

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Вебсайт "Кінопошук"»

Виконав студент групи ІПЗк-23-1
СЛІПКОВСЬКИЙ Ілля Сергійович

Керівник роботи:
ЧИЖМОТРИЯ Олексій Володимирович

Рецензент:
ГОДЛЕВСЬКИЙ Юрій Олександрович

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна Вакалюк

«14» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **СЛІПКОВСЬКИЙ Ілля Сергійович**

Керівник роботи: **ЧИЖМОТРИЯ Олексій Володимирович**

Тема роботи: **«Вебсайт "Кінопошук"»**,

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: предметом дослідження виступають веборієнтовані технології, а об'єктом — процес онлайн-пошуку, оцінювання та каталогізації фільмів.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотрия О.В.	14.02.2025р.	14.02.2025р.
2	Чижмотрия О.В.	14.03.2025р.	14.03.2025р.
3	Чижмотрия О.В.	14.04.2025р.	14.04.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота складається з розробленого вебдодатку для онлайн пошуку фільму та пояснювальної записки.

Пояснювальна записка до випускної роботи містить 87 сторінок, 22 ілюстрації, 8 таблиць, 9 додатків та 30 використаних джерел.

Метою кваліфікаційної роботи є проєктування та створення вебсайту "Кінопошук", що надає користувачам зручний інструмент для пошуку фільмів, перегляду детальної інформації, можливості їх оцінювання та залишення відгуків. Платформа також дозволяє адміністраторам керувати контентом: додавати та редагувати інформацію про фільми, акторів, жанри, а також модерувати відгуки користувачів.

У роботі сформульовано та розв'язано основні задачі, необхідні для створення вебсайту "Кінопошук", здійснено детальний аналіз та порівняння існуючих аналогічних онлайн-платформ та кінопорталів.

Проведено обґрунтування вибору інструментальних засобів і технологій, що відповідають сучасним тенденціям у сфері веброзробки. Особливу увагу приділено аналізу варіантів використання та визначенню структури додатку.

Спроектовано та розроблено базу даних для зберігання необхідної інформації про користувачів, фільми, жанри, акторів, режисерів, оцінки, відгуки та списки улюблених фільмів.

Описано процес технічної реалізації вебдодатку, який повністю відповідає сучасним вимогам до інформаційних вебсервісів у сфері кіно та готовий до впровадження та експлуатації.

КЛЮЧОВІ СЛОВА: ВЕБДОДАТОК, КІНОПОШУК, ФІЛЬМИ, РЕЙТИНГИ, ВІДГУКИ, LARAVEL, REACT, DOCKER

					ІПЗ.КР.Б – 121 – 25 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Вебсайт «Кінопошук» Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		Сліпковський І.С						
Керівник		Чижмотря О. В.					3	87
Рецензент		Годлевський Ю.О				Житомирська політехніка, група ІПЗк-23-1		
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's qualification work consists of a web application for online movie search and an explanatory note.

The explanatory note accompanying the thesis comprises 87 pages, 22 illustrations, 8 tables, 9 appendices, and 30 references.

The purpose of the qualification work is the design and implementation of the “Kinopoisk” website, which provides users with a convenient tool for finding films, viewing detailed information, rating them, and leaving reviews. The platform also enables administrators to manage content: add and edit information about films, actors, and genres, as well as moderate user reviews.

The research formulates and solves the key tasks required to create the “Kinopoisk” web system, including a detailed analysis and comparison of existing online platforms and movie portals.

A thorough justification is given for the selection of tools and technologies that align with current trends in web development. Special attention is devoted to analyzing use cases and determining the application's structure.

A database has been designed and implemented to store essential information about users, films, genres, actors, directors, ratings, reviews, and favorites lists.

The thesis also describes the technical implementation process of the web application, which fully meets modern requirements for information web services in the film industry and is ready for deployment and operational use.

