МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інформаційних технологій

БАКАЛАВРСЬКА РОБОТА

на тему:

«ОНЛАЙН-СЕРВІС ДЛЯ КІНОМАНІВ»

Рівень «бакалавр»

Галузь знань 12 «Інформаційні технології»

спеціальність 121 «Інженерія програмного забезпечення»

Виконав здобувач 4 курсу

Дацко Іван Андрійович

Керівник к.т.н., доцент

Трофименко Олена Григорівна

Рецензент к.т.н., доцент

Чикунов Павло Олександрович

Одеса – 2024

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»

Факультет кібербезпеки та інформаційних технологій
Кафедра інформаційних технологій
Галузь знань 12 Інформаційні технології
Спеціальність 121 «Інженерія програмного забезпечення»
Назва освітньої програми «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних технологій
доцент Н.І. Логінова
_____ «___» _____20__ року

З А В Д А Н Н Я
на кваліфікаційну (бакалаврську) роботу
здобувачу Дацко Івану Андрійовичу

1. Тема роботи: «Онлайн-сервіс для кіноманів»
   Керівник роботи: к.т.н, доцент Трофименко Олена Григорівна
   затверджені наказом НУ «ОЮА» від «02» квітня 2024 року № 809-06
2. Дата видачі завдання «15» вересня 2023 року
3. Строк подання здобувачем роботи «20» травня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Формування функціональних вимог	20.10.2023	
2	Аналіз та вибір засобів розробки	30.10.2023	
3	Розробка архітектури вебзастосунку	20.11.2023	
4	Проєктування бази даних	25.12.2023	
5	Розробка серверної частини з використанням ASP.NET CORE	12.02.2024	
6	Розробка клієнтської частини засобами Bootstrap	19.03.2024	
7	Тестування функціональності сайту	09.04.2024	
8	Оформлення звітної частини	20.05.2024	

Здобувач _____
(підпис) (прізвище та ініціали)

Керівник роботи _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дацко І. А. Онлайн-сервіс для кіноманів. – Кваліфікаційна робота бакалавра за спеціальністю 121 «Інженерія програмного забезпечення», освітньою програмою «Інженерія програмного забезпечення».

Кваліфікаційна робота викладена на 52 сторінці основного тексту і 63 сторінках загального тексту, містить 3 розділи, 27 рисунків, 15 таблиць, 2 додатки, 13 використаних джерел.

Метою роботи є розробка онлайн-сервісу для кіноманів, який слугуватиме платформою для надання довідкової інформації про фільми (актори, режисер, опис), відгуки, оцінки підписників та іншу інформацію про улюблені відеострічки.

Об’єктом дослідження є онлайн-сервіси для кіноманів та організація контенту на їх них.

Предмет дослідження – засоби вебтехнологій для розробки онлайн-сервісів для фанатів фільмів.

У першому розділі виконано аналіз предметної області та вибір засобів розробки. У другому розділі здійснено проєктування вебзастосунку. У третьому розділі описано розробку та тестування вебзастосунку.

Ключові слова: онлайн-сервіс, кіноман, C#, ASP.NET, вебзастосунок, фільм, MVC, відгук, авторизація.

ABSTRACT

Datsko I. A. Online service for movie database. – Bachelor's qualifying work on specialty 121 "Software engineering", educational program "Software Engineering".

The work consists of 52 pages of main text and 63 pages in total, including 3 chapters, 26 figures, 15 tables, 2 appendices, and 13 sources.

The goal of the work is a development of an online service for movie fans, which will serve as a platform for providing background information about movies (actors, director, description), reviews, user ratings and other information about favorite films.

The object of the study is online platforms for moviegoers and the organization of content on them.

The subject of research is web technology tools for online services for moviegoers.

In the first section, the analysis of the subject area and the selection of development tools were performed. In the second section, the design of the web application was carried out. The third section describes the development and testing of the web application.

Keywords: online service, movie fan, C#, ASP.NET, web application, movie, MVC, feedback, authorization.

ЗМІСТ

ВСТУП

Популярність кінофільмів і телевізійних шоу зростає з кожним роком, що створює великий попит на онлайн-сервіси, які надають рецензії, оцінки та іншу інформацію про фільми. Особливістю таких сайтів є те, що вони мають велику кількість контенту, який можна оцінювати, фільтрувати, писати відгуки та рекомендації. Поява онлайн-платформ для стрімінгу (Netflix, Amazon Prime, Disney+) надала зручний доступ до перегляду фільмів і серіалів. Крім того, кіно та телевізійні шоу часто стають об’єктом обговорення у різних соціальних мережах, що стимулює попит на стрімінгові онлайн-сервіси з рецензіями та оцінками. Тому розробка сучасного вебсайту для кіноманів, який матиме широкий функціонал та зручний інтерфейс користувача є актуальною.

Метою роботи є розробка онлайн-сервісу для кіноманів, який слугуватиме платформою для надання довідкової інформації про фільми (актори, режисер, опис), відгуки, оцінки підписників та іншу інформацію про улюблені відеострічки.

Об’єктом дослідження є онлайн-сервіси для кіноманів та організація контенту на їх них.

Предмет дослідження – засоби вебтехнологій для розробки онлайн-сервісу для фанатів фільмів.

Відповідно до поставленої мети **завданнями** роботи є:

– формування функціональних вимог до створюваного програмного вебсайту;

– аналіз предметної галузі, включаючи аналіз наявних сайтів зі схожою структурою та функціоналом, для з’ясування та уточнення функцій, структури, інтерфейсу, які матиме онлайн-сервіс для якнайкращого задоволення вимог користувача;

– аналіз та вибір технологій для розробки frontend та backend сайту для онлайн-кіноманів;

– проєктування структури бази даних та моделей даних вебзастосунку;

– розробка серверної частини з використанням ASP.NET CORE;

– розробка клієнтської частини засобами Bootstrap;

– тестування розробленого вебзастосунку.

Результати роботи були апробовані у таких наукових публікаціях автора:

1. Дацко І. А. «Засоби .NET для розробки програмного забезпечення. *Кібербезпека в сучасному світі*»: матер. III всеукр. наук.-практ. конф. (м. Одеса, 19 листопада 2021 р.). Одеса, 2021. С. 124-127. URL: https://dspace.onua.edu.ua/items/deb11a37-0bd2-4f3b-9f9c-f783fdf8063c.

2. Дацко І. А. Засоби Hardhat для автоматизованого тестування програмних застосунків у мережі блокчейн. *Інформаційне суспільство: проблеми та перспективи* : матер. VIII всеукр. наук.-практ. конф. (м. Одеса, 2 черв. 2023 р.). Одеса, 2023. 93-94 с. URL: https://doi.org/10.32837/11300.25263.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИБІР ЗАСОБІВ РОЗРОБКИ

1.1 Аналіз особливостей розробки онлайн-сервісів для кіноманів

Кіноіндустрія є масштабним проявом соціокультурного буття сучасного суспільства. Кінематограф має значний бюджет на рекламу та маркетинг, тому сайти з рецензіями можуть залучати рекламодавців і партнерів для співпраці. Також вплив кінокритиків та рецензентів суттєво впливають на популярність фільму, тому такі онлайн-сервіси відіграють вагому роль у формуванні публічної думки про той чи інший фільм.

Розробка сайту з рецензіями на фільми має свої особливості через специфіку контенту та потреб користувачів. Аспекти, які варто врахувати під час розробки онлайн-сервісу для кіноманів:

– *база даних фільмів:* сайт має мати обширну базу даних фільмів з інформацією про назви, жанри, роки випуску, режисерів, акторів та інші характеристики фільмів. Ця база даних має вчасно оновлюватись та бути актуальною;

– *функціональність рецензування:* користувачі мають мати можливість писати рецензії на фільми та виставляти їм оцінки. Такий функціонал стосується текстових відгуків, рейтингів за зірками або числових оцінок;

– *система рейтингів та відгуків:* онлайн-сервіс повинен мати систему оцінювання фільмів або телесеріалів у формі певної кількості зірок або числовому еквіваленті;

– *функціональність пошуку та фільтрації:* користувачам треба надати зручний інструментарій для пошуку фільмів за різними критеріями: назвою, режисером, жанром, акторами, роком випуску тощо. Тобто потрібна ефективна система фільтрації контенту;

– *авторизація та аутентифікація користувачів:* сайт повинен мати можливість реєстрації своїх користувачів, а зареєстрованим користувачам можна надавати додаткові функціональні можливості, наприклад: створення списку

переглянутого контенту, написання рецензій, додавання фільмів у список запланованих на перегляд тощо;

– *адміністративний розділ:* на сайті має бути модерація, що буде фільтрувати контент, який потрапляє на сайт, буде слідкувати за рецензіями, додаванням нових фільмів та враховувати інші можливі аспекти адміністрування такого онлайн-сервісу;

– *адаптація до мобільних пристроїв:* більшість користувачів у сучасному світі використовують мобільні пристрої для доступу до інтернету, тож сайт має бути адаптований до різних мобільних телефонів: розмірів екранів, браузерів та операційних систем;

– *захист даних:* важливим аспектом є захист користувацьких даних, їхніх акаунтів, рецензій, оцінок та іншої конфіденціальної інформації.

Розглянутий перелік особливостей розробки онлайн-сервісу для кіноманів по суті формує основні вимоги до функціоналу подібних сайтів. Для створення ж MVP такого сервісу достатнім може бути наявність функціональної системи рецензій та відгуків, та можливість базової взаємодії із базою даних фільмів для їх пошуку та категоризації.

1.2 Огляд програмних конкурентів

Онлайн-сервіси для потокового перегляду фільмів і телевізійних шоу полегшують виробництво та розповсюдження різноманітного та нішевого контенту. Через пандемію COVID-19 кількість прихильників та поціновувачів таких онлайн-сервісів суттєво зросла. Потокові сервіси Netflix, Amazon Prime і Hulu змінили споживання медіа, оскільки тепер споживачі можуть отримати інтернет-доступ до величезної бази відеоконтенту лише за декілька кліків [1]. Наразі спостерігається зростання кількості не лише англомовних, а й українських стрімінгових сервісів та онлайн-ресурсів для любителів кіно. Ці сервіси пропонують широкий вибір фільмів – від українських класиків до міжнародних блокбастерів, а також новини, рецензії та інтерв’ю [2].

Для аналізу програмних конкурентів такого онлайн-сервісу для кіноманів були вибрані IMDb, Rotten Tomatoes, Metacritic, CinemaBlend, Letterboxd, TMDB:

1) *IMDb* (Internet Movie Database, https://www.imdb.com/) є одним з найбільш відомих і популярних сайтів для кінофанатів від Amazon. Він має велику базу даних фільмів, акторів, телевізійних шоу та рецензій, а також пропонує функціонал, який дозволяє створювати списки перегляду, оцінювати та обговорювати фільми. IMDb є одним з найбільш відомих сайтів про кіно та телевізійні шоу. Аудиторія IMDb охоплює кіноманів, акторів, продюсерів, режисерів і всіх, хто цікавиться фільмами та телебаченням. IMDb має інтерфейс різними мовами (англійська, французька, німецька, іспанська, португальська, італійська та інші) і є популярним у багатьох країнах світу [3]. IMDb може допомогти користувачеві пригадати фільм, серіал або актора, найти для перегляду найкращий фільм чи шоу, поділитися своїми враженнями зі спільнотою шанувальників. Сервіс має розклад місцевих кіносеансів, продаж квитків, трейлери, відгуки критиків і користувачів, персоналізовані рекомендації, фотогалереї, розважальні новини, цитати, дрібниці, дані про касові збори, розділи редакційних тем тощо;

2) *Rotten Tomatoes* (https://www.rottentomatoes.com/) – популярний сайт для збору рецензій на фільми та телевізійні шоу. Він відомий своїми "томатними оцінками" (Tomatometer), які базуються на агрегованих відгуках сотень кіно– та телекритиків. Ще за часів відкритих театрів, коли вистава була провальною, глядачі висловлювали своє невдоволення не лише освистуванням, а й кидали на сцену все, що потрапляло під руку, в тому числі овочі та фрукти. Оцінка Tomatometer розраховується для фільму чи телешоу після того, як він отримує принаймні п’ять рецензій. Якщо принаймні 60% рецензій на фільм або телешоу є позитивними, відображається червоний помідор, який вказує на його «свіжий червоний статус». Якщо менше 60% рецензій на фільм або телешоу є позитивними, відображається позначка «зелений помідор», яка вказує на посередній статус. Позначка «зів'ялий помідор» свідчить про те, що недостатньо оцінок для створення статусу [4]. Rotten Tomatoes має велику аудиторію, особливо серед кінофанів, які цікавляться оглядами і рейтингами фільмів та телевізійних шоу;

3) *Metacritic* (https://www.metacritic.com/) відомий своїми агрегованими рейтингами фільмів і телевізійних шоу. Він збирає рецензії від критиків і на основі середньої оцінки формує загальний рейтинг. Metacritic залучає аудиторію як фанатів кінофільмів, так і їх критиків. Цей сайт використовує агреговані оцінки від критиків та звичайних користувачів для створення загального рейтингу фільмів і телевізійних шоу. Він має велику аудиторію підписників, які цікавляться кіноіндустрією;

4) *CinemaBlend* (https://www.cinemablend.com/) – це сайт з рецензіями та новинами про кіноіндустрію. Він відомий своїми оригінальними матеріалами, статтями та інтерв'ю з акторами та режисерами. Cinemablend дозволяє кіношанувальникам досліджувати та обговорювати фільми, телешоу та найкращі потокові пропозиції;

5) *Letterboxd* (https://letterboxd.com/) є соціальною мережею для кіноманів, де користувачі можуть відслідковувати свої перегляди, писати рецензії та обмінюватися думками про фільми з іншими користувачами. Letterboxd залучає аудиторію як кінофанів, так і пасіонатів, які люблять вести облік своїх переглядів і писати рецензії. Це особливо популярно серед молоді та активних користувачів соціальних мереж;

6) *TMDb* (The Movie Database, https://www.themoviedb.org/) – це сайт з відкритою базою даних фільмів, який збирає рецензії та іншу інформацію про фільми та акторів. На відміну від схожого комерційного сайту IMDb, який стягує немалі щорічні внески з клієнтів за використання своїх даних, TMDb має відкритий вихідний код і доступ для кінофанів та професіоналів кіноіндустрії. TMDb має широку аудиторію кіноманів, які цікавляться детальною інформацією про фільми та акторів. Це важливий ресурс.

Кожен з цих та подібного роду онлайн-ресурсів для кіноманів має свої особливості і приваблює аудиторію різноманітними функціями та підходами до збору й подання рецензій та іншої інформації про фільми. Для конкурування з ними створюваний сайт має володіти своїми унікальними перевагами та пропозиціями для користувачів. Загалом, кожен з розглянутих сайтів-конкурентів має широку

аудиторію поціновувачів і важливе місце в кіноіндустрії. Але попри це, вони одночасно достатньо схожі за своїм функціоналом для того, щоб аналізуючи їх усіх можна було провести узагальнений SWOT-аналіз онлайн-сервісів для кіноманів: IMDb, Rotten Tomatoes, Metacritic, CinemaBlend, Letterboxd, TMDB.

Сильні сторони вибраних для аналізу програм-конкурентів:

1) *обширна база даних*: сайт буде мати обширну базу даних фільмів, акторів, режисерів, оглядів і оцінок на фільми, що буде привертати користувачів та робить такий сервіс цінним ресурсом для кіноманів;

2) *розмаїття контенту*: окрім оглядів та оцінок, сайт може пропонувати своїм користувачам різноманітні статті, новини та добірки фільмів, що привертатиме та утримуватиме аудиторію;

3) *користувацький досвід*: User-friendly інтерфейс сайту може забезпечити зручність використання сервісу для користувача будь-якого віку;

4) *спільнота користувачів*: наявність активної спільноти користувачів, що обговорюватимуть фільми, ділитимуться своїми враженнями та рецензіями на різні фільми, даватимуть рекомендації іншим користувачам, буде також сприяти привертанню та утриманню аудиторії.

