

65 packages are looking for funding
  run `npm fund` for details
```

Рисунок 4.1 – Завантаження інструментів Firebase до проєкту

Першим етапом розгортання є ініціалізація проєкту у Firebase. За допомогою команди `firebase init` обираємо опції проєкту та налаштовує основні параметри, такі як публічний каталог та конфігурація для Firebase Hosting (Рисунок 4.2). У випадку застосунка "Mediatoria Streaming Service", публічний каталог обрано як `build`.

```
PS C:\Users\milag\OneDrive\Documents\CodeZone\S7_CourseWork_Cinema\streaming-service-app> firebase init

#####

  You're about to initialize a Firebase project in this directory:

    C:\Users\milag\OneDrive\Documents\CodeZone\S7_CourseWork_Cinema\streaming-service-app

  ? Are you ready to proceed? Yes
  ? Which Firebase features do you want to set up for this directory? Press Space to select features, then Enter to confirm your choices. Hosting: Configure files for Firebase Hosting and (optionally) set up GitHub Action deploys

  === Project Setup

  First, let's associate this project directory with a Firebase project.
  You can create multiple project aliases by running firebase use --add,
  but for now we'll just set up a default project.

  ? Please select an option: Use an existing project
  ? Select a default Firebase project for this directory: mediatoria-stream-service (mediatoria-stream-service)
  1 Using project mediatoria-stream-service (mediatoria-stream-service)

  === Hosting Setup

  Your public directory is the folder (relative to your project directory) that
  will contain Hosting assets to be uploaded with firebase deploy. If you
  have a build process for your assets, use your build's output directory.

  ? What do you want to use as your public directory? build
  ? Configure as a single-page app (rewrite all urls to /index.html)? Yes
  ? Set up automatic builds and deploys with GitHub? No
  + Wrote build/index.html

  1 Writing configuration info to firebase.json...
  1 Writing project information to .firebaserc...
  + Firebase initialization complete!
```

Рисунок 4.2 – Ініціалізація проєкту у Firebase

Після ініціалізації важливо завантажити необхідні залежності та зібрати проєкт у вигляді оптимізованого продукту. Для цього маємо виконати команду `yarn run build`.

Останній крок - розгортання застосунка на Firebase Hosting. Виконуючи команду `firebase deploy`, ми завантажуюмо створений оптимізований білд на сервери Firebase. У виводі консолі видно, що розгортання проходить успішно, і ми отримуємо публічний Hosting URL, за яким можна перейти до застосунка в Інтернеті (Рисунок 4.3).

```
=== Deploying to 'mediatoria-stream-service'...

i  deploying hosting
i  hosting[mediatoria-stream-service]: beginning deploy...
i  hosting[mediatoria-stream-service]: found 53 files in build
+  hosting[mediatoria-stream-service]: file upload complete
i  hosting[mediatoria-stream-service]: finalizing version...
+  hosting[mediatoria-stream-service]: version finalized
i  hosting[mediatoria-stream-service]: releasing new version...
+  hosting[mediatoria-stream-service]: release complete

+  Deploy complete!