KEYWORDS: WEB APPLICATION, KINOPOISK, FILMS, RATINGS, REVIEWS, LARAVEL, REACT, DOCKER

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ВЕБСАЙТУ «КІНОПОШУК».....	9
1.1. Опис та постановка задачі.....	9
1.2. Дослідження, аналіз та порівняння функціональних можливостей аналогів програмного продукту.....	12
1.3. Вибір архітектури вебдодатку	19
1.4. Обґрунтування вибору інструментальних засобів для розробки.....	22
Висновки до першого розділу.....	24
РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБСАЙТУ «КІНОПОШУК».....	27
2.1. Визначення варіантів використання та структури системи	27
2.2. Проєктування та розробка бази даних вебдодатку.....	33
2.3. Реалізація функціоналу роботи вебдодатку для вебсайту «кінопошук»	38
Висновки до другого розділу	45
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК ВЕБСАЙТУ «КІНОПОШУК»	47
3.1. Порядок встановлення та налаштування системи.....	47
3.2. Структура інтерфейсу вебсайту «кінопошук» та порядок роботи	47
3.3. Тестування роботи вебдодатку.....	53
Висновки до третього розділу.....	58
ВИСНОВКИ.....	60
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	63
ДОДАТКИ.....	66

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

CRUD – Create, Read, Update, Delete

JSON – JavaScript Object Notation

JWT – JSON Web Token

REST – Representational State Transfer

SQL – Structured Query Language

БД – база даних

СУБД – система управління базами даних

					ІІЗк.КР.Б – 121 – 25 – ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

У сучасному середовищі масового медіаспоживання кінематограф став одним із найдинамічніших секторів розважальної індустрії. Щодня глядачі отримують десятки нових трейлерів і прем'єр, а каталоги стрімінгових сервісів безупинно поповнюються свіжими релізами, що ускладнює вибір контенту та спонукає користувачів шукати надійні джерела систематизованої інформації. Разом із кількістю фільмів зростають і вимоги до зручності пошуку, швидкого порівняння рейтингів, ознайомлення з відгуками та відстеження творчості улюблених акторів і режисерів.

У цих умовах виникає необхідність у централізованому веб-ресурсі, який об'єднає в одному інтерфейсі повну базу даних стрічок і надасть інструменти для персоналізованої взаємодії. Запропонований вебсайт «Кінопошук» покликаний задовольнити цей попит, дозволяючи користувачам за кілька секунд знаходити фільми за назвою, прізвищем режисера чи актора, жанром або роком випуску, переглядати трейлер і постер, стежити за середнім рейтингом, залишати власні оцінки та рецензії, а також формувати персональні добірки. Адміністративна частина ресурсу забезпечить швидке оновлення даних, модерацію нових записів і контроль за користувацьким контентом, що особливо важливо для підтримання якості інформації та належного рівня взаємодії у спільноті.

Актуальність теми зумовлена зростанням очікувань аудиторії щодо швидкодії, достовірності й інтегрованості сервісів: сучасний глядач прагне миттєво отримувати всю потрібну інформацію, не переходячи між численними сторонніми платформами. Вебсайт «Кінопошук» пропонує комплексне рішення цієї проблеми, об'єднуючи детальні картки фільмів, механізми соціальної взаємодії та ефективні інструменти адміністрування.

Метою кваліфікаційної роботи є проектування та реалізація вебсайту «Кінопошук», який поєднує розширений пошук, детальну інформацію про стрічки, можливості оцінювання й коментування та надійну систему керування контентом.

					ІІЗк.КР.Б – 121 – 25 – ІЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

Для досягнення поставленої мети визначено наступні завдання:

1. Провести аналіз існуючих кіно-платформ і виявити їхні сильні та слабкі сторони.
2. Сформулювати функціональні й нефункціональні вимоги до вебсайту.
3. Визначити ролі користувачів і сценарії взаємодії з системою.
4. Спроекувати логічну архітектуру з основними модулями та взаємозв'язками.
5. Розробити оптимізовану базу даних для швидкого пошуку й масштабування.
6. Створити прототип інтерфейсу, придатний для використання на різних пристроях.
7. Реалізувати вебсайт із повним переліком заявленого функціоналу.
8. Провести модульне, інтеграційне та користувацьке тестування й перевірити відповідність вимогам.
9. Оцінити ефективність рішення та визначити перспективи подальшого розвитку, зокрема інтеграцію API, мобільної версії та алгоритмів рекомендацій.

Об'єктом дослідження є процес пошуку, оцінювання та обговорення фільмів в інтернет-середовищі, а предметом — веб-технології й методи проєктування, що забезпечують створення інтерактивної кіно-платформи з персоналізацією та гнучким адмініструванням контенту.

Предметом дослідження є веборієнтовані технології для розробки вебдодатку для онлайн пошуку фільмів, що включає функцію оцінок, коментарів та списку улюблених фільмів.