Слабкі сторони:

1) *недостатність модерації*: сайт може страждати від недостатньої модерації контенту, що може призвести до появи неякісних або образливих рецензій, що в свою чергу може негативно вплинути на репутацію самого онлайн-сервісу;

2) *обмежені ресурси* можуть перешкоджати швидкому розвитку сайту та своєчасному оновленню контенту, додаванню нових функцій та завчасної технічної підтримки;

3) *конкуренція*: наявність великих сайтів зі схожим контентом, як-от: IMDB або Rotten Tomatos, може зменшити активний трафік на сайті;

4) *недостатність оригінального контенту*: можливо сайт буде страждати від недостатньої кількості оригінального контенту, що зможе зменшити його привабливість для користувачів та зменшити його конкурентоспроможність.

Можливості:

1) *розвиток мобільного застосунку*: створення мобільного застосунку може привернути нову аудиторію і забезпечити більш зручний доступ до онлайн-сервісу для користувачів;

2) *розширення партнерських відносин*: встановлення партнерських відносин з великими кінокомпаніями або платформами для стрімінгу може забезпечити доступ до ексклюзивного контенту і підвищити привабливість сайту для користувачів;

3) *розвиток персоналізованих рекомендацій*: можливість впровадження складних алгоритмів та машинного навчання для надання персоналізованих рекомендацій для кожного користувача може покращити досвід використання сайту та утримання аудиторії;

4) *міжнародне розширення*: можливість розширити географію аудиторії шляхом надання контенту різними мовами і за допомогою адаптації до культурних особливостей різних країн.

Загрози:

1) *конкуренція*: зростання конкуренції з іншими подібними сайтами може призвести до втрати активних користувачів;

2) *зміни в законодавстві*, що стосуються авторських прав або реклами в інтернеті, можуть вплинути на бізнес-модель сайту і збільшити витрати на дотримання норм права;

3) *технічні проблеми*: можливість технічних збоїв та хакерських атак.

Загалом SWOT-аналіз є важливим інструментом, який допомагає оцінити зовнішні та внутрішні фактори, що впливають або будуть впливати на проєкт. За його допомогою можна оцінити поточний стан проєкту та виявити ключові аспекти, які слід враховувати при розробці. Проведений SWOT-аналіз дозволив виявити слабкі та сильні сторони програмного забезпечення (ПЗ), що дозволяє вести розробку, яка базуватиметься на можливостях та проблемах ПЗ. Загалом, такий аналіз є важливим інструментом для розробки ПЗ, що допоможе виявити

ефективний напрямок розвитку проєкту, спрямований на досягнення поставлених цілей.

Попри можливі виклики, подібний онлайн-сервіс для кіноманів має великий потенціал для успіху. Враховуючи свої сильні сторони та можливості, а також реагуючи на слабкі сторони та загрози, сайт може стати цінним ресурсом для фанатів фільмів та рецензентів.

1.3 Розробка функціональних вимог проєкту

Створення функціональних вимог до застосунку є важливою частиною його розробки. Такі вимоги допомагають краще вибрати напрямок, в якому проєкт буде розвиватися, дозволяють краще зрозуміти потреби користувача та робить сам процес розробки програмного забезпечення більш ефективним.

Онлайн-сервіс для кіноманів має великий потенціал для розширення свого функціоналу. Тому для розробки MVP потрібно розуміти, які саме функції будуть найважливішими для мінімальної життєздатності проєкту.

Основні функціональні вимоги створюваного сайту:

- Авторизація:
    - можливість зареєструвати новий акаунт, за допомогою email, імені користувача та паролю;
    - можливість зайти у вже існуючий акаунт за допомогою email та паролю;
- Взаємодія користувачів із фільмами:
    - отримання базової інформації про фільм на його сторінці;
    - можливість пошуку потрібних користувачеві фільмів за допомогою фільтрації за жанрами, режисерами, акторами, роком випуску тощо;
    - формування рецензій та виставляння оцінок до фільмів.
    - Додавання нового фільму до каталогу існуючих.

Такий набір базових функцій має бути достатнім для реалізації MVP подібного продукту, для подальшого його тестування та є базою для подальшого, можливого розширення проєкту новим функціоналом.

1.4 Вибір засобів розробки

C# – це об’єктно-орієнтована, кросплатформна мова програмування з відкритим вихідним кодом, що використовується для розробки програмного забезпечення на платформі .NET. Програми, написані мовою C#, можуть запускатися на різних девайсах та виконувати різний функціонал: від розробки для інтернету речей (IoT) до хмарних технологій. За допомогою цієї мови програмування можна створювати застосунки для телефонів, десктопу та серверів [5].

C# є популярною мовою програмування, що забезпечує розробника доступом до великої кількості навчального матеріалу, а також до готових рішень різноманітних проблем, що можуть виникати при розробці проєкту засобами C#. За даними сайту Dou.ua у рейтингу мов програмування 2024 року мова C# посідає п’яте місце [6].

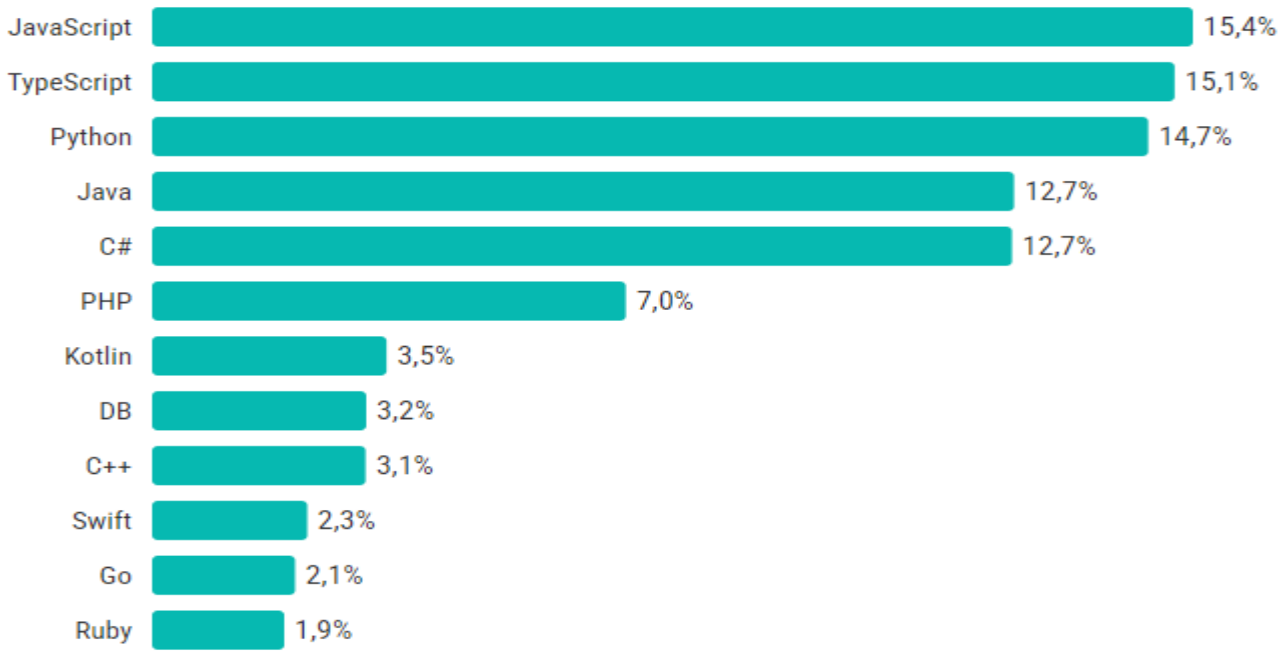

Рисунок 1.1 – Рейтинг мов програмування за 2024 рік

C# має численні корисні функції для розробки [5]:

– прибиральник сміття: автоматично очищує пам’ять від об’єктів, які вже не використовуються;

– оброблення винятків: C# має структурований, розширюваний підхід до оброблення помилок та пошуку рішень до них;

– лямбда-вирази: такі вирази підтримують прийоми, що використовуються у функціональному програмуванні;

– синтаксис LINQ: допомагає при роботі з даними з різних джерел;

– асинхронні операції: потрібні при роботі з мережевими запитами, введенням чи виведенням даних та операціями, що можуть використовувати багато часу на виконання.

1.4.1 Інструменти для розробки back-end частини застосунку

Back-end частина є однією із суттєвих складових будь-якого вебзастосунку, вона відповідає за обробку запитів користувачів, взаємодією з базою даних та загалом забезпечує необхідний функціонал для реалізації бізнес-логіки застосунку. Для розробки онлайн-сервісу для кіноманів був вибраний ASP.NET CORE.

ASP.NET CORE – це безкоштовний, кросплатформний фреймворк з відкритим вихідним кодом для розробки вебпрограм, хмарних застосунків, застосунків для інтернету речей, серверної частини для мобільних застосунків тощо [7].

Доволі часто можна сплутати ASP.NET CORE з ASP.NET, хоча це, насамперед, повністю переписані засоби розробки для роботи з .NET CORE. Цей фреймворк швидкий, його можна легко налаштовувати під себе, має модульну структуру та підтримку крос-платформи. Він добре підходить для розробки вебзастосунків, застосунків для мобільних телефонів та розробки програмного забезпечення для інтернету речей.

ASP.NET CORE має такий вбудований функціонал для розробки [8]:

– фреймворк для логів з можливістю розширювання;

– новий, швидкий та крос-платормений вебсервер Kestrel. Тож вебзастосунки тепер можуть запускатися без IIS, Apache та Nginx;

– підтримка різних платформ для хостингу;

– модульність фреймворку, що дозволяє кожному розробнику використовувати той функціонал, що йому потрібен;

– наявність файлу appsettings.json дозволяє зберігати кастомні налаштування у зручному форматі json;

– підтримка роботи з асинхронними функціями.

1.4.2 Інструменти для розробки front-end частини застосунку

Front-end також є невід'ємною частиною будь-якого сучасного вебзастосунку. Він не тільки відповідає за інтерфейс, та має бути інтуїтивно зрозумілим більшості користувачів, але також займає вагоме місце у процесі оптимізації застосунку, бо саме він відповідає за відправку запитів до контролерів. Для реалізації front-end частини проєкту був вибраний Bootstrap.

Bootstrap – це потужний фреймворк для розробки вебінтерфейсів, який забезпечує швидкий та ефективний спосіб створення сучасних вебзастосунків. Однією з ключових переваг Bootstrap є його готові компоненти: кнопки, форми, навігаційні панелі, таблиці, картки та багато іншого. Ці компоненти можна легко використовувати та налаштовувати, що дозволяє розробникам швидко створювати стильні та функціональні інтерфейси без потреби великої кількості власного CSS та JavaScript коду.

Один з найбільш важливих аспектів Bootstrap – це його активна спільнота та широка екосистема. У фреймворку існують безліч розширень, тем, шаблонів та інших ресурсів, що сприяють подальшому розвитку та використанню Bootstrap для розробки front-end частини у різних проєктах. В цілому, Bootstrap є потужним інструментом для швидкого та ефективного створення сучасних вебінтерфейсів.

1.5 Маркетингові інструменти в створенні та просуванні програмного продукту

Маркетинг розробки програмного забезпечення має поєднувати різні канали. На відміну від традиційних маркетингових підходів, цифровий маркетинг має три ключові фактори, які показують, чому цифровий маркетинг є кращим варіантом для розробки програмного забезпечення, ніж традиційний маркетинг. Це широке

охоплення клієнтів, навчання замість нав'язливих продажів та швидкий маркетинговий цикл. На відміну від традиційного маркетингу, цифровий маркетинг дозволяє генерувати сотні продажів та створювати нових потенційних клієнтів за лічені години після запуску маркетингової кампанії [9].

Для ефективного маркетингу онлайн-сервісу з рецензіями на фільми можна використовувати різноманітні інструменти та стратегії з маркетингу. По-перше, варто визначитися з цільовою аудиторією такого сайту. Вона може бути дуже різнорідною, бо складається з фанатів різних фільмів та телешоу, які також розраховані на різні цільові аудиторії. Потрібно розуміти потреби та інтереси своїх користувачів, що допоможе створити ефективну стратегію маркетингу.

Активна присутність сервісу у соціальних мережах дозволить сайту залучити нову аудиторію та підтримувати зв'язок з користувачами. Також з'являється можливість публікувати новини, різноманітні анонси оновлень на сайті та інші цікаві матеріали. Активність у соціальних медіа також сприятиме взаємодії з аудиторією та обміну враженнями.

Для утримання активної аудиторії можна використовувати електрону пошту для розсилки новин, акцій та іншої цікавої інформації. Створення цікавого контенту на сайті, наприклад, статей про кіно, інтерв'ю з різними режисерами та акторами, списки новинок та рекомендованих фільмів тощо, може привертати нову аудиторію та утримувати активних користувачів на сайті [10].

Укладання партнерських угод зі сторонніми брендами, такими як кінотеатри, стрімінгові платформи Netflix, Amazon Prime або Disney+, виробники фільмів тощо, може допомогти долучити нову аудиторію. Використання рекламних кампаній в інтернеті, наприклад контекстну рекламу, рекламу у соціальних медіа або банери на інших сайтах, також сприятимуть зростанню активної аудиторії онлайн-сервісу.

Також важливо приділяти достатньо уваги SEO (оптимізація пошукових систем). Така оптимізація сайту допоможе покращити його видимість у результатах пошукових систем і привернути до себе більше органічного трафіку.

Не менш важливим є аналіз активної стратегії маркетингу та на його основі вносити відповідні зміни до стратегії для поліпшення результатів.

1.6 Висновки до розділу 1

Під час аналізу предметної області були визначені ключові особливості розробки онлайн-сервісів для кіноманів. Проведений аналіз програмних конкурентів у формі SWOT-аналізу показав, що створення онлайн-сервісу для кіноманів є виправданим і перспективним.

На основі аналізу предметної області та програмних конкурентів були сформульовані функціональні вимоги до такого онлайн-сервісу. Для розробки back-end частини вебзастосунку був вибраний фреймворк ASP.NET Core, а для front-end частини – бібліотека Bootstrap.

Крім того, розроблена маркетингова стратегія для просування та розвитку вебзастосунку.

2 РОЗРОБКА АРХІТЕКТУРИ ВЕБСАЙТУ ДЛЯ КІНОМАНІВ

2.1 Middleware

Middleware – це така концепція, яка використовується майже у сучасних фреймворках для розробки вебзастосунків, зокрема у ASP.NET CORE. Вона дозволяє структурувати код самого застосунку у вигляді ланцюга подій або обробників, що опрацьовують запити та відправляють відповіді.

У ASP.NET CORE middleware подані класами, що реалізуються через інтерфейс `IMiddleware`, або функціональними компонентами, які отримують на вході `RequestDelegate`. Кожен middleware може обробляти `HttpContext`, який містить дані про поточний HTTP-запит та може модифікувати та доповнювати його перед тим, як він дістанеться до наступного middleware [11].

```
app.UseExceptionHandler("/Error");
app.UseHsts();
app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();
app.UseCors();
app.UseAuthentication();
app.UseAuthorization();
app.UseSession();
app.MapControllers();
//add your custom middlewares
app.Run(); proper order to use the middleware.
```

Рисунок 2.1 – Порядок виконання Middleware у ASP.NET CORE

Middleware може використовуватись для логування деталей про запити та відповіді, що може допомогти при проведені аналізу роботи вебзастосунку. Також його використовують для перевірки прав доступу до різних ресурсів у різних користувачів, для проведення процедури аутентифікації перед передачею запитів до контролерів.