Project Console: https://console.firebase.google.com/project/mediatoria-stream-service/overview
Hosting URL: https://mediatoria-stream-service.web.app
PS C:\Users\milag\OneDrive\Documents\CodeZone\S7_CourseWork_Cinema\streaming-service-app> |
```

Рисунок 4.3 – Успішне розгортання застосунку

При переході за посиланням успішно завантажувється сторінка головного екрану (Рисунок 4.4).

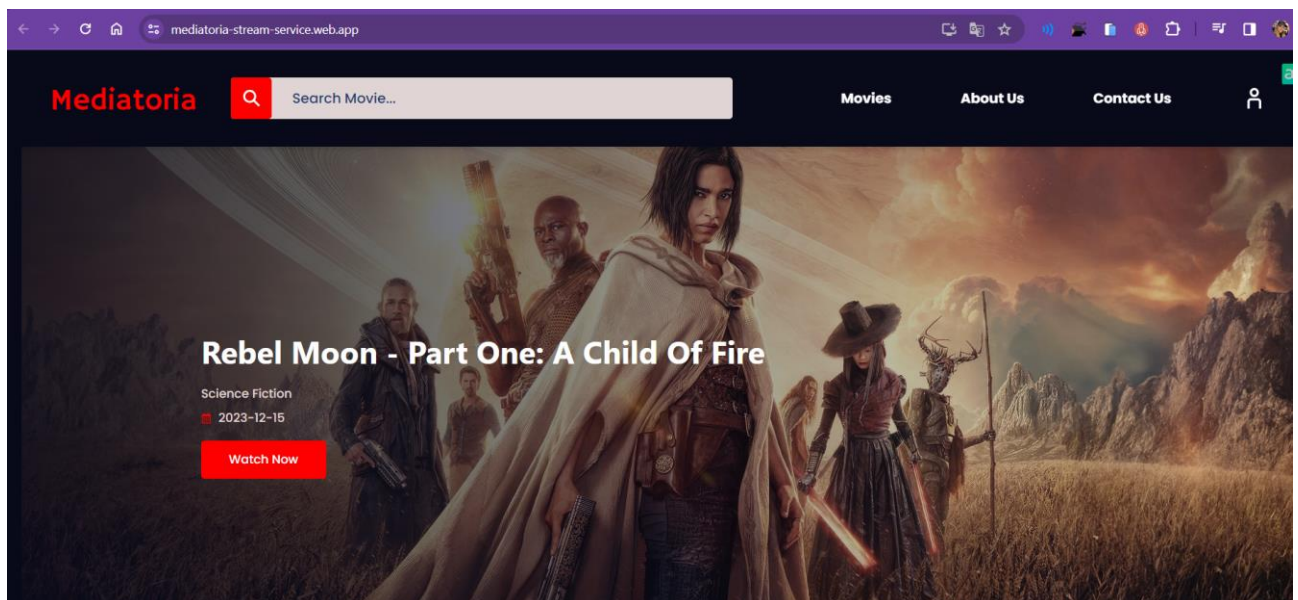


Рисунок 4.4 – Успішне завантаження застосунку в інтернеті

## 4.2 Підтримка програмного забезпечення

Підтримка ПЗ – етап, що спрямований на забезпечення стабільної та безперебійної роботи продукту після його розгортання.

Одним із способів забезпечення користувачів завжди останньою версією є автоматичні оновлення. Оскільки маємо веб-застосунок, то за оновлення відповідає сам розробник. Для веб-сайту, розгорнутого на Firebase розробником, можна використовувати механізми, які забезпечують автоматичне оновлення клієнтської частини через використання сервісу Firebase Hosting. Він автоматично розгортає нові версії застосунку.

Firebase також надає потужні інструменти для збору та аналізу даних взаємодії користувачів. Він дозволяє отримувати детальну інформацію про взаємодію з контентом, конверсії та інші ключові метрики. Серед яких є наступні:

- Події, які відбуваються на веб-сайті: як приклад, перегляд сторінки, клік на кнопку, реєстрація користувача тощо. Збір та аналіз подій дозволяє здійснювати глибокий розгляд користувацького досвіду;
- Аналіз утримання (Retention): визначає, яка частина користувачів залишається активною протягом певного періоду. Це дозволяє визначити ефективність стратегій залучення користувачів;
- Ворота для аналізу конверсії: наприклад, для визначення кількості користувачів, які відвідали головну сторінку, потрапили на сторінку реєстрації та завершили реєстрацію.

### Висновки до розділу

В даному розділі спочатку був детально розглянутий процес розгортання програмного забезпечення за допомогою Firebase. Встановлення Firebase CLI та ініціалізація проекту дозволили ефективно оптимізувати фронтенд та забезпечити доступ до застосунку онлайн. За допомогою Firebase Hosting досягли швидкого та зручного розгортання веб-сайту.

Далі було обговорено підтримку та оновлення програмного забезпечення. Firebase Hosting автоматично надає підтримку оновлення версій, що дозволяє розробнику зосередитися на функціональності, забезпечуючи високу доступність та надійність веб-сайту. Завдяки можливостям Firebase Analytics відбувається збір та аналіз даних взаємодії користувачів, що дозволяє отримати інсайти для вдосконалення застосунку.

## ВИСНОВКИ

В результаті виконання курсової роботи було спроектовано та розгорнуто веб-застосунок «Mediatoria», що став предметом досліджень та реалізації зручного та функціонального інструменту для перегляду та взаємодії з фільмами онлайн з різних пристроїв.

Науково-дослідницька частина роботи включала аналіз функціоналу онлайн кінотеатрів, детальний опис і аналіз предметної області та вивчення існуючих технологій, таких як Firebase Auth, Firestore та TMDb. Зокрема, було проведено аналіз функціоналу популярних онлайн-платформ, а також досліджено попередні рішення, які забезпечували б аутентифікацію, збереження даних та отримання інформації про фільми. Результатом було чітке формулювання вимог до функціоналу Mediatoria, враховуючи найкращі практики та існуючі стандарти.

В процесі моделювання та конструювання Mediatoria була розроблена чітка архітектура програмного забезпечення. Клієнт-серверна архітектура забезпечує ефективну взаємодію з Firebase Auth та Firestore. Створено схему у вигляді діаграми компонентів, яка ілюструє основні компоненти та їх взаємодію. Також, у розділі аналізу безпеки даних висвітлено заходи безпеки, що використовуються Firebase Auth та Firestore для захисту конфіденційної інформації користувачів.