					ІПЗк.КР.Б – 121 – 25 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ВЕБСАЙТУ «КІНОПОШУК»

1.1. Опис та постановка задачі

У сучасних реаліях кінематограф став не лише популярною формою розваг, а й важливою складовою культурного життя суспільства. Щороку виходить величезна кількість фільмів, що охоплюють різноманітні жанри та теми, а кількість людей, які активно цікавляться кіно, постійно зростає. Водночас розвиток цифрових технологій та Інтернету стимулює збільшення попиту на онлайн-сервіси, що надають інформацію про фільми, дозволяють їх оцінювати та обговорювати. Для сучасного кіномана швидкість, зручність та можливість самостійно контролювати процес пошуку інформації про кіно та формування власних уподобань стали базовими вимогами. Саме тому цифрові платформи посідають провідне місце у сфері доступу до кінематографічного контенту та інформації про нього.

Однак, попри значну кількість онлайн-платформ, присвячених кіно, користувачі все частіше стикаються з певними обмеженнями, які ускладнюють вибір фільмів для перегляду або отримання повної та об'єктивної інформації. Багато сервісів зосереджуються на великій кількості контенту, але не завжди надають достатньо зручних інструментів для детального пошуку за специфічними критеріями (наприклад, пошук за декількома акторами одночасно, за країною та роком випуску з урахуванням жанру, фільтрація за специфічними нагородами). Крім того, деякі відомі платформи можуть надавати перевагу більш комерційно успішним або популярним фільмам, унаслідок чого менш відомі, артхаусні чи нішеві стрічки мають менші шанси бути поміченими потенційними глядачами.

Окремої уваги потребує і питання якості та структурованості інформації. Не всі платформи забезпечують достатню повноту даних про фільми (наприклад, детальний акторський склад, інформація про знімальну групу, альтернативні

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

назви, наявність різних версій фільму), або ж інтерфейс може бути переважаним чи не інтуїтивним. З іншого боку, для підтримки актуальності та повноти бази даних необхідні ефективні інструменти для адміністраторів контенту, які дозволяють легко додавати, редагувати та оновлювати інформацію про фільми, жанри, персон тощо.

У таких умовах актуальною є розробка вебсайту "Кінопошук", який об'єднуватиме зручність розширеного пошуку, повноту та структурованість інформації, а також інтерактивні функції, такі як оцінювання та коментування. Такий вебсайт має створити комфортне середовище для користувачів, які шукають інформацію про кіно, та надати адміністраторам ефективні інструменти для управління контентом, сприяючи підвищенню якості та доступності кінематографічної інформації.

Для успішної реалізації вебсайту "Кінопошук" необхідно вирішити комплекс завдань, що охоплюють усі етапи розробки програмного продукту: від дослідження предметної області до впровадження та тестування готової системи. Розробка такого вебсайту вимагає врахування потреб різних категорій користувачів, забезпечення високої якості сервісу, функціональної повноти та захищеності даних. Водночас необхідно інтегрувати функціонал персоналізації (списки улюбленого, історія переглядів/оцінок), що підвищує зручність користування платформою та допомагає користувачам орієнтуватися у світі кіно.

Особливу увагу під час виконання роботи слід приділити аналізу існуючих рішень на ринку кінопорталів, визначенню ключових вимог до системи, проєктуванню бази даних, розробці архітектури та інтерфейсу користувача, а також реалізації безпечної та ефективної взаємодії між усіма компонентами платформи.

У межах даної кваліфікаційної роботи ставляться такі основні завдання:

1. Провести аналіз існуючих онлайн-платформ та вебсайтів, присвячених кіно, з метою виявлення їхніх сильних і слабких сторін, недоліків та можливостей для удосконалення.

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

2. Сформулювати вимоги до системи "Кінопошук", включаючи функціональні та нефункціональні вимоги, з урахуванням потреб кінцевих користувачів (глядачів) та адміністраторів сайту.

3. Визначити основні ролі користувачів системи (zareestrovaniy користувач, незареєстрований користувач, адміністратор) та описати сценарії їхньої взаємодії з вебсайтом (варіанти використання).

4. Розробити логічну структуру вебдодатку, визначивши основні модулі, компоненти та їх взаємозв'язки.

5. Спроекувати базу даних, що забезпечить зберігання, обробку та взаємодію всіх даних системи (інформація про фільми, жанри, акторів, режисерів, користувачів, відгуки), дотримуючись принципів нормалізації та забезпечуючи цілісність даних.