На рис. 2.2 показано загальну схему роботи middleware засобами ASP.NET CORE [12].

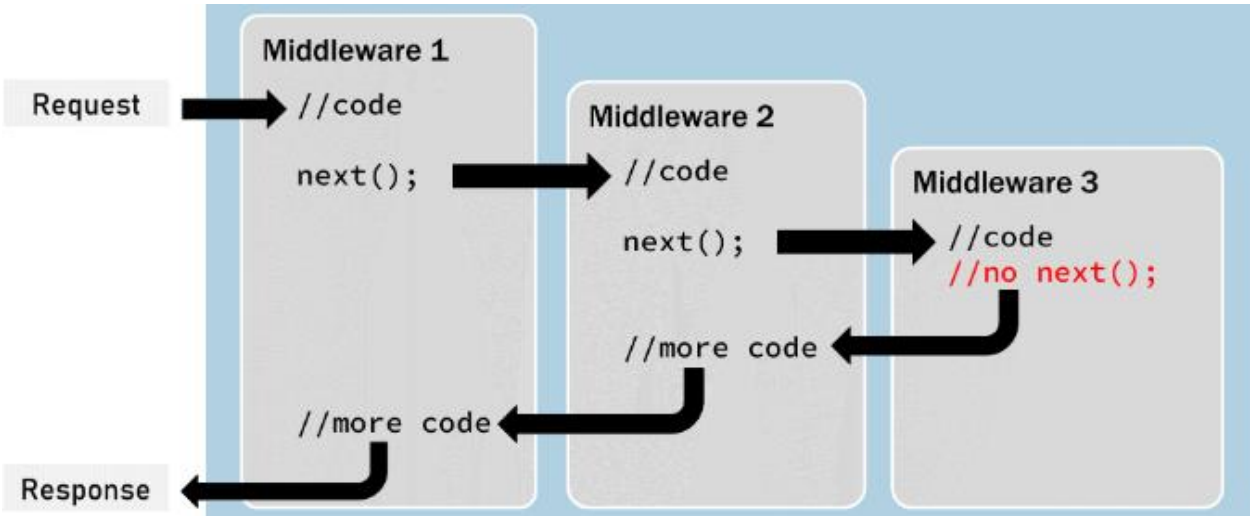

Рисунок 2.2 – Схема виконання middleware у ASP.NET CORE

Middleware дозволяє розділити логіку оброблення запитів на невеликі, прості компоненти, що у свою чергу дозволяє зберегти програмний код самого застосунку організованим та чистим. Це допомагає покращити легкість читання, масштабованість та зручність подальшої розробки.

2.2 Model-View-Controller

Model-View-Controller (MVC) – архітектурна модель, яка дозволяє розділити логіку розробки застосунку на три основні компоненти: модель, подання та контролер. Вона сприяє повторному використанню компонентів коду та дозволяє використовувати більш модульний підхід до розробки програмного забезпечення [13].

Архітектурна модель MVC визначає, що застосунок складається з трьох основних компонентів: моделі даних, репрезентації інформації та контрою подання цієї інформації. Такий розподіл дозволяє при розробці застосунку зручніше та ефективніше вносити зміни у окремі його компоненти, бо зміна одного не вимагає від розробника вносити зміни в інший (рис. 2.3) [14].

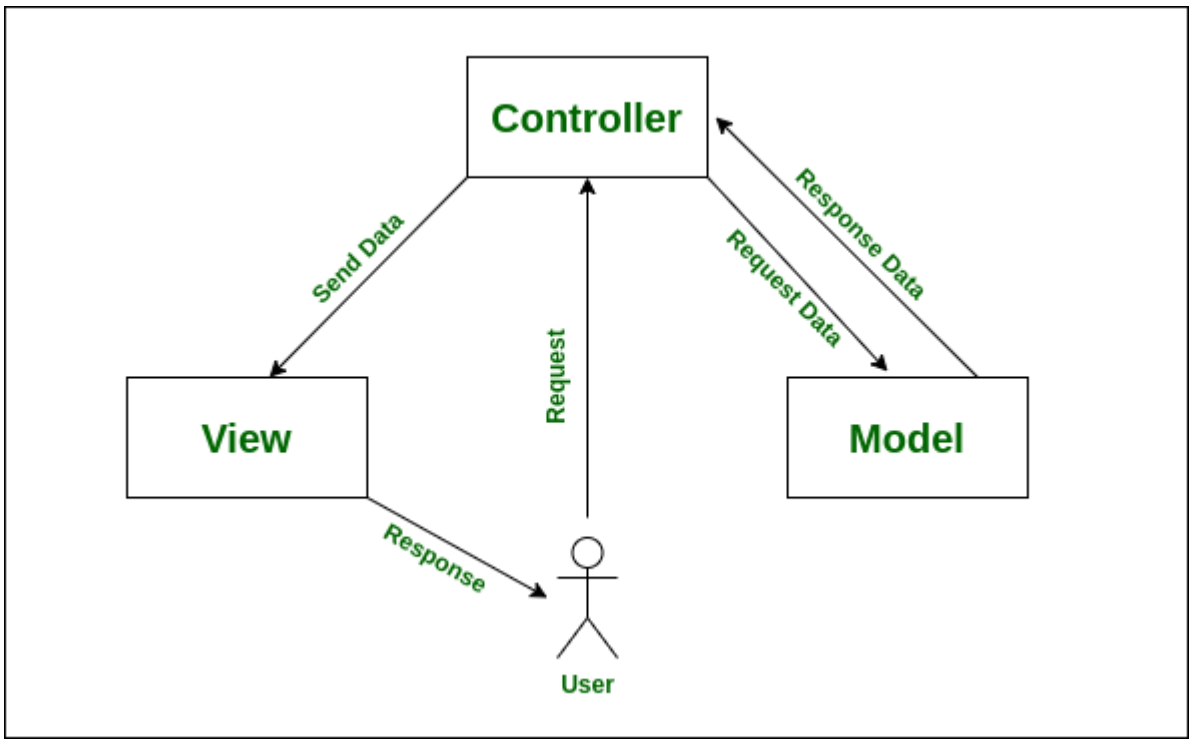

Рисунок 2.3 – Схема роботи архітектурної моделі MVC

Компонентами такої моделі є:

– model – модель даних, якає сукупністю даних та бізнес-логіки застосунку. цей компонент відповідає за управління цими даними, обробку логіки та відповіді на запити від інших компонентів;

– view – подання даних отриманих від моделі та надсилання введених даних користувача до контролера. Цей компонент доволі пасивний і не контактує напряму з даними, він лише отримує ці дані від моделі;

– controller – контролер є мостом між model та view. Він обробляє отримані від view дані користувача та взаємодіє з model на основі отриманих вказівок.

Одним із недоліків архітектурної моделі MVC є можливість створення надмірно складної структури застосунку, що у свою чергу може призвести до розробки складного коду, який може призвести до втрати швидкості виконання ПЗ. Інколи розробники, використовуючи такий підхід до створювання програм, можуть перевантажити код застосунку додаючи зайві рівні абстракції, а це може призвести до надмірного збільшення кількості файлів у проєкті. Усі ці фактори призводять до труднощів у навігації та обслуговування чи рефакторингу коду.

Тож шаблон MVC дозволяє розділити логіку застосунку на компоненти, що полегшує розробку, тестування та підтримку написаного коду. Це покращує структуру застосунку, забезпечує гнучкість при його розробці. Але модель MVC потрібно використовувати з обережністю, щоб не стикнутись із проблемами у майбутньому, при підтримці програмного забезпечення.

2.3 Розробка архітектури бази даних

База даних є чи не найважливішою частиною застосунку, бо саме вона відповідає за зберігання, організацію та доступ до різних даних. У MVC саме model взаємодіє напряму з базою даних, запитуючи у неї різні дані, та вносячи у неї різні зміни.

Для розробки бази даних для створюваного у роботі онлайн-сервісу основними сутностями є: користувач, рецензії, фільм. Для кожної з них визначено атрибути:

- Користувач:
  - Ідентифікаційний номер користувача (UserId);
  - Email користувача (UserEmail);
  - Ім’я користувача (UserName);
  - Пароль користувача (UserPassword);
- Рецензія:
  - Ідентифікаційний номер рецензії (ReviewId);
  - Ідентифікаційний номер користувача, що залишив рецензію (Id_User);
  - Ідентифікаційний номер фільму, на який користувач залишив цю рецензію (Id_Film);
  - Загальна оцінка фільму за 10-ти бальною шкалою (Rating);
  - Текст рецензії до фільму (ReviewText);
- Фільм:
  - Ідентифікаційний номер фільму (FilmId);
  - Назва фільму (FilmName);

- Постер (FilmPoster);
- Рік випуску (FilmYear);
- Рейтинг фільму (FilmRating);
- Дата прем’єри фільму (FilmDate);
- Режисер фільму (FilmDirector);
- Актори фільму (FilmActors);
- Студія, яка виробила фільм (FilmStudio);
- Жанр або жанри фільму (FilmGenre);
- Тривалість фільму (FilmDuration);
- Текстовий опис фільму (FilmDescription);
- Країна виробник фільму (FilmCountry);

Для створення ER-діаграми цієї бази даних потрібно визначитись зі зв’язками між її сутностями.

Зв’язок між сутностями *Користувач – Рецензія* буде виглядати як 1:N, тобто один до багатьох. Тобто, один користувач може писати більш ніж одну рецензію (рис. 2.4).

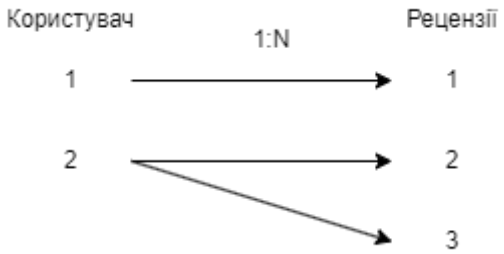

Рисунок 2.4 – Схема зв’язку між сутностями Користувач – Рецензії

Зв’язок між сутностями *Фільм – Рецензія* буде також мати вигляд 1:N, бо один фільм може мати більш ніж одну рецензію на себе (рис. 2.5).

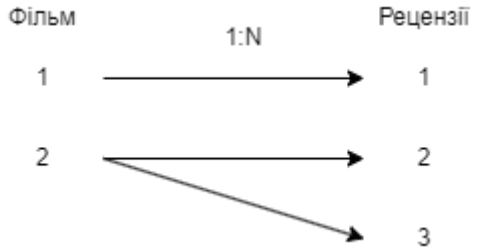

Рисунок 2.5 – Схема зв’язку між сутностями Фільм – Рецензії

Далі наведено entity-relationship (ER) діаграму, яка демонструє усі сутності та відносини між ними (рис. 2.6).

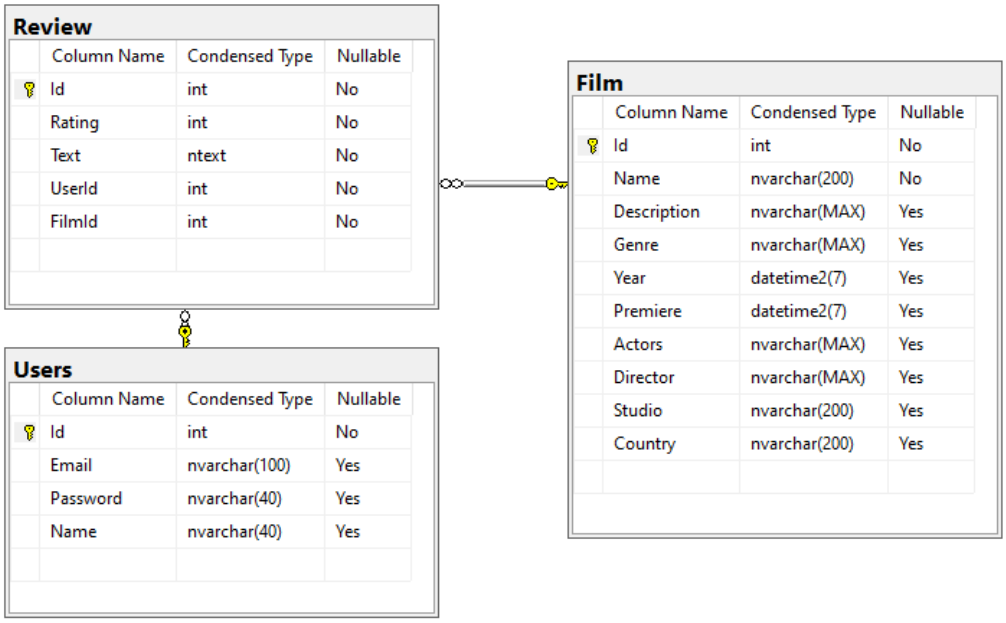

Рисунок 2.6 – Entity-Relationship model

У табл. 2.1-2.3 наведені атрибути кожної сутності розробленої бази даних, а також тип даних кожного атрибуту та короткий опис.

Таблиця 2.1 – User

Атрибут	Тип даних атрибуту	Опис
Id	int	Первинний ключ
Email	nvarchar	Email користувача
Password	nvarchar	Пароль користувача
Name	nvarchar	Ім’я користувача

Таблиця 2.2 – Film

Атрибут	Тип даних атрибуту	Опис
Id	int	Первинний ключ
Name	nvarchar	Назва
Description	navrchar	Опис
Genre	nvarchar	Жанр\жанри
Year	int	Рік випуску
Premiere	datetime	Дата прем’єри
Actors	nvarchar	Актори
Director	nvarchar	Режисер
Studio	nvarchar	Студія
Country	nvarcahr	Країна виробник

Таблиця 2.3 – Review

Атрибут	Тип даних атрибуту	Опис
Id	int	Первинний ключ
Rating	int	Загальна оцінка фільму
Text	ntext	Текст рецензії
UserId	int	Вторинний ключ користувача
FilmId	int	Вторинний ключ фільму

2.4 Моделювання функціоналу системи

Use-case діаграми є потужним інструментом для аналізу та моделювання функціоналу системи з точки зору її користувача. Вони допомагають зрозуміти який функціонал будуть використовувати користувачі та як він між собою пов’язаний.