В розділі аналізу якості та тестування ПЗ використовувалися різні види тестів. Кожен модуль та функціональність програмного забезпечення були піддані ретельному аналізу з метою виявлення помилок та недоліків. Контрольні приклади були продемонстровані для перевірки коректності роботи основних функцій та взаємодії з Firebase Auth та Firestore.

Результати розробки Mediatoria повністю відповідають сучасному рівню наукових та технічних знань у галузі веб-розробки та хмарних сервісів. Використання передових технологій, таких як Firebase, дозволило створити продукт, який відповідає сучасним вимогам щодо безпеки, ефективності та зручності використання.

Розгортання Mediatoria через Firebase Hosting дозволило забезпечити доступ до застосунку в онлайн-режимі. Підтримка оновлень та розвиток продукту здійснюється автоматично через взаємодію розробника застосунку із системою. Можливість збору статистики за допомогою аналітичних інструментів Google Analytics відкриває можливості для подальшого вдосконалення сервісу до потреб користувачів.

Завдяки комплексному підходу до розробки веб-застосунків Mediatoria став не тільки результатом наукових досліджень, але й ефективним інструментом для задоволення потреб користувачів. Розвиток застосунку та його підтримка реалізуються відповідно до сучасних вимог, а використання Firebase забезпечує надійність та безпеку обробки користувацьких даних. Продукт залишається відкритим для подальших вдосконалень та розширень, враховуючи зростаючі потреби та тенденції у сфері онлайн-розваг.



## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) Firestore | Firebase. URL: <https://firebase.google.com/docs/firestore> (дата звернення: 30.12.2023).
- 2) Firebase Authentication. URL: <https://firebase.google.com/docs/auth> (дата звернення: 30.12.2023).
- 3) Firebase Authentication Password Hashing. URL: <https://firebaseopensource.com/projects/firebase/scrypt/> (дата звернення: 30.12.2023).
- 4) Kosinski M., Forrest A. What is 2FA: two-factor authentication?. URL: <https://www.ibm.com/topics/2fa> (дата звернення: 30.12.2023).
- 5) Introduction to the Admin Auth API | Firebase Authentication. URL: <https://firebase.google.com/docs/auth/admin> (дата звернення: 30.12.2023).
- 6) React. URL: <https://react.dev/> (дата звернення: 30.12.2023).
- 7) Getting started with React - Learn web development | MDN. URL: [https://developer.mozilla.org/en-US/docs/Learn/Tools\\_and\\_testing/Client-side\\_JavaScript\\_frameworks/React\\_getting\\_started](https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started) (дата звернення: 30.12.2023).
- 8) Home v6.21.1 | React Router. URL: <https://reactrouter.com/> (дата звернення: 30.12.2023).
- 9) Authenticate Using Google with JavaScript | Firebase. URL: <https://firebase.google.com/docs/auth/web/google-signin> (дата звернення: 30.12.2023).
- 10) Add multi-factor authentication to your web app | Firebase Authentication. URL: <https://firebase.google.com/docs/auth/web/multi-factor> (дата звернення: 30.12.2023).
- 11) Control Access with Custom Claims and Security Rules | Firebase Authentication. URL:

- <https://firebase.google.com/docs/auth/admin/custom-claims> (дата звернення: 30.12.2023).
- 12) My SQL. My SQL. URL: <https://dev.mysql.com/doc/> (дата звернення: 30.12.2023).
- 13) PostgreSQL. PostgreSQL. URL: <https://www.postgresql.org/docs/> (дата звернення: 30.12.2023).
- 14) What is MongoDB? MongoDB Manual. MongoDB Documentation. URL: <https://docs.mongodb.com/manual/> (дата звернення: 30.12.2023).