6. Розробити прототип інтерфейсу користувача (UI/UX) для вебсайту "Кінопошук", що буде зручним, інтуїтивно зрозумілим та візуально привабливим.

7. Реалізувати вебдодаток із використанням сучасних вебтехнологій, забезпечивши основні функції: реєстрацію та авторизацію користувачів, розширений пошук та фільтрацію фільмів, відображення детальної сторінки фільму, систему оцінювання та коментування, ведення профілю користувача зі списками улюблених, а також адміністративну панель для управління контентом (додавання/редагування фільмів, жанрів, акторів, модерація відгуків).

8. Провести комплексне тестування вебпорталу, перевіривши коректність роботи кожного модуля, загальну стабільність, рівень безпеки та відповідність визначеним вимогам.

9. Проаналізувати отримані результати, визначити ефективність створеної системи, окреслити її сильні сторони та орієнтовні напрями подальшого розвитку..

Враховуючи стрімке зростання попиту на онлайн-сервіси у сфері кіно, у проєкті зроблено акцент на створенні платформи, що поєднує швидкий пошук і

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

можливість порівнювати фільми. Сформульовані завдання охоплюють увесь життєвий цикл розробки — від аналізу аналогів і визначення вимог до реалізації, тестування та оцінки результатів, що забезпечує ґрунтовну основу для комплексного втілення задуму.

1.2. Дослідження, аналіз та порівняння функціональних можливостей аналогів програмного продукту

Для розробки конкурентоздатного вебсайту "Кінопошук" першочерговим завданням є ретельне вивчення рішень, які вже функціонують на ринку кінопорталів та спеціалізованих інформаційних ресурсів, присвячених кінематографу. Здійснення аналізу аналогічних платформ відкриває можливість для всебічної оцінки їхнього набору функцій, ідентифікації сильних та слабких аспектів, а також для точного визначення тих потреб користувацької аудиторії, які на поточний момент залишаються незадоволеними або задоволеними не повною мірою

Зіставлення функціональних характеристик існуючих систем сприяє виявленню найбільш популярних та затребуваних опцій, таких як розширені інструменти пошуку, системи формування рейтингів, механізми для публікацій. Такий аналіз також дозволяє дослідити різноманітні підходи до впровадження ключових сервісів, зокрема способів подання інформації про кінострічки та організації взаємодії між користувачами. Це, у свою чергу, допомагає визначити потенційні шляхи для вдосконалення загального користувацького досвіду. Більше того, глибокий огляд конкурентних програмних продуктів слугує фундаментальною основою для формування чітких вимог до власного вебсайту, що розробляється. Такий підхід дозволяє уникнути повторення недоліків, притаманних існуючим аналогам, та інтегрувати новітні функціональні можливості, які здатні забезпечити додаткову цінність для кінцевих споживачів контенту.

					ІПЗк.КР.Б – 121 – 25 – ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

Отже, виконання аналітичного огляду аналогів є невід'ємним та критично важливим етапом у процесі проектування. Цей етап забезпечує належне обґрунтування для всіх прийнятих технічних та концептуальних рішень, сприяє формуванню оптимально збалансованої структури майбутнього функціоналу та суттєво підвищує шанси на успішну реалізацію і запуск вебсайту "Кінопошук" у висококонкурентному середовищі сучасних інформаційних онлайн-сервісів.

IMDb (Internet Movie Database) (рис. 1.1) позиціонується як одна з найбільш авторитетних та широко використовуваних у глобальному масштабі онлайн-баз даних, що акумулює інформацію про світовий кінематограф, телевізійні програми, відеоігри та персон, задіяних у їх створенні [4].