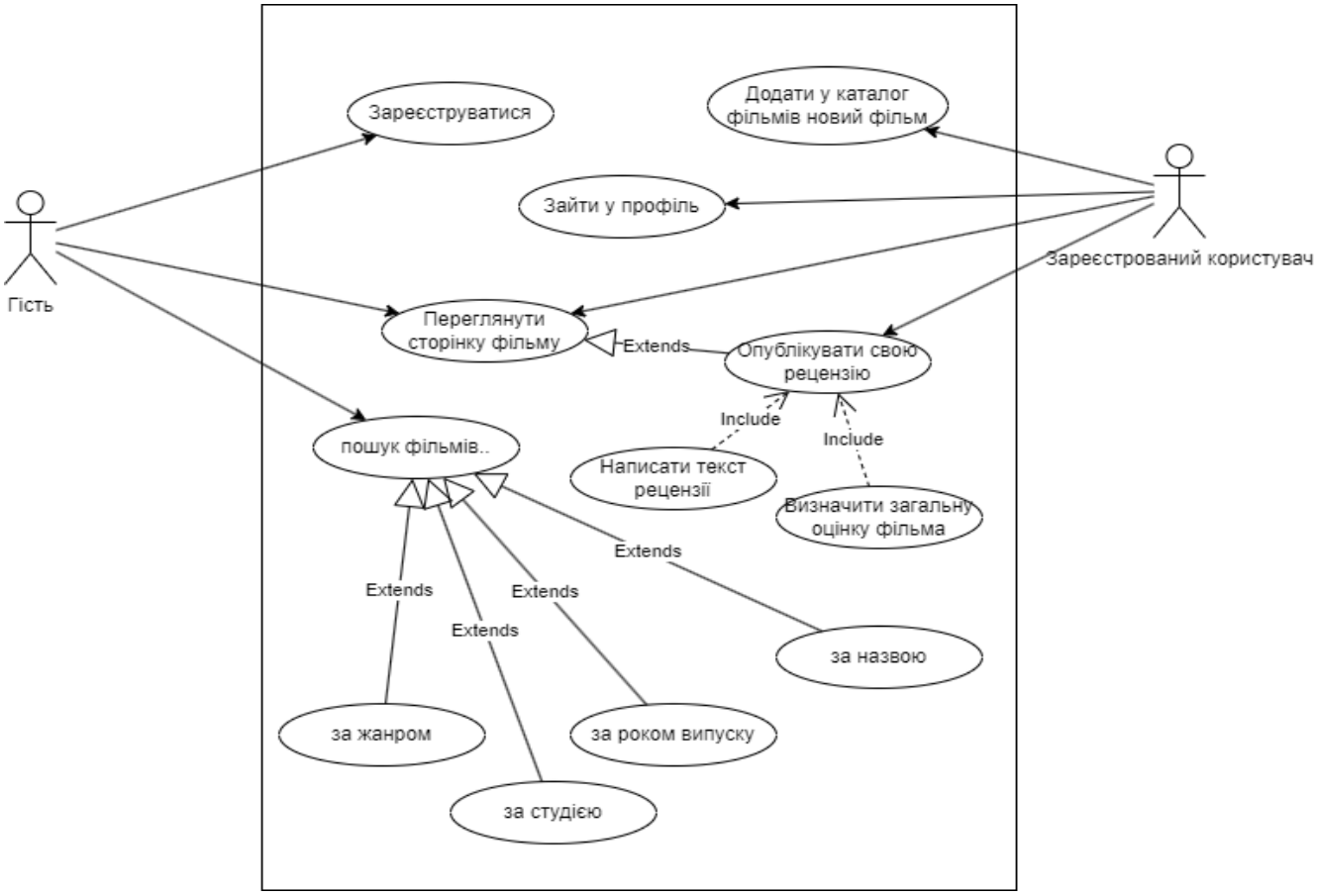

Рисунок 2.7 – Use-case діаграма онлайн-сервісу для кіноманів

На рис. 2.7 зображена Use-case діаграма онлайн-сервісу для кіноманів. Діаграма описує можливості користувачів, в залежності від їх статусу в системі. На ній знаходиться два актора: Гість, та зареєстрований користувач. У гостя є можливість лише переглянути інформацію про фільми на сайті, застосувати форму пошуку та зареєструватися.

Зареєстрований користувач має більший доступ до функціоналу вебзастосунку. Він має можливість створити рецензію на фільм, вказавши його загальну оцінку та написавши до неї тест. Також у зареєстрованого користувача з'являється можливість до вже існуючих на сайті фільмів додати новий, вказавши у спеціальній формі усю необхідну для цього інформацію.

2.5 Висновки до розділу 2

Спроєктована у роботі архітектура сайту використовує концепції middleware та архітектурну схему MVC (Model-View-Controller). Була розроблена реляційна база даних, яка зберігає інформацію про користувачів, фільми та рецензії. Проведене моделювання функціоналу системи дозволило створити діаграму прецедентів (use-case діаграму) для застосунку.

Архітектура розроблена з урахуванням можливості подальшого розширення функціоналу, що дозволить легко додавати нові можливості та інтеграції. Також передбачено використання засобів безпеки у формі автентифікації та авторизації користувачів для захисту персональних даних і забезпечення безпечного доступу до ресурсів сайту.

3 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка програмного забезпечення

Онлайн-сервіс для кіноманів має декілька складових, а саме: контролери, подання, моделі даних та міграції (рис. 3.1).

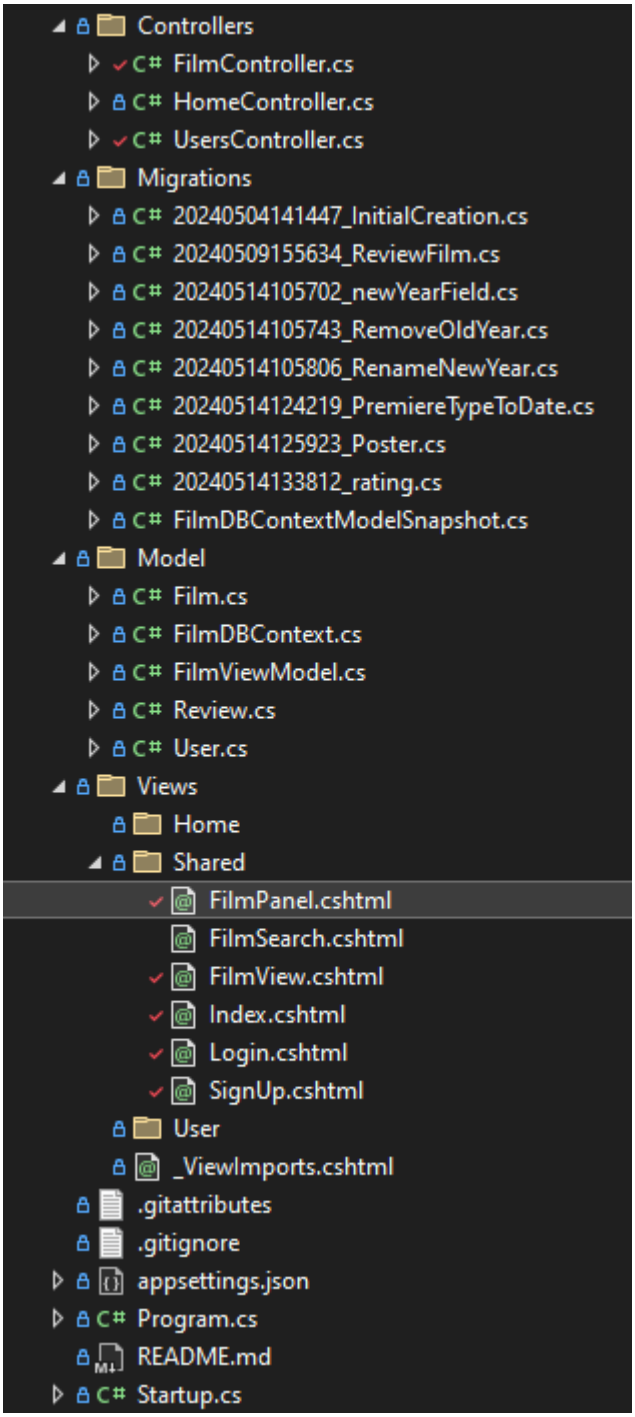

Рисунок 3.1 – Каталог файлів проєкту

Міграції у ASP.NET CORE є потужним інструментом для маніпуляцій з базою даних. Вони генеруються автоматично на основі коду C# та дозволяють за його допомогою змінювати базу даних проєкту. Генерація проходить на основі наявних у проєкті моделей. Міграції є двосторонніми, що означає можливість повертатися до більш ранніх ітерацій бази даних. Проте використовувати такий функціонал потрібно обережно, бо це може призвести до втрати важливих даних. Розберемо код міграцій на основі файлу `20240504141447_InitialCreation.cs`.

У класі є два основних метода:

```
protected override void Up(MigrationBuilder migrationBuilder)
protected override void Down(MigrationBuilder migrationBuilder)
```

Метод `Up` проводить саму міграцію, тоді як `Down` повертає базу даних до того стану, в якому була проведена ця міграція. У нашому випадку `Down` видалить таблицю `Users`, позаяк це перша її ітерація, і ця міграція її створює.

Програмний код методу `Down`:

```
protected override void Down(MigrationBuilder migrationBuilder)
{
        migrationBuilder.DropTable(
            name: "Users");
}
```

Метод `Up`, у свою чергу, створює таблицю `Users` та привласнює їй атрибути, зазначені у файлі `User.cs` у каталозі `Model`.

Програмний код методу `Up` (рис. 3.2)

Створена таблиця `Users` містить поля: ідентифікатор (Id), ім’я користувача, поштова скринька. Поле Id є ключовим (`PrimaryKey`).

Далі проведений аналіз файлу моделі на прикладі `Review.cs`. Цей файл визначає модель рецензій до фільмів. У цієї моделі є `Many-to-One` відносини з `User` та `Film`. Тож для початку визначаємо `PrimaryKey`:

```
[Key]
public int Id { get; set; }
```

```
protected override void Up(MigrationBuilder migrationBuilder)
{
    migrationBuilder.CreateTable(
        name: "Users",
        columns: table => new
        {
            Id = table.Column<int>(type: "int", nullable: false)
                .Annotation("SqlServer:Identity", "1, 1"),
            Email = table.Column<string>(type: "nvarchar(max)", nullable: false),
            Password = table.Column<string>(type: "nvarchar(max)", nullable: false)
        },
        constraints: table =>
        {
            table.PrimaryKey("PK_Users", x => x.Id);
        });
}
```

Рисунок 3.2 – Функція Up в міграції

Далі потрібно присвоїти їх основні параметри:

```
[Required]
public int Rating { get; set; }
[Required, Column(TypeName = "ntext")]
public string Text { get; set; }
```

Обидва з них мають атрибути `Required`, що означає неможливість створення рецензії без цих двох параметрів. Також Text має атрибут, визначаючий його тип даних у базі даних як `ntext`. Це текст із можливістю підтримки юнікоду[13].

Далі визначаються два `ForeignKey`, один на `User`, другий на `Film`:

```
public int UserId { get; set; } //foreign key to Users
public int FilmId { get; set; } //foreign key to Films
```

І після цього у моделі з’являється два поля, що представляють собою об’єкти, на які посилається рецензія:

```
[BindNever]
// Required reference navigation to Users
public User? User { get; set; }
[BindNever]
// Required reference navigation to Films
public Film? Film { get; set; }
```

Атрибути `BindNever` потрібні для коректної подальшої перевірки валідності моделі у контролерах.

Увесь код класу `Review.cs`:

```
namespace mdb_project.Model
{
    public class Review
    {
        [Key]
        public int Id { get; set; }
        [Required]
        public int Rating { get; set; }
        [Required, Column(TypeName = "ntext")]
        public string Text { get; set; }
        public int UserId { get; set; } //foreign key to Users
        public int FilmId { get; set; } //foreign key to Films
        [BindNever]
        // Required reference navigation to Users
        public User? User { get; set; }
        [BindNever]
        // Required reference navigation to Films
        public Film? Film { get; set; }
    }
}
```

Далі розберемо роботу контролеру на прикладі `FilmController.cs`

Для коректної роботи контролера до нього підключається база даних `SQL Server`, в якої зберігаються всі дані проєкту. До проєкта база даних підключається за допомогою спеціальної `ConnectionString`, що у випадку цього застосунку виглядає як:

```
builder.Services.AddDbContext<FilmDBContext>(options =>
{
    options.UseSqlServer("Server=DESKTOP-
                          PDNGHNS\\SQLEXPRESS;Database=FilmDB;" +
        "Trusted_Connection=True;TrustServerCertificate=True");
});
```

Цей рядок містить усі необхідні налаштування для коректної взаємодії застосунку з базою даних. `FilmDBContext` у свою чергу – це модель, що

представляє собою контекст бази даних, зберігаючи у собі колекції моделей. Програмний код `FilmDBContext.cs`:

```
namespace mdb_project.Model
{
    public class FilmDBContext : DbContext
    {
        public DbSet<User> Users { get; set; }
        public DbSet<Film> Films { get; set; }
        public DbSet<Review> Reviews { get; set; }
        public FilmDBContext(DbContextOptions options) : base(options)
        {

        }
    }
}
```

Для того, щоб контролер міг взаємодіяти із базою даних, у самому контролері потрібно визначити змінну, яка буде зберігати контекст бази даних. Її визначення відбувається в конструкторі самого контролера:

```
public class FilmController : Controller
{
    private readonly FilmDBContext _context;
    public FilmController(FilmDBContext context)
    {
        _context = context;
    }
```

Розберемо роботу контролера на прикладі метода `Create`, який додає в базу даних новий фільм.

Такий метод буде мати два допоміжних атрибута, `[Route(FilmPanel/created)]` та `[HttpPost]`. Перший визначає маршрут методу, інший специфікує, що запрос має бути `Http Post`.

Метод має приймати об’єкт `film`, переданий йому через подання, що містить усю інформацію, потрібну для додання нового фільму у таблицю.

```
public IActionResult Create(Film film)
```

Далі використовуємо умовний оператор if для перевірки валідності переданих даних, чи відповідають вони потрібним типам та чи не є важливі атрибути Null. Це все можна перевірити вбудованою функцією `ModelStae.IsValid()`:

```
if (ModelState.IsValid)
```

Після цього додаємо дані у базу даних та зберігаємо зміни цими двома методами:

```
_context.Films.Add(film);
_context.SaveChanges();
```

Останньою дією буде передання повідомлення про успішне створення фільму через `ViewBag` та переадресація користувача на потрібну сторінку:

```
ViewBag.Message = "Film created";
return View("FilmPanel");
```

Увесь код функції `Create`:

```
[Route("FilmPanel/created")]
[HttpPost]
public IActionResult Create(Film film)
{
    if (ModelState.IsValid)
    {
        _context.Films.Add(film);
        _context.SaveChanges();
    }
    ViewBag.Message = "Film created";
    return View("FilmPanel");
}
```

Далі розберемо роботу метода контролера, що відповідає за отримання даних із бази даних, та їх відображення на сторінці. Для аналізу був вибраний метод `FilmView`, що відповідає за відображення даних про конкретний фільм та рецензій, написаних на нього. Він приймає змінну `id`, що відображає `id` фільму, вибраного користувачем:

```
public IActionResult FilmView(int id)
```

За допомогою вбудованої мови запитів .NET LINQ проводиться запит на дані конкретного фільму через його `id`, після чого відбувається перевірка на наявність помилкових ситуацій з незаповненими даними (`Null`).

```
 var film = _context.Films.FirstOrDefault(f => f.Id == id);
if (film == null)
{
  return NotFound();
}
```

Далі таким самим методом визначається список рецензій на цей фільм:

```
var reviews = _context.Reviews.Where(r => r.FilmId == id).ToList();
```

Після чого отримуємо запитом список `id` користувачів, які залишали рецензії:

```
var usersWhoReviewedFilmId = _context.Reviews
    .Where(r => r.FilmId == id)
    .Select(r => r.UserId)
    .Distinct()
    .ToList();
```

За допомогою цих `id` отримуємо список об’єктів користувачів, які залишали рецензії на цей фільм:

```
var users = _context.Users
    .Where(u => usersWhoReviewedFilmId.Contains(u.Id))
    .ToList();
```

Далі створюємо змінну типу FilmViewModel та записуємо у неї зібрані дані:

```
var viewModel = new FilmViewModel
{
   Film = film,
   Reviews = reviews,
   Users = users
};
```

Та у кінці повертаємо результат роботи функції:

```
return View(viewModel);
```

У методі `LogIn` контролера `Users` реалізований механізм авторизації користувача за допомогою cookies

Спочатку проводяться перевірки на правильність введених даних:

```
if (CheckUserLogin(user)) return View("Login");
if (IsValidUser(user.Email, user.Password))
```

Далі використовується механізм `Claim`, що відповідає за додавання певних параметрів до активної сесії користувача:

```
var claims = new List<Claim>
{
    new Claim(ClaimTypes.Name, name),
    new Claim(ClaimTypes.NameIdentifier, id.ToString())
};
```

Далі ці параметри записуються асинхронною функцією у активну сесію користувача:

```
 var identity = new ClaimsIdentity(claims,
CookieAuthenticationDefaults.AuthenticationScheme);
 var principal = new ClaimsPrincipal(identity);
 await
 HttpContext.SignInAsync(CookieAuthenticationDefaults.Authenticatio
 nScheme, principal);
```

У поданні використовуються Razor Pages, що дозволяє використовувати код c# в середині розмітки Html. Такі файли мають розширення `.cshtml`. Для аналізу роботи такого файлу був вибраний `Index.cshtml`.