- 15) Google Cloud documentation | Documentation. Google Cloud. URL: <https://cloud.google.com/docs/> (дата звернення: 30.12.2023).
- 16) Welcome to AWS Documentation. AWS. URL: <https://docs.aws.amazon.com/> (дата звернення: 30.12.2023).
- 17) Azure documentation. Microsoft Azure. URL: <https://docs.microsoft.com/en-us/azure/> (дата звернення: 30.12.2023).
- 18) Firestore and its advantages - GeeksforGeeks. GeeksforGeeks. URL: <https://www.geeksforgeeks.org/firestore-and-its-advantages/> (дата звернення: 30.12.2023).
- 19) What is Firebase? Tips, Tricks, & Secrets Unlocked. Back4App Blog. URL: <https://blog.back4app.com/firebase/#:~:text=Cloud%20Firestore,%20also%20known%20as,web%20applications%20via%20native%20SDKs.> (дата звернення: 30.12.2023).
- 20) Getting Started. The Movie Database (TMDB). URL: <https://developer.themoviedb.org/reference/intro/getting-started> (дата звернення: 30.12.2023).
- 21) Introduction to The Movie Database API. Educative. URL: <https://www.educative.io/courses/movie-database-api-python/introduction-to-the-movie-database-api> (дата звернення: 30.12.2023).
- 22) Hayani S. Here are the most popular ways to make an HTTP request in JavaScript. freeCodeCamp.org. URL:

- <https://www.freecodecamp.org/news/here-is-the-most-popular-ways-to-make-an-http-request-in-javascript-954ce8c95aaa/> (дата звернення: 30.12.2023).
- 23) Olawanle J. JavaScript Get Request – How to Make an HTTP Request in JS. freeCodeCamp.org. URL: <https://www.freecodecamp.org/news/javascript-get-request-tutorial/> (дата звернення: 30.12.2023).
- 24) Axios vs. fetch(): Which is best for making HTTP requests? - LogRocket Blog. LogRocket Blog. URL: <https://blog.logrocket.com/axios-vs-fetch-best-http-requests/> (дата звернення: 30.12.2023).
- 25) An In-Depth Guide to Axios: Making HTTP Requests with Ease. Medium. URL: <https://medium.com/@selieshjksofficial/an-in-depth-guide-to-axios-making-http-requests-with-ease-6eefa557bb07> (дата звернення: 30.12.2023).
- 26) Promise - JavaScript | MDN. MDN Web Docs. URL: [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Promise](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise) (дата звернення: 30.12.2023).
- 27) Event Streaming Made Easy with Event-Stream and JavaScript Fetch. Medium. URL: <https://medium.com/@bs903944/event-streaming-made-easy-with-event-stream-and-javascript-fetch-8d07754a4bed> (дата звернення: 30.12.2023).
- 28) Front End JavaScript Development Handbook – React, Angular, and Vue Compared. FreeCodeCamp. URL: <https://www.freecodecamp.org/news/front-end-javascript-development-react-angular-vue-compared/> (дата звернення: 30.12.2023).
- 29) jQuery. jQuery. URL: <https://jquery.com/> (дата звернення: 30.12.2023).
- 30) Angular vs React vs Vue: Which Framework Is Better? 2023. aThemes. URL: <https://athemes.com/guides/angular-vs-react-vs-vue/> (дата звернення: 30.12.2023).

- 31) Visual Studio: IDE and Code Editor for Software Developers and Teams. Visual Studio. URL: <https://visualstudio.microsoft.com/> (дата звернення: 30.12.2023).
- 32) IntelliJ IDEA overview. IntelliJ IDEA | Jet Brains. URL: <https://www.jetbrains.com/help/idea/discover-intellij-idea.html#user-interface> (дата звернення: 30.12.2023).
- 33) Meet WebStorm. WebStorm | Jet Brains. URL: <https://www.jetbrains.com/help/webstorm/meet-webstorm.html> (дата звернення: 30.12.2023).
- 34) WebStorm vs VS Code: Compare top IDEs. TechRepublic. URL: <https://www.techrepublic.com/article/webstorm-vs-vscode/> (дата звернення: 30.12.2023).
- 35) Netflix. Netflix. URL: <https://about.netflix.com/en> (дата звернення: 30.12.2023).