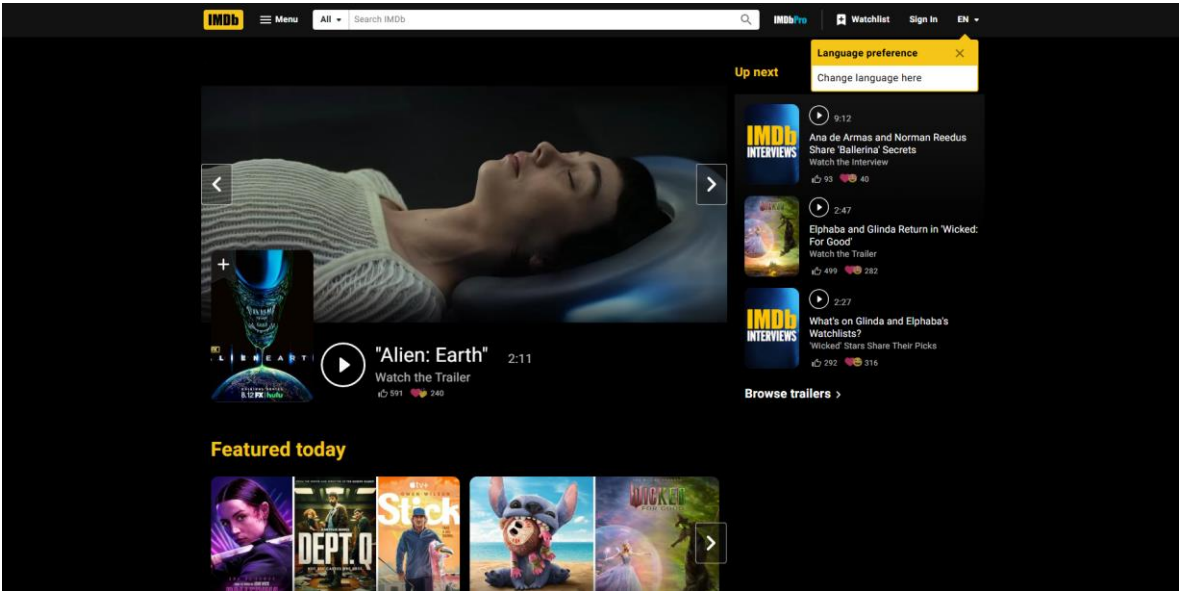


Рисунок 1.1 – Головна сторінка онлайн-платформи IMDb

Даний сервіс пропонує користувачам вичерпні відомості про фільми, включаючи детальний опис сюжетної лінії, повний перелік акторського складу та членів знімальної групи, точні дати прем'єрних показів, специфічні технічні характеристики, перелік отриманих нагород та номінацій. Окрім цього, платформа містить користувацькі рейтинги, професійні та аматорські рецензії, актуальні новини зі світу кіноіндустрії, офіційні трейлери та велику колекцію фотоматеріалів. Ключовою перевагою IMDb є колосальний обсяг накопичених даних та всеохопне глобальне покриття, що надає можливість знаходити

інформацію практично про будь-яку кінострічку чи особу, дотичну до сфери кіно.

Функціонал платформи характеризується значною широтою: він включає розгорнуті інформаційні сторінки, присвячені окремим фільмам та творчим особистостям, тематичні списки (наприклад, відомий "Top 250 Movies"), а також надає зареєстрованим користувачам інструментарій для формування індивідуальних списків для майбутнього перегляду (Watchlist), виставлення оцінок переглянутим фільмам та публікації власних відгуків. Вбудована система рейтингів, що комбінує середньозважені показники та індивідуальні користувацькі оцінки, разом із масивом рецензій, відіграє роль важливого допоміжного інструменту при прийнятті користувачем рішення щодо вибору фільму для перегляду. Інтерфейс вебверсії IMDb, попри свою інформаційну насиченість, демонструє відносну структурованість та забезпечує доступ до ключових функцій, серед яких пошук та можливість фільтрації результатів за множиною критеріїв (таких як жанрова приналежність, рік випуску, тип контенту).

Разом з тим, у функціональності IMDb присутні певні аспекти, які можна розглядати як потенційні недоліки або напрямки для подальшого вдосконалення. Одним із помітних викликів є ймовірність перевантаження інтерфейсу для частини користувачів, зумовлена надзвичайно великим обсягом представленої інформації та численними рекламними блоками, що, в свою чергу, може негативно позначатися на швидкості завантаження сторінок. Хоча пошукова система ресурсу є досить потужною, процес знаходження вузькоспеціалізованих даних або застосування складних комбінованих фільтрів іноді може виявитися не цілком інтуїтивним для користувачів без значного досвіду роботи з платформою. Можливості для персоналізації інтерфейсу та налаштування індивідуальних рекомендацій, хоча й наявні, могли б характеризуватися більшою гнучкістю та кращою адаптивністю до особистих уподобань кожного конкретного користувача. Також, незважаючи на величезну кількість

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

користувачьких відгуків, їхня якість може суттєво варіюватися, а система модерації контенту не завжди оперативно справляється з усім потоком нової інформації, що часом призводить до появи неконструктивних коментарів або спаму.