На початку файлу імпортуємо модель даних для взаємодії із даними, які були передані через контролер, та бібліотеку для взаємодії з параметрами користувача в активній сесії:

```
@model IEnumerable<mdb_project.Model.Film>
@using System.Security.Claims
```

У панелі навігації, за допомогою умовного оператору та взаємодії із Claims регулюємо інтерфейс, доступ до якого буде мати гість та зареєстрований користувач:

```
@if (!User.Identity.IsAuthenticated)
{
    <li class="nav-item">
        <a class="nav-link" href="/login">LogIn</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="/signup">Sign Up</a>
    </li>
}
@if (User.Identity.IsAuthenticated)
{
    <li class="nav-item">
        <a class="nav-link" asp-controller="Users" asp-action="Logout">Log out</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="/FilmPanel">Film Panel</a>
    </li>
}
```

Рисунок 3.3 – Фільтрація інтерфейсу

Основний список фільмів виводимо за допомогою `foreach` циклу, використовуючи список фільмів, отриманий через ViewModel від контролеру:

```
@foreach (var film in Model)
{
    <a asp-controller="Film" asp-action="FilmView" asp-route-id=@film.Id>
        <div class="col">
            <div class="card">
    <img src=@film.Poster class="card-img-top"
     height="400" alt="no poster :(">
                <div class="card-body">
                    <h5 class="card-title">@film.Name</h5>
                    <p class="card-text">@film.Description</p>
                </div>
            </div>
        </div>
    </a>
}
```

3.2 Функціональні можливості вебзастосунку

Розроблений у роботі онлайн-сервіс для кіноманів має реалізацію повного функціоналу для такого застосунку.

Головний екран сайту має навігаційну панель, що містить посилання на головний екран, на форму реєстрації та авторизації, а також панель для пошуку фільмів за назвою. На головній сторінці вебзастосунку є перелік наявних на сайті фільмів (рис. 3.4).

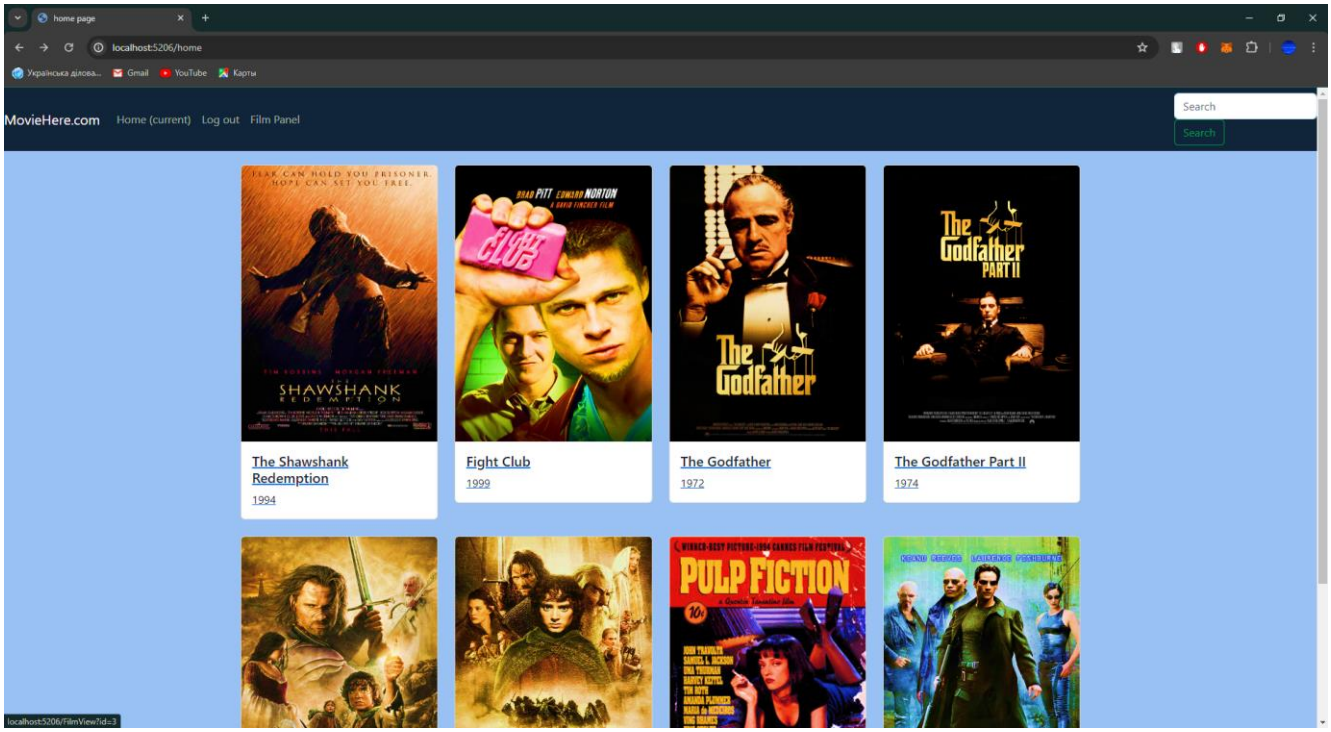

Рисунок 3.4 – Головна сторінка застосунку

При виборі конкретного фільму здійснюється перехід на сторінку з детальною інформацією про нього: жанр, актори, режисер тощо. Також на цій сторінці можна побачити середню оцінку фільму серед рецензентів та прочитати самі відгуки (рис. 3.5, 3.6).

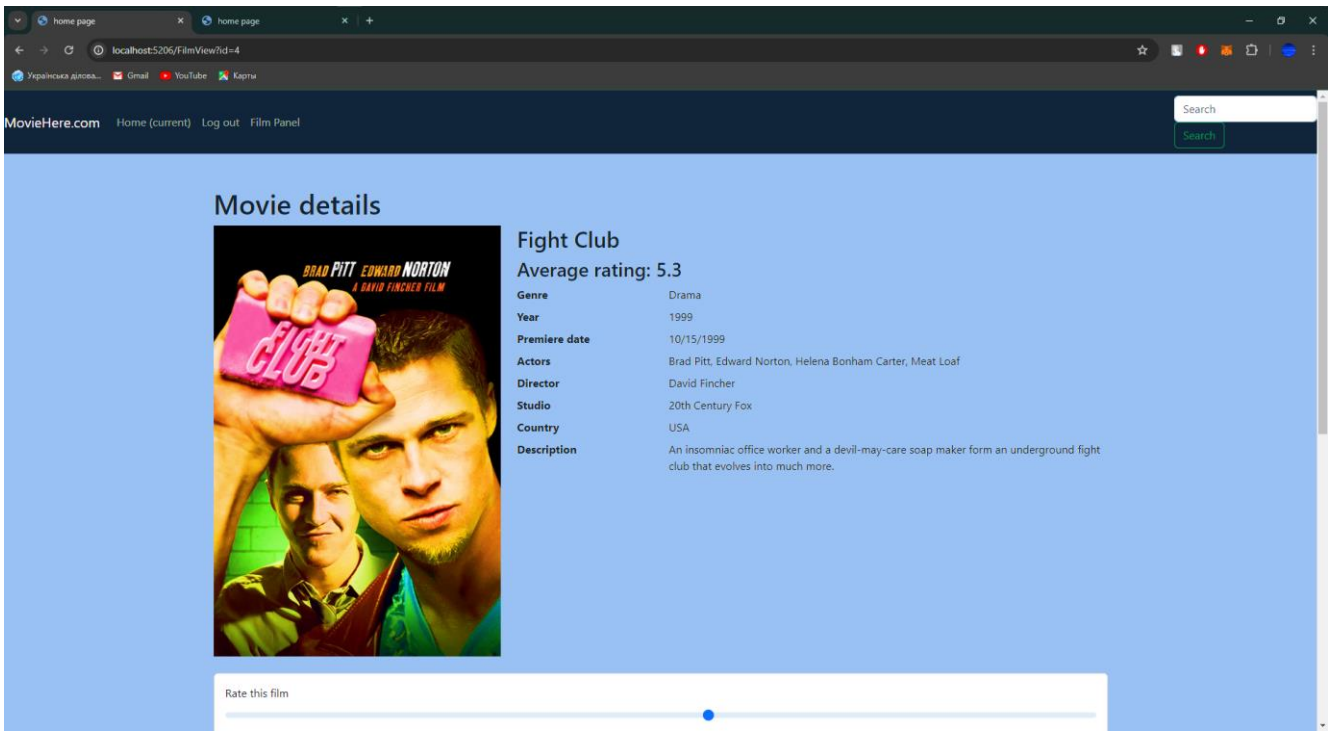

Рисунок 3.5 – Сторінка з інформацією про фільм

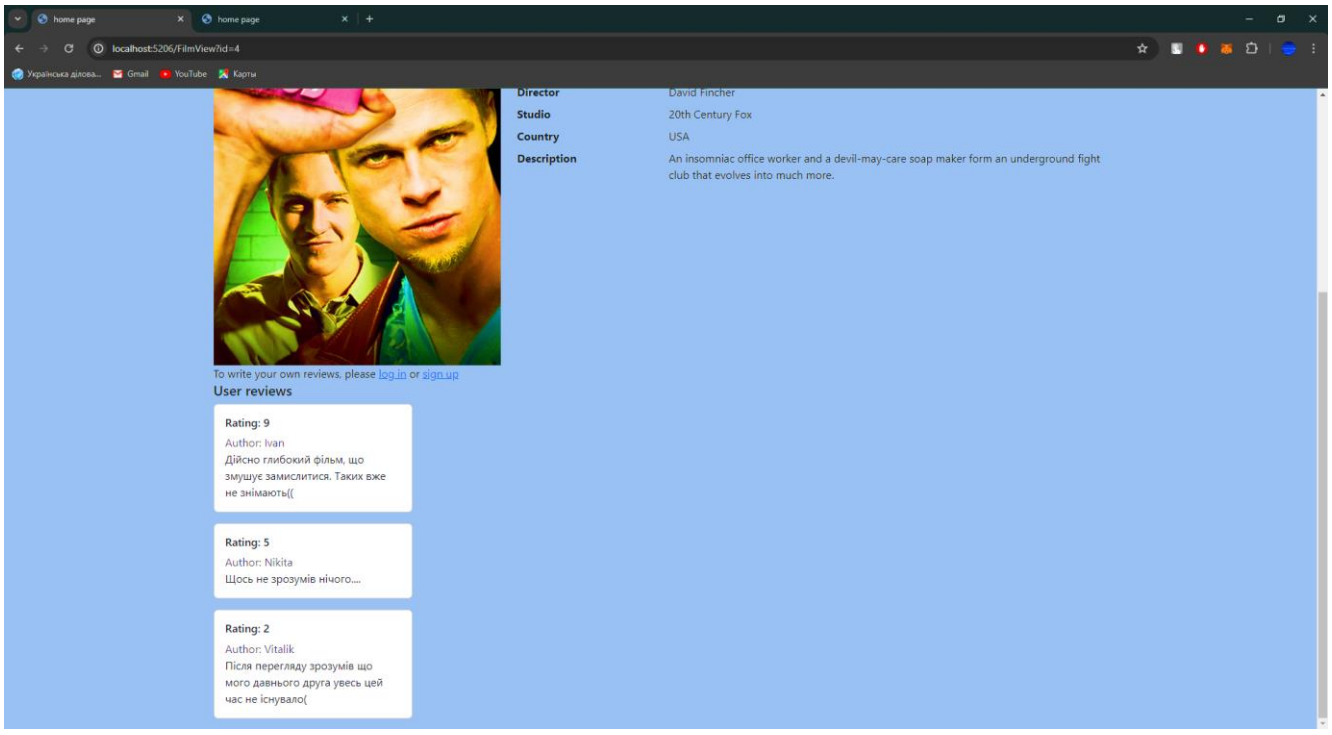

Рисунок 3.6 – Продовження сторінки з інформацією про фільм

Для того, щоб залишити рецензію, потрібно авторизуватися. Якщо аккаунту користувач ще не має, то треба зареєструватися. Це можна зробити або скориставшись навігаційною панеллю вгорі сторінки, або натиснувши кнопки авторизації чи реєстрації на тому місці, де буде форма для написання рецензії.

Сторінка реєстрації містить базову форму для введення даних, а саме: ім’я користувача, його поштова скринька та пароль. Також є кнопка для відправки цієї форми (рис. 3.7). Якщо користувач не введе якісь дані, наприклад, не вкаже ім’я користувача, то застосунок не відправить форму на сервер та напише, які конкретно поля не заповнив користувач (рис. 3.8). Теж саме буде, якщо користувач буде намагатися використати поштову скриньку, яка вже кимось використовується(рис. 3.9). Також неправильний формат введеної пошти відстежується і повідомляється користувачу (рис. 3.10).

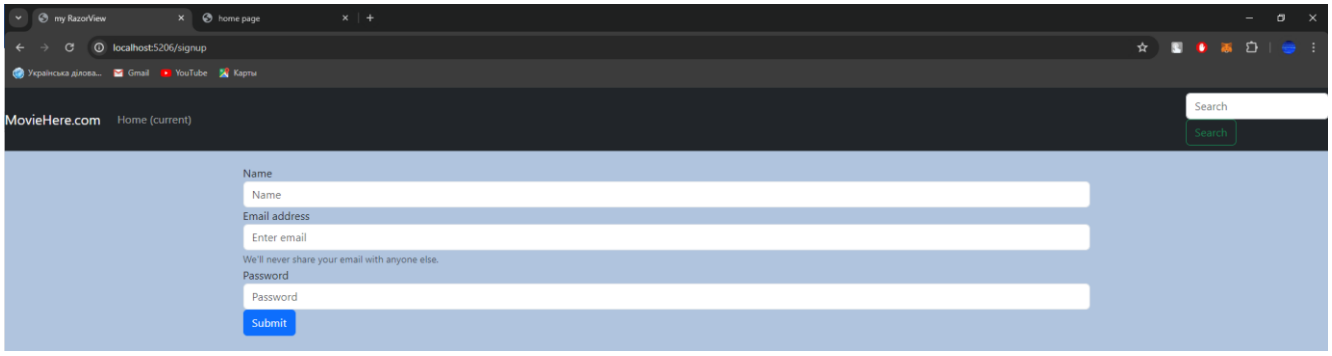

Рисунок 3.7 – Форма реєстрації

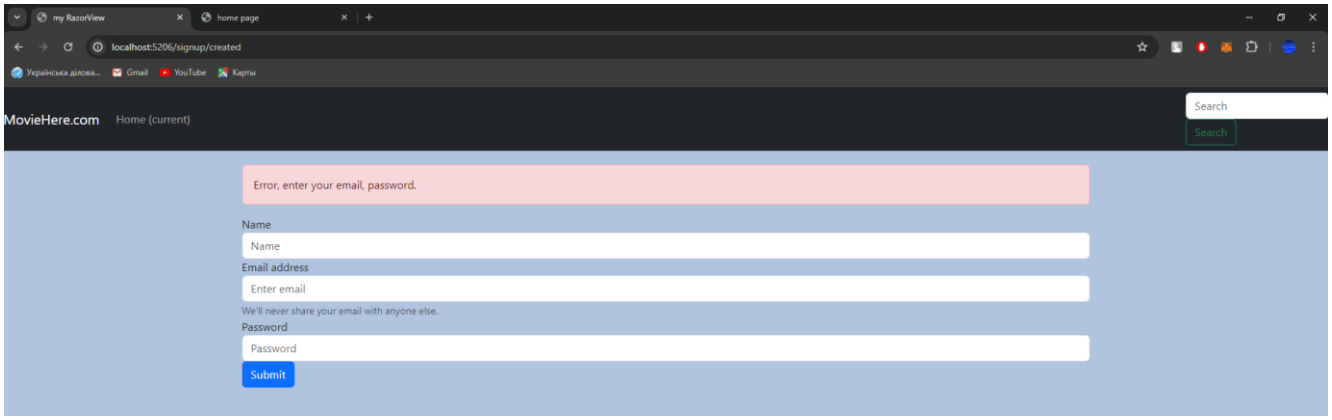

Рисунок 3.8 – Повідомлення про пропуск поля користувачем

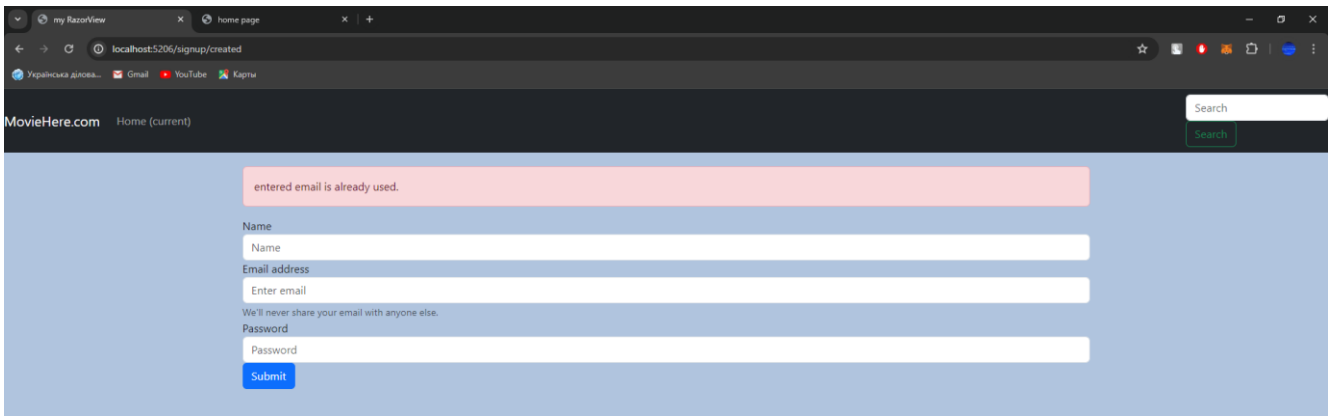

Рисунок 3.9 – Повідомлення про вже використаний для реєстрації раніше email

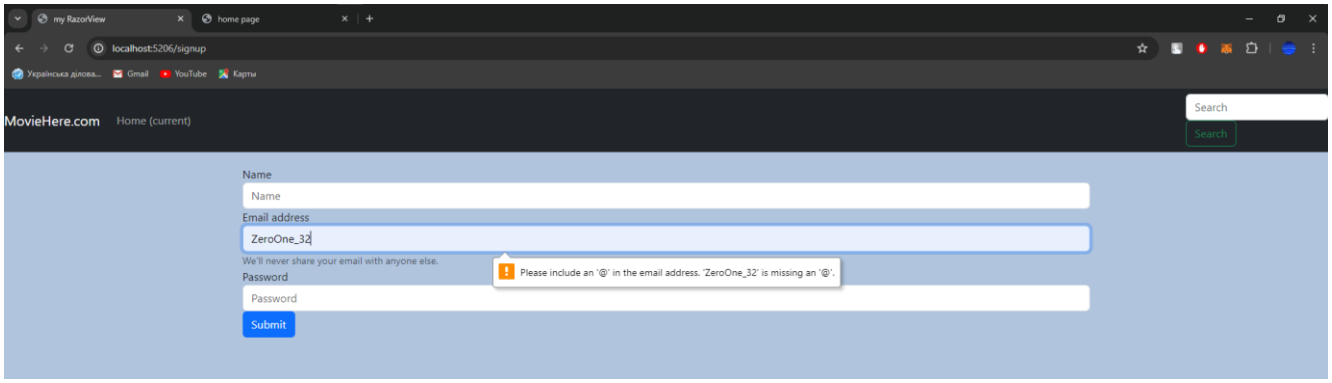

Рисунок 3.10 – Повідомлення про неправильно вказаний email

Після успішної реєстрації користувача переадресовує на сторінку авторизації, де він має зайти у свій новий акаунт.

Сторінка авторизації має схожу з сторінкою реєстрації структуру, маючи два поля для введення: поштову скриньку та пароль, а також кнопку підтвердження відправки форми, та кнопки для переходу на сторінку реєстрації, якщо користувач не зареєстрований (рис. 3.11). Ця сторінка також має відстеження помилок, таких як не введення поштової скриньки чи пароля, або введення неправильних даних для входу (рис. 3.12, 3.13).

Рисунок 3.11 – Сторінка авторизації

Рисунок 3.12 – Повідомлення про пропуск поля користувачем

Рисунок 3.13 – Повідомлення про неправильно введені дані для входу

Після реєстрації чи авторизації на сторінках з інформацією про конкретний фільм з’являється форма для створення рецензії на нього, що складається з: поля введення рейтингу, що надає користувач фільму, поля для тексту рецензії, а також кнопки для підтвердження відправки форми (рис. 3.14).

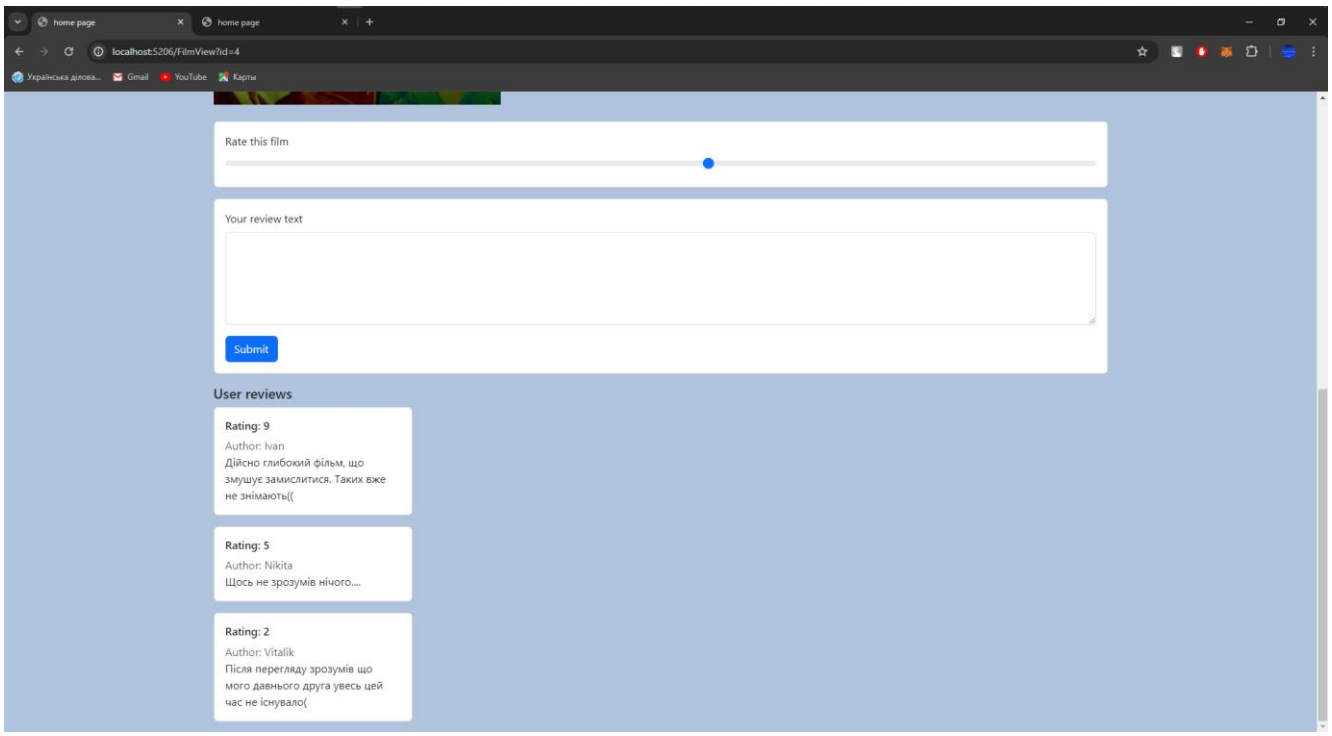

Рисунок 3.14 – Форма для написання рецензії

Також після авторизації можна побачити на верхній панелі нові кнопки Lo out та Film Panel (рис. 3.15)

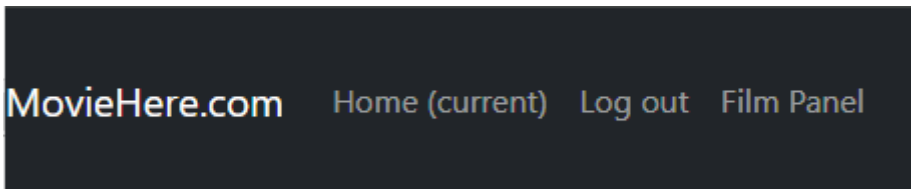

Рисунок 3.15 – Кнопка Log out

Film Panel переадресовує користувача на сторінку з формою для створення нового фільму. Форма містить такі поля як: поле для введення імені, опису, URL адреси постера, жанру, рока випуску, дати прем’єри, акторів, режисера, студії, та країни виробника фільму. Також є кнопка для відправки форми на сервер(рис. 3.16).

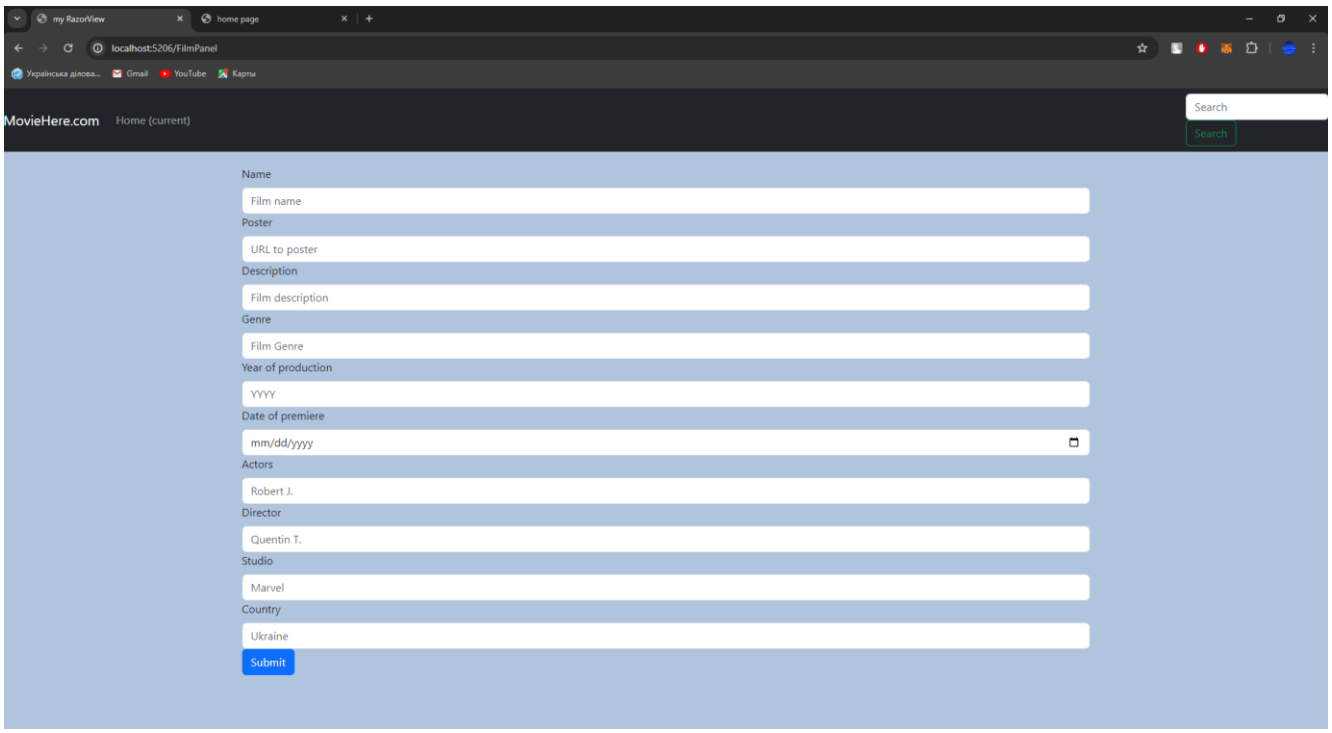

Рисунок 3.16 – Сторінка для додавання нового фільму

Також навігаційна панель має форму для пошуку фільмів за назвою, що складається з поля для введення назви та кнопки для відправлення форми на сервер (рис. 3.17)

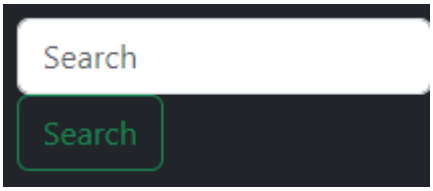

Рисунок 3.17 – Форма для пошуку фільму за назвою

Після відправки форми користувача переадресовує на сторінку з шуканими фільмами (рис. 3.18)

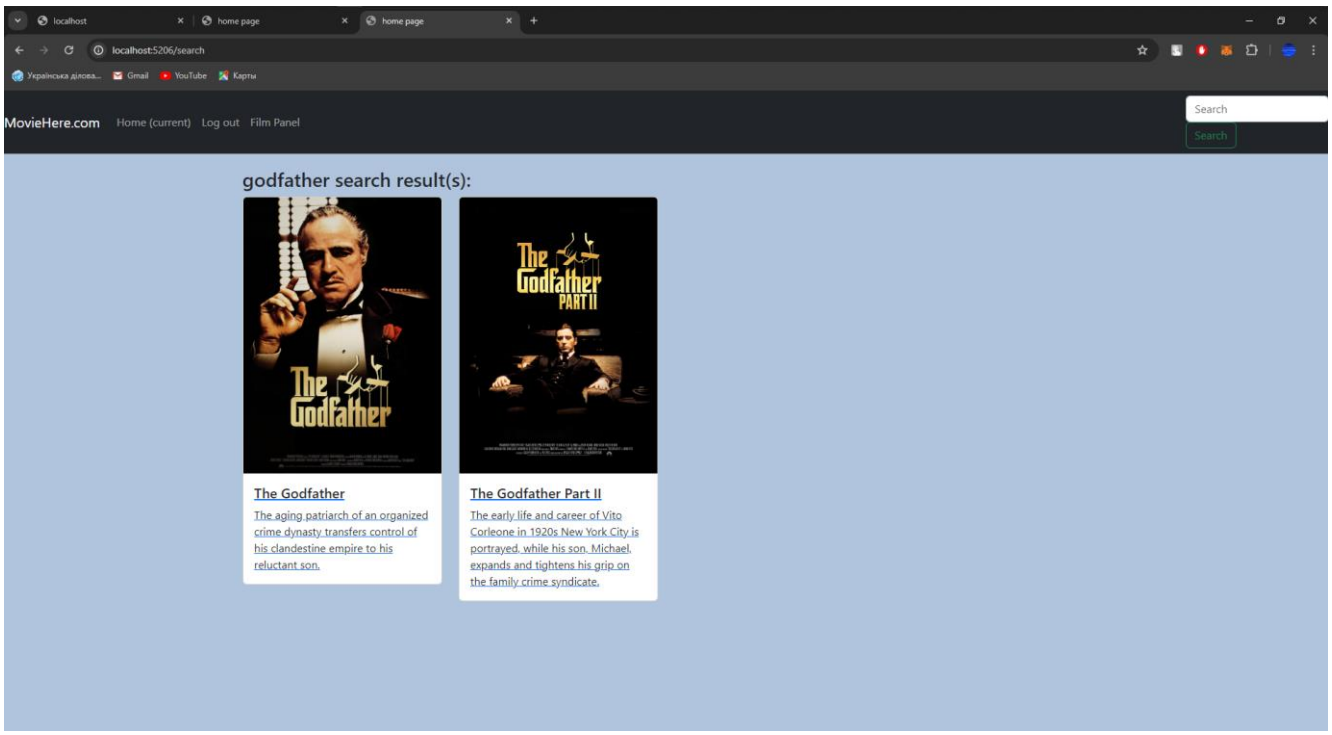

Рисунок 3.18 – Форма для пошуку фільму за назвою

3.3 Процес тестування програмного забезпечення

Для тестування онлайн-сервісу для кіноманів були розроблені тест-кейси. Тест-кейси представляють собою опис послідовності дій, та очікуваний результат до цих дій. Тест-кейс вважається успішно пройденим при правильній реакції програмного забезпечення на послідовність описаних дій [15].