Підсумовуючи, IMDb є наочним прикладом масштабної та високоавторитетної платформи, яка володіє вичерпною базою даних та добре розвиненою екосистемою для генерування користувачького контенту. Проте, потенційні шляхи для покращення при розробці нового вебзастосунку можуть охоплювати створення більш інтуїтивного, "легшого" та менш обтяженого рекламними оголошеннями інтерфейсу, надання розширених опцій для персоналізації та фільтрації контенту, а також, можливо, імплементацію більш досконалих інструментів для проведення аналізу та порівняння кінофільмів за декількома параметрами одночасно.

Rotten Tomatoes (рис. 1.2) є широко визнаним вебсайтом, що спеціалізується на агрегації рецензій на кінофільми та телевізійні серіали [12].

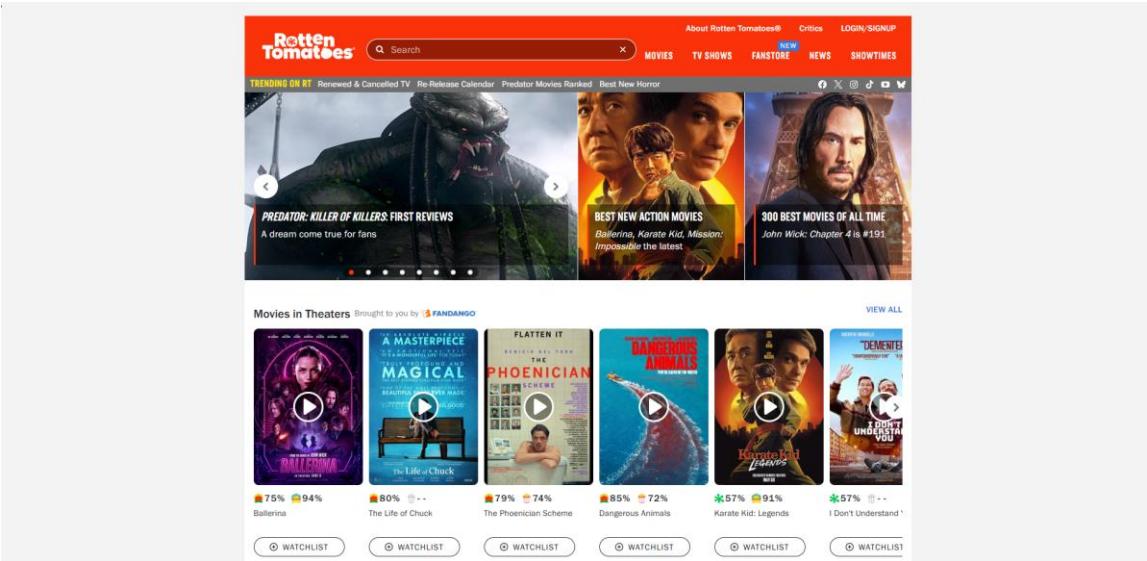


Рисунок 1.2 – Головна сторінка онлайн-платформи Rotten Tomatoes

Відмінною рисою цього сервісу виступає система "Томатометр" (Tomatometer), яка візуалізує відсоткове співвідношення позитивних відгуків, залишених професійними кінокритиками. Паралельно з цим існує окремий рейтинг, що формується на основі оцінок аудиторії (Audience Score). Платформа

здійснює збір рецензій з різноманітних джерел, включаючи друковані видання, інтернет-портали, телевізійні та радіопрограми, після чого класифікує їх на "свіжі" (позитивні) та "гнилі" (негативні), і на підставі цієї класифікації розраховує узагальнений рейтинг для кожного твору.

До переваг Rotten Tomatoes варто віднести наявність чіткої та легко зрозумілої системи оцінювання, яка дозволяє користувачам оперативно сформувати уявлення про те, як конкретний фільм сприймається професійними критиками та широкою глядацькою аудиторією. Вебсайт надає доступ до коротких уривків з рецензій, а також прямі посилання на їх повні тексти. Окрім цього, тут можна знайти інформацію про касові збори, дані про акторський склад та останні новини кіноіндустрії. Користувачам також надається можливість залишати власні оцінки (за 5-зірковою шкалою) та короткі текстові відгуки.