Test case 1 відповідає за перевірку роботи механізмів авторизації вебзастосунку (табл 3.1).

Таблиця 3.1 – test case 1

Кроки	Очікуваний результат
1. Зайти на сторінку з авторизацією 2. Ввести коректні дані логіну	Користувач зайде у свій профіль

Test case 2 відповідає за перевірку реакції механізму авторизації на неправильно введені дані користувача (табл. 3.2).

Таблиця 3.2 – test case 2

Кроки	Очікуваний результат
1. Зайти на сторінку з авторизацією 2. Ввести некоректні дані логіну	З’явиться попередження про неправильно введені дані

Test case 3 відповідає за перевірку реакції програмного забезпечення на відправлення неповноцінної форми авторизації (табл. 3.3).

Таблиця 3.3 – test case 3

Кроки	Очікуваний результат
1. Зайти на сторінку з авторизацією 2. Відправити форму не вказавши логін або пароль користувача	З’явиться попередження про відсутність того чи іншого елемента у відправленій формі

Test case 4 відповідає за перевірку роботи механізму реєстрації (табл. 3.4).

Таблиця 3.4 – test case 4

Кроки	Очікуваний результат
1. Зайти на сторінку з реєстрацією 2. Відправити форму з коректними даними нового користувача	У базі даних з’явиться новий користувач, можна зайти в його профіль

Test case 5 відповідає за перевірку реакції механізму реєстрації на використання вже існуючого у базі даних Email (табл. 3.5).

Таблиця 3.5 – test case 5

Кроки	Очікуваний результат
1. Зайти на сторінку з реєстрацією 2. Відправити форму з вказанням Email, що вже зареєстрований	З’явиться попередження про те, що введений Email вже використовується

Test case 6 відповідає за перевірку реакції програмного забезпечення на відправлення неповноцінної форми реєстрації (табл. 3.6).

Таблиця 3.6 – test case 6

Кроки	Очікуваний результат
1. Зайти на сторінку з реєстрацією 2. Відправити форму не вказавши логін, ім’я або пароль користувача	З’явиться попередження про відсутність того чи іншого елемента у відправленій формі

Test case 7 відповідає за перевірку роботи рецензій на сайті (табл. 3.7).

Таблиця 3.7 – test case 7

Кроки	Очікуваний результат
1. Зайти у профіль користувача 2. Перейти на сторінку будь-якого фільму 3. Написати рецензію на фільм, вказавши рейтинг та текст рецензії	На сторінці фільму з’явиться рецензія користувача на нього

Test case 8 відповідає за перевірку реакції програмного забезпечення на відправлення неповноцінної рецензії на фільм (табл. 3.8).

Таблиця 3.8 – test case 8

Кроки	Очікуваний результат
1. Зайти у профіль користувача 2. Перейти на сторінку будь-якого фільму 3. Написати рецензію на фільм, вказавши рейтинг рецензії, але не написавши текст	З’явиться попередження про відсутність тексту рецензії

Test case 9 відповідає за перевірку реакції програмного забезпечення повторне написання рецензії користувачем (табл. 3.9).

Таблиця 3.9 – test case 9

Кроки	Очікуваний результат
1. Зайти у профіль користувача 2. Перейти на сторінку фільму, де користувач вже залишав раніше рецензію на фільм 3. Написати рецензію на фільм знову	Нова рецензія має замінити стару

Test case 10 відповідає за перевірку роботи функції пошуку фільмів за назвою на сайті (табл. 3.10).

Таблиця 3.10 – test case 10

Кроки	Очікуваний результат
1. Ввести у форму пошуку конкретну назву будь-якого фільму	З’явиться посилання на цей фільм

Test case 11 відповідає за перевірку роботи функції пошуку фільмів за назвою на сайті (табл. 3.11).

Таблиця 3.11 – test case 11

Кроки	Очікуваний результат
1. Ввести у форму пошуку рядок, що міститься у назві декількох фільмів	З’являться посилання на усі фільми, що мають назви, які підходять під шуканий рядок.

На основі вказаних тест-кейсів було проведено мануальне тестування вебзастосунку, у результаті якого були сформовані чек-ліст (рис. 3.19) та баг-репорт (табл. 3.12).

Id	Name	Status
1	Перевірка роботи авторизації	passed
2	Авторизація користувача з некоректними даними	passed
3	Авторизація користувача не вказуючи деякі дані	passed
4	Перевірка роботи реєстрації	passed
5	Реєстрація користувача з все існуючим у базі даних Email	passed
6	Реєстрація користувача не вказуючи деякі дані	passed
7	Перевірка роботи рецензій	passed
8	Відправка рецензії користувача без тексту	failed
9	Повторне написанння рецензії користувачем	passed
10	Пошук конкретного фільму	passed
11	Пошук серії фільмів чи фільмів зі схожою назвою	passed

Рисунок 3.19 – Чек-Ліст тестів

Таблиця 3.12 – Баг-репорт

Id репорту	1
summary	Відправка рецензії без тексту ніяк не повідомляє користувача про його відсутність.
Precondition	Зайти на онлайн-сервіс для кіноманів.
Steps	1. Авторизуватися на сайті 2. Зайти на сторінку будь-якого фільму 3. Вказати рейтинг у рецензії, та не заповнити поле з текстом 4. Натиснути кнопку «Sumbit»
Expected result	З’явиться попередження про відсутність тексту у рецензії.
Actual result	Сторінка оновлюється, але ніякого попередження не з’являється.

Було проведено ручне тестування розробленого вебзастосунку, під час якого перевірялися всі основні функціональні модулі. Результатом цього тестування став чек-лист, який допоміг систематизувати процес перевірки і виявити недоліки. Крім

того, був створений один звіт про помилки, що дозволив оперативно виправити виявлені проблеми.

3.4 Висновки до розділу 3

Було реалізовано функціонал програмного забезпечення, який має такі компоненти: каталог фільмів на головній сторінці сайту, система реєстрації та авторизації користувачів, панель для додавання нових фільмів у каталог, сторінка з детальною інформацією про конкретний фільм та рецензії на нього, форма для написання рецензій, а також форма для пошуку фільмів за назвою.

Проведене ручне тестування розробленого вебзастосунку дозволило створити чек-ліст тестів та один баг-репорт. Це тестування забезпечило виявлення та виправлення помилок, що сприяло підвищенню стабільності та якості роботи системи.

ВИСНОВКИ

У кваліфікаційній роботі бакалавра розроблено онлайн-сервіс для кіноманів, який надає можливість переглядати список фільмів, ознайомлюватися з основною інформацією про них та залишати відгуки. Цей проєкт створений засобами мови програмування C#, фреймворка ASP.NET Core та бібліотеки Bootstrap, що широко застосовуються у сучасній розробці вебзастосунків.

При аналізі предметної області з'ясовано особливості розробки онлайн-сервісів для кіноманів. Проведений аналіз програмних конкурентів дозволив створити SWOT-аналіз, в результаті якого було визначено сильні і слабкі сторони, можливості та загрози для створення такого сервісу. Аналіз показав, що створення онлайн-сервісу для кіноманів є доцільним та перспективним, зважаючи на постійно зростаючий попит на такі ресурси серед користувачів.

На основі аналізу предметної області та конкурентів були розроблені функціональні вимоги до онлайн-сервісу. Для реалізації back-end частини вебзастосунку був вибраний фреймворк ASP.NET Core, який забезпечує високу продуктивність, безпеку та можливість легкої масштабованості. Для front-end частини використовувалась бібліотека Bootstrap, яка дозволяє створювати адаптивний та привабливий інтерфейс користувача.

Також була розроблена маркетингова стратегія для просування та розвитку вебзастосунку, яка охоплює використання соціальних медіа, SEO-оптимізацію та партнерські програми для залучення нових користувачів.

Архітектура сайту спроєктована з урахуванням використання концепції middleware та архітектурної схеми MVC (Model-View-Controller). Це забезпечує поділ логіки застосунку на три взаємодіючі компоненти, що сприяє більшій гнучкості та підтримуваності коду. Була розроблена реляційна база даних, яка зберігає інформацію про користувачів, фільми та рецензії, забезпечуючи надійне та ефективне зберігання даних.

Проведене моделювання функціоналу системи дозволило створити діаграму прецедентів (use-case діаграму) для застосунку. Це дало змогу чітко визначити

взаємодію користувачів із системою та спланувати основні сценарії використання сервісу.

В роботі реалізовано повний функціонал вебсайту для кіноманів, який слугуватиме платформою для надання довідкової інформації про фільми (актори, режисер, опис), відгуки, оцінки підписників та іншу інформацію про улюблені відеострічки:

– відображення каталогу фільмів на головній сторінці сайту: користувачі можуть переглядати список доступних фільмів;

– система реєстрації та авторизації користувачів забезпечує можливість створення облікового запису та входу до системи;

– панель для додавання нових фільмів до каталогу: адміністратори можуть додавати нові фільми, включаючи їх опис та постери;

– сторінка з детальною інформацією про конкретний фільм та рецензії на нього: користувачі можуть переглядати детальну інформацію про фільм та читати рецензії інших користувачів;

– форма для написання рецензій: авторизовані користувачі можуть залишати свої відгуки про фільми;

– форма для пошуку фільмів за назвою: користувачі можуть швидко знаходити потрібні фільми за допомогою пошукової функції.

Проведено мануальне тестування розробленого вебзастосунку, що включало перевірку всіх основних функціональних модулів. У результаті тестування був створений чек-ліст тестів, що допоміг систематизувати процес перевірки та виявити недоліки. Крім того, був складений один баг-репорт, який дозволив оперативно виправити виявлені помилки.

У рамках подальших досліджень і розвитку онлайн-сервісу для кіноманів, особливу увагу варто приділити більш надійному зберіганню користувацької інформації. Крім того, розробка персоналізованих алгоритмів рекомендацій на основі машинного навчання та аналізу великих даних дозволить покращити користувацький досвід, пропонуючи кожному користувачеві індивідуальні рекомендації фільмів на основі їхніх уподобань, історії переглядів та оцінок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The Best Movie Streaming Services for 2024. URL: https://www.pcmag.com/picks/best-movie-streaming-services

2. 10 українських сервісів для справжніх кіноманів. URL: https://double.news/2023/06/30/10-ukrayinskyh-servisiv-dlya-spravzhnih-kinomaniv/

3. What is IMDb? URL: https://help.imdb.com/article/imdb/general-information/what-is-imdb/G836CY29Z4SGNMK5?ref_=helpart_nav_1#

4. About Rotten Tomatoes. URL: https://www.rottentomatoes.com/about#whatisthetomatometer.

5. A tour of the C# language. URL: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

6. Рейтинг мов програмування 2024. URL: https://dou.ua/lenta/articles/language-rating-2024/

7. Overview of ASP.NET Core. URL: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0

8. Overview of ASP.NET Core. URL: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0

9. Наливайко А. Переваги цифрового маркетингу для розробників софту. URL: https://bizzzdev.com/marketing-for-software-companies-the-whats-and-whys/

10. The “How To Market Software Products” Playbook – 14 Product Marketing Strategies For SaaS. URL: https://userpilot.com/blog/how-to-market-software-products/

11. Oladeinde A. Creating a web socket Using Middleware with ASP.NET core 3.1. URL: https://medium.com/@mamaradonakindookindoo/creating-a-web-socket-using-middleware-with-asp-net-core-3-1-b406defb84e6

12. Дацко І. А. Засоби .NET для розробки програмного забезпечення. *Кібербезпека в сучасному світі* : матер. III всеукр. наук.-практ. конф. (м. Одеса, 19 листопада 2021 р.). Одеса, 2021. С. 124-127. URL: https://dspace.onua.edu.ua/items/deb11a37-0bd2-4f3b-9f9c-f783fdf8063c.

13. MVC Design Pattern. URL: https://www.geeksforgeeks.org/mvc-design-pattern/

14. ntext, text, and image (Transact-SQL) URL: https://learn.microsoft.com/en-us/sql/t-sql/data-types/ntext-text-and-image-transact-sql?view=sql-server-ver16.

15. Дацко І. А. Засоби Hardhat для автоматизованого тестування програмних застосунків у мережі блокчейн. *Інформаційне суспільство: проблеми та перспективи* : матер. VIII всеукр. наук.-практ. конф. (м. Одеса, 2 черв. 2023 р.). Одеса, 2023. 93-94 с. URL: https://doi.org/10.32837/11300.25263.

ДОДАТКИ

Додаток А. Фрагменти програмного коду

Index.cshtml

```
@model IEnumerable<mdb_project.Model.Film>
@using System.Security.Claims


<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet"
    integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">

    <title>home page</title>
</head>
<body style="background-color:lightsteelblue">
    <header>
        <nav class="navbar sticky-top navbar-expand-lg navbar-dark bg-dark">
            <a class="navbar-brand" href="#">MovieHere.com</a>
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNavDropdown"
            aria-controls="navbarNavDropdown" aria-expanded="false" aria-label="Toggle
navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNavDropdown">
                <ul class="navbar-nav">
                    <li class="nav-item active">
                        <a class="nav-link" asp-controller="Home" asp-action="Index">Home
<span class="sr-only">(current)</span></a>
                    </li>
                    @if (!User.Identity.IsAuthenticated)
                    {
                        <li class="nav-item">
                            <a class="nav-link" href="/login">LogIn</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="/signup">Sign Up</a>
                        </li>
                    }
                    @if (User.Identity.IsAuthenticated)
                    {
                        <li class="nav-item">
                            <a class="nav-link" asp-controller="Users" asp-
action="Logout">Log out</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="/FilmPanel">Film Panel</a>
                        </li>
                    }
                </ul>
            </div>
            <form class="form-inline" asp-controller="Film" asp-action="FilmSearch">
```

```
            <input class="form-control mr-sm-2" type="search" name="SearchString"
placeholder="Search" aria-label="Search">
                <button class="btn btn-outline-success my-2 my-sm-0"
type="submit">Search</button>
            </form>
        </nav>
    </header>

    <div style="padding-top: 20px; padding-bottom: 20px; padding-left: 18%; padding-
right: 18%">

        <div class="row row-cols-1 row-cols-md-4 g-4">

            @foreach (var film in Model)
            {
                @if (film.Name.Contains("test"))
                {
                    continue;
                }
                <a asp-controller="Film" asp-action="FilmView" asp-route-id=@film.Id>
                    <div class="col">
                        <div class="card">
                            <img src=@film.Poster class="card-img-top" height="400"
alt="no poster :(">
                            <div class="card-body">
                                <h5 class="card-title">@film.Name</h5>
                                <p class="card-text">@film.Year</p>
                            </div>
                        </div>
                    </div>
                </a>
            }

        </div>
    </div>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
</body>
      </html>
```

FilmPanel.cshtml

```
@*
    For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
}
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet"
          integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">

    <title>my RazorView</title>
</head>
```

```
<body style="background-color:lightsteelblue">
    <header>
        <nav class="navbar sticky-top navbar-expand-lg navbar-dark bg-dark">
            <a class="navbar-brand" href="#">MovieHere.com</a>
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNavDropdown"
                    aria-controls="navbarNavDropdown" aria-expanded="false" aria-
label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNavDropdown">
                <ul class="navbar-nav">
                    <li class="nav-item">
                        <a class="nav-link" href="/home">Home <span class="sr-
only">(current)</span></a>
                    </li>
                </ul>
            </div>
            <form class="form-inline" asp-controller="Film" asp-action="FilmSearch">
                <input class="form-control mr-sm-2" type="search" placeholder="Search"
aria-label="Search">
                <button class="btn btn-outline-success my-2 my-sm-0"
type="submit">Search</button>
            </form>
        </nav>
    </header>

    <div div style="padding-top: 20px; padding-bottom: 20px; padding-left: 18%; padding-
right: 18%">
        @if (ViewBag.Message != null)
        {
            <div class="alert alert-success" role="alert">
                @ViewBag.Message
            </div>
        }

        <!-- connection to Users controller here -->
        <form asp-controller="Film" asp-action="Create">
            <label for="inputPassword5" class="form-label">Name</label>
            <input class="form-control" type="text" name="Name" placeholder="Film name"
aria-label="default input example">

            <label for="inputPassword5" class="form-label">Poster</label>
            <input class="form-control" type="text" name="Poster" placeholder="URL to
poster" aria-label="default input example">

            <label for="inputPassword5" class="form-label">Description</label>
            <input class="form-control" type="text" name="Description" placeholder="Film
description" aria-label="default input example">

            <label for="inputPassword5" class="form-label">Genre</label>
            <input class="form-control" type="text" name="Genre" placeholder="Film Genre"
aria-label="default input example">

            <label for="inputPassword5" class="form-label">Year of production</label>
            <input class="form-control" type="text" name="Year" placeholder="YYYY" aria-
label="default input example">

            <label for="inputPassword5" class="form-label">Date of premiere</label>
            <input class="form-control" type="date" name="Premiere"
placeholder="MM/DD/YYYY" aria-label="default input example">

            <label for="inputPassword5" class="form-label">Actors</label>
            <input class="form-control" type="text" name="Actors" placeholder="Robert J."
aria-label="default input example">
```

```
            <label for="inputPassword5" class="form-label">Director</label>
            <input class="form-control" type="text" name="Director" placeholder="Quentin
T." aria-label="default input example">

            <label for="inputPassword5" class="form-label">Studio</label>
            <input class="form-control" type="text" name="Studio" placeholder="Marvel"
aria-label="default input example">

            <label for="inputPassword5" class="form-label">Country</label>
            <input class="form-control" type="text" name="Country" placeholder="Ukraine"
aria-label="default input example">

            <button type="submit" class="btn btn-primary">Submit</button>
        </form>
    </div>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
            integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
</body>
      </html>
```

FilmPanel.cshtml

```
@*
    For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
}
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet"
          integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">

    <title>my RazorView</title>
</head>

<body style="background-color:lightsteelblue">
    <header>
        <nav class="navbar sticky-top navbar-expand-lg navbar-dark bg-dark">
            <a class="navbar-brand" href="#">MovieHere.com</a>
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNavDropdown"
                    aria-controls="navbarNavDropdown" aria-expanded="false" aria-
label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNavDropdown">
                <ul class="navbar-nav">
                    <li class="nav-item">
                        <a class="nav-link" href="/home">Home <span class="sr-
only">(current)</span></a>
                    </li>
                </ul>
```

```
        </div>
            <form class="form-inline" asp-controller="Film" asp-action="FilmSearch">
                <input class="form-control mr-sm-2" type="search" placeholder="Search"
aria-label="Search">
                <button class="btn btn-outline-success my-2 my-sm-0"
type="submit">Search</button>
            </form>
        </nav>
    </header>

    <div div style="padding-top: 20px; padding-bottom: 20px; padding-left: 18%; padding-
right: 18%">
        @if (ViewBag.Message != null)
        {
            <div class="alert alert-success" role="alert">
                @ViewBag.Message
            </div>
        }

        <!-- connection to Users controller here -->
        <form asp-controller="Film" asp-action="Create">
            <label for="inputPassword5" class="form-label">Name</label>
            <input class="form-control" type="text" name="Name" placeholder="Film name"
aria-label="default input example">

            <label for="inputPassword5" class="form-label">Poster</label>
            <input class="form-control" type="text" name="Poster" placeholder="URL to
poster" aria-label="default input example">

            <label for="inputPassword5" class="form-label">Description</label>
            <input class="form-control" type="text" name="Description" placeholder="Film
description" aria-label="default input example">

            <label for="inputPassword5" class="form-label">Genre</label>
            <input class="form-control" type="text" name="Genre" placeholder="Film Genre"
aria-label="default input example">

            <label for="inputPassword5" class="form-label">Year of production</label>
            <input class="form-control" type="text" name="Year" placeholder="YYYY" aria-
label="default input example">

            <label for="inputPassword5" class="form-label">Date of premiere</label>
            <input class="form-control" type="date" name="Premiere"
placeholder="MM/DD/YYYY" aria-label="default input example">

            <label for="inputPassword5" class="form-label">Actors</label>
            <input class="form-control" type="text" name="Actors" placeholder="Robert J."
aria-label="default input example">

            <label for="inputPassword5" class="form-label">Director</label>
            <input class="form-control" type="text" name="Director" placeholder="Quentin
T." aria-label="default input example">

            <label for="inputPassword5" class="form-label">Studio</label>
            <input class="form-control" type="text" name="Studio" placeholder="Marvel"
aria-label="default input example">

            <label for="inputPassword5" class="form-label">Country</label>
            <input class="form-control" type="text" name="Country" placeholder="Ukraine"
aria-label="default input example">

            <button type="submit" class="btn btn-primary">Submit</button>
        </form>
    </div>
```

```
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
            integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
</body>
      </html>
```

FilmController.cs

```
using mdb_project.Migrations;
using mdb_project.Model;
using Microsoft.AspNetCore.Mvc;
using Microsoft.CodeAnalysis.Elfie.Serialization;
using Microsoft.EntityFrameworkCore;
using static System.Net.Mime.MediaTypeNames;

namespace mdb_project.Controllers
{
    public class FilmController : Controller
    {
        private readonly FilmDBContext _context;

        public FilmController(FilmDBContext context)
        {
            _context = context;
        }

        [Route("FilmPanel/created")]
        [HttpPost]
        public IActionResult Create(Film film)
        {
            if (ModelState.IsValid)
            {
                _context.Films.Add(film);
                _context.SaveChanges();
            }
            ViewBag.Message = "Film created";
            return View("FilmPanel");
        }

        [Route("/FilmView")]
        public IActionResult FilmView(int id)
        {
            var film = _context.Films.FirstOrDefault(f => f.Id == id);

            if (film == null)
            {
                return NotFound();
            }

            var reviews = _context.Reviews.Where(r => r.FilmId == id).ToList();

            var usersWhoReviewedFilmId = _context.Reviews
                .Where(r => r.FilmId == id)
                .Select(r => r.UserId)
                .Distinct()
```

```
.ToList();

            var users = _context.Users
                .Where(u => usersWhoReviewedFilmId.Contains(u.Id))
                .ToList();

            var viewModel = new FilmViewModel
            {
                Film = film,
                Reviews = reviews,
                Users = users
            };

            return View(viewModel);
        }

        [Route("search")]
        [HttpPost]
        public IActionResult FilmSearch(string searchString)
        {
            if (string.IsNullOrEmpty(searchString))
            {
                return RedirectToAction("Index", "Home");
            }

            ViewBag.SearchString = searchString;

            var movies = _context.Films
                .Where(f => f.Name.Contains(searchString))
                .ToList();

            return View("FilmSearch", movies);
        }

        [Route("FilmView/ReviewCreated")]
        [HttpPost]
        public IActionResult CreateReview(Review review)
        {
            if (ModelState.IsValid)
            {
                var existingReview = _context.Reviews.FirstOrDefault(r =>
                r.UserId == review.UserId && r.FilmId == review.FilmId);

                if (existingReview != null)
                {
                    // Updating existing review
                    existingReview.Rating = review.Rating;
                    existingReview.Text = review.Text;

                    _context.SaveChanges();

                    return RedirectToAction("FilmView", new { id = review.FilmId });
                }

                _context.Reviews.Add(review);
                _context.SaveChanges();
            }
            return RedirectToAction("FilmView", new {id = review.FilmId});
        }

    }
}
```

Додаток Б. Копії документів про апробацію результатів роботи

Дацко Іван Андрійович

Національний університет «Одеська юридична академія»,

студент 2-го курсу факультету кібербезпеки та інформаційних технологій

ЗАСОБИ .NET ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

.NET – це платформа від Microsoft, що дозволяє розробляти програмне забезпечення. Є доволі популярна думка, що .NET використовується тільки для роботи з мовою програмування C#, але це не так [1]. Звісно, вони тісно пов'язані, але нині цей фреймворк також підтримує розробку мовами C++, F# та VB.NET.

Код компілюється в збірку загальною мовою CIL (Common Intermediate Language), яка далі зберігається у файлі формату PE (Portable Executable). Ця збірка компілюється у байт-код та за допомогою технології JIT (Just-in-time compilation), транслюється у машинний код під час роботи програми [2]. Саме через CIL стає можливою розробка різних модулів програмного забезпечення різними мовами.

2016 року Microsoft випустила у реліз .NET Core, який зараз має назву .NET 6. Головною відміною від свого попередника .NET Framework, що зараз офіційно не підтримується (останнє оновлення було 2019 року [3]), є наявність кросплатформності. Тобто програми на .NET 6 можна розробляти як на Windows, так і на MacOS та Linux. Крім того, з'явилася можливість створювати застосунки для мобільних платформ.

Окрім кросплатформності та підтримки декількох мов програмування, фреймворк від Microsoft має доволі велику та різноманітну бібліотеку класів, яка дозволяє застосовувати їх для розробки різних застосунків. Завдяки цій

бібліотеці та загальномовному середовищу виконання CLR [4], що по суті є віртуальною машиною для виконання програмного коду, написаного відповідними мовами, .NET підтримує широкий спектр технологій, які в свою чергу розширюють можливості для створення програм. Так, для розробки баз даних за допомогою цієї платформи використовують ADO.NET, що є набором класів для отримання доступу до даних у реляційних і XML-базах даних та даних з клієнтських застосунків [5]. Також для роботи з БД користуються мовою запитів Transact-SQL та ORM-фреймворком Entity Framework (ORM – об'єктно реляційна проекція, тобто технологія програмування, що зв'язує між собою бази даних та основні концепції ООП, і створює так звану «віртуальну базу даних» [6]), що дозволяє розробникам працювати з даними через об'єкти класів, не працюючи з базовими таблицями баз даних.

Для розробки вебзастосунків в .NET є ASP.NET – кросплатформне середовище з відкритим вихідним кодом. Воно підходить як для створення сучасних хмарних застосунків, так і для написання серверної частини клієнтського забезпечення, та навіть для програмування застосунків для роботи Інтернету речей (IoT) [7]. Одночасно ASP.NET може бути рішенням і для створення вебінтерфейсу, і для API (Application Programming Interface). У контексті цього середовища також використовують вищезгаданий ADO.NET для легкого доступу до опрацьовуваних даних.

За допомогою .NET можна розробляти комп'ютерні ігри. Популярним середовищем для розробки ігор, особливо для розробників інді-ігор, є Unity. Воно підтримує розробку скриптів на .NET та розширює його функціонал численними класами, які дозволяють взаємодіяти з ігровими об'єктами. Рушій Unity надає широкий вибір інструментів для взаємодії з елементами у грі та їх редагуванням. Unity дозволяє навіть новачкам створювати як 2D, так і 3D ігри під будь-які платформи – будь-то VR, мобільні девайси, ПК або інші засоби відтворення ігор. Також це середовище приваблює вже готовими рішеннями для фізики об'єктів, оскільки має великий вибір параметрів для налаштування [8].

Іншим кросплатформним фреймворком для розробки клієнтських застосунків для мобільних телефонів на .NET, є платформа Xamarin [9]. Цей

фреймворк з відкритим вихідним кодом надає можливість створювати застосунки як для Android, так і для iOS, при чому при портуванні програми з однієї платформи на іншу 90% її коду не буде змінено. Програмне забезпечення на Xamarin можна розробляти як на Windows, так і на MacOS, і компілювати його у пакети, наприклад, .apk для Android чи .ipa для iOS.

Потужним засобом розробки серверної частини вебсайтів є ASP.NET MVC. Він забезпечує виведення необхідного контенту з бази даних у потрібні ділянки веб-сайту, автоматизує процес збору інформації про користувачів, визначає роботу бізнес логіки на сервері, захищає сайт від злому, DoS та DDoS атак.

Отже, основними перевагами у використанні платформи .NET є: кросплатформність, технології CLR, CIL і JIT, стандартна бібліотека класів, широкий вибір допоміжних технологій та платформ (ASP.NET, Unity, Xamarin та ін.), що розширюють можливості .NET.

Зараз .NET програмісти доволі затребувані на ринку праці, оскільки .NET дозволяє розробляти програмне забезпечення самого різного призначення: веб-застосунки, десктопні програми, хмарні послуги, ігри, мобільні застосунки тощо. Наприклад, Unity/Game Developer – це фахівець, який розробляє ігри за допомогою рушія Unity та мови C#. Крім того, цей розробник може створювати бізнес-програми, які можуть застосовуватися в рекламних кампаніях..NET Desktop Developer розробляє десктопне програмного забезпечення для цілого спектру продуктів Microsoft з ОС Windows 10: десктопи, планшети, пристрої інтернету речей, мобільні пристрої, приставки Xbox тощо. WPF і UWP технології допомагають цьому фахівцю в роботі, а ITVDN стане надійним помічником у вивченні основ, що дозволяють створювати якісні настільні програми для користувачів Windows. ASP.NET MVC Developer відповідає за розробку серверної частини вебсайту. Цей перелік фахівців в .NET далеко не вичерпний, а технології .NET розвиваються і вдосконалюються.

Список використаних джерел:

1. Что такое .NET и чем занимаются .NET-разработчики. URL: https://training.epam.ua/#!/News/301?lang=ru

2. CIL – Common Intermediate Language. URL: https://www.scriptol.com/programming/cil.php

3. Язык C# и платформа .NET Core. URL: https://metanit.com/sharp/tutorial/1.1.php

4. Common Language Runtime. (CLR) overview. URL: https://docs.microsoft.com/en-us/dotnet/standard/clr

5. ADO.NET. URL: https://docs.microsoft.com/ru-ru/dotnet/framework/data/adonet/

6. What is Entity Framework? URL: https://www.entityframeworktutorial.net/what-is-entityframework.aspx

7. Документация ASP.NET. URL: https://docs.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-5.0

8. Overview of .NET in Unity. URL: https://docs.unity3d.com/Manual/overview-of-dot-net-in-unity.html

9. Что такое Xamarin? URL: https://docs.microsoft.com/ru-ru/xamarin/get-started/what-is-xamarin

Ключові слова: .NET, кросплатформність, C#, веб, CLR.

Ключевые слова: .NET, кроссплатформность, C#, веб, CLR.

Key words: .NET, cross platform, C#, web, CLR.

Науковий керівник: *к.т.н., доц. Трофименко О. Г.*