Тим не менш, незважаючи на свою значну популярність, Rotten Tomatoes також має низку певних обмежень. Система оцінювання "свіжий/гнилий" за своєю суттю є бінарною і не завжди здатна відобразити всі нюанси та відтінки рецензій – наприклад, критик міг надати фільму змішану, але в цілому схвальну оцінку (скажімо, 3 бали з 5), проте така оцінка все одно буде класифікована як "свіжа", аналогічно до захопленої рецензії з оцінкою 5 з 5. Це може призводити до надмірно спрощеного сприйняття фільмів і не завжди адекватно відображати ступінь позитивності чи негативності загального враження. Час від часу виникають публічні дискусії стосовно можливого впливу "Томатометра" на показники касових зборів та на формування глядацьких очікувань, а також лунають звинувачення у потенційній маніпуляції рейтингами. Рівень деталізації інформації про сам фільм (зокрема, технічні аспекти виробництва, повний перелік членів знімальної групи, детальні біографії акторів) значно поступається аналогічним показникам на IMDb. Функціонал пошуку та фільтрації контенту також може виявитися менш гнучким у порівнянні з більш спеціалізованими базами даних, часто обмежуючись лише пошуком за назвою, жанром та роком випуску.

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

Таким чином, Rotten Tomatoes є ефективним інструментом для швидкого отримання узагальненої оцінки фільмів на основі агрегованих рецензій. Однак, його система оцінювання не завжди може бути достатньою для проведення глибокого та всебічного аналізу, а ступінь деталізації інформації про кінострічки є обмеженим. Це, в свою чергу, створює певну ринкову нішу для платформ, здатних запропонувати користувачам більш розширені аналітичні можливості, більш детальну та повну інформацію, а також, потенційно, альтернативні підходи до агрегації та представлення оцінок.

Letterboxd (рис. 1.3) функціонує як соціальна інтернет-платформа, що орієнтована на спільноту кіноманів та надає їм інструменти для відстеження переглянутих фільмів, написання власних рецензій, створення тематичних списків, а також для активної взаємодії з іншими користувачами ресурсу [8].

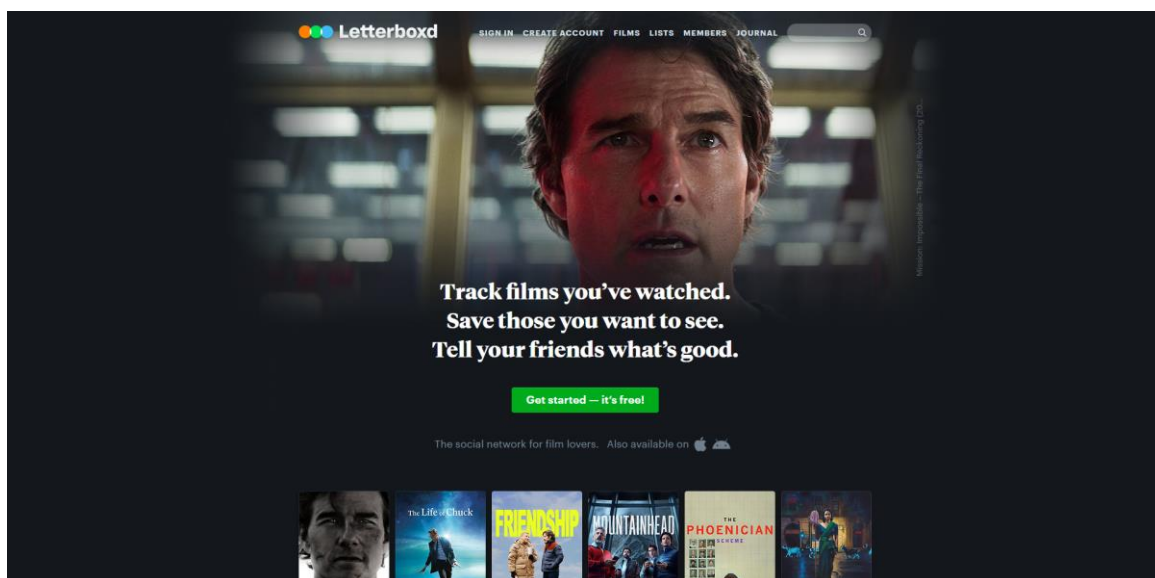


Рисунок 1.3 – Головна сторінка онлайн-платформи Letterboxd

На відміну від IMDb, який за своєю суттю є перш за все масштабною базою даних, та Rotten Tomatoes, що концентрується на агрегації професійних та користувацьких рецензій, Letterboxd робить основний акцент на індивідуальному досвіді перегляду кіно та на соціальній взаємодії, що розгортається навколо кінематографічної тематики.

Ця платформа вирізняється своїм мінімалістичним та естетично привабливим дизайном. Вона надає користувачам можливість вести

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

деталізований щоденник переглядів (із зазначенням точної дати перегляду, виставленням оцінки за стандартною 5-зірковою шкалою, а також можливістю відмітити фільм як такий, що "сподобався"), створювати та поширювати серед інших учасників спільноти тематичні списки фільмів, а також стежити за активністю друзів та інших користувачів, що мають схожі кінематографічні смаки. Для наповнення своєї бази даних інформацією про фільми Letterboxd активно використовує дані, надані проектом TMDb (The Movie Database), що забезпечує високий рівень інформаційної насиченості ресурсу.

Проте, незважаючи на значні переваги, які Letterboxd пропонує для спільноти ентузіастів кіно, ця платформа також має певні характерні обмеження. Оскільки вона більшою мірою позиціонується як соціальна мережа, аніж як енциклопедичний ресурс, глибина та детальність інформації про окремих акторів, режисерів або специфічні технічні аспекти виробництва фільмів тут є меншою, ніж на IMDb. Спеціалізований новинний розділ або інформація про касові збори на платформі практично відсутні. Хоча Letterboxd пропонує безкоштовну версію для користування, деякі розширені функціональні можливості (наприклад, доступ до детальної статистики переглядів, можливість фільтрації стрічки активності друзів без показу реклами) доступні виключно в рамках платних підписок Pro або Patron. Для користувачів, основною метою яких є пошук виключно об'єктивної інформації та професійних рецензій, Letterboxd може видатися менш корисним інструментом, ніж спеціалізовані агрегатори рецензій чи великі бази даних.

Серед функціональних обмежень варто відзначити менш розвинені інструменти для здійснення пошуку фільмів за дуже специфічними та комбінованими критеріями (наприклад, пошук за поєднанням декількох акторів, країни виробництва та конкретного року випуску в одному пошуковому запиті) у порівнянні з можливостями, що надаються IMDb. Основний фокус платформи спрямований на індивідуальний користувацький досвід та на обмін думками й враженнями всередині спільноти.

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

Отже, Letterboxd пропонує унікальний соціальний досвід для поціновувачів кіно, концентруючись на особистих враженнях та взаємодії, однак може поступатися іншим платформам у глибині енциклопедичної інформації або наявності інструментів для професійного аналізу кіноринку. Цей факт вказує на потенційну можливість створення такого сервісу, який би успішно поєднував інформаційну повноту та насиченість з інтерактивними та соціальними функціями.

1.3. Вибір архітектури вебдодатку

Визначення оптимальної архітектурної моделі є фундаментальним кроком у процесі розробки будь-якої сучасної програмної системи. Обрана архітектура безпосередньо впливає на ефективність взаємодії між окремими компонентами застосунку, його здатність до масштабування, загальну стабільність функціонування та можливості для подальшого розвитку й модернізації. Враховуючи специфіку функціональних вимог, що висуваються до вебсайту "Кінопошук", було ухвалено стратегічне рішення щодо імплементації архітектури клієнт-серверного типу. Ця модель передбачає чіткий розподіл функціональних обов'язків між клієнтською частиною (фронтенд) та серверною частиною (бекенд), які здійснюють обмін даними через стандартизований інтерфейс RESTful API. Такий підхід забезпечує високий рівень гнучкості системи, значно полегшує процеси її технічної підтримки та супроводу, а також надає можливість незалежно масштабувати окремі її складові відповідно до зростаючих потреб або навантаження. [17]

Під час попереднього аналізу було розглянуто три альтернативні архітектурні підходи. Спершу увагу звернули на монолітну архітектуру, де всі функціональні компоненти об'єднані в один застосунок: таке рішення спрощує початкову розробку й розгортання, проте обмежує масштабованість, ускладнює внесення змін і підвищує складність підтримки великих проєктів. Далі вивчили клієнт-серверну модель із чітким розділенням фронтенду й бекенду, які

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

взаємодіють через API; ця схема забезпечує гнучкість, дає змогу командам працювати паралельно й дозволяє масштабувати окремі частини системи незалежно. Нарешті, проаналізували мікросервісну архітектуру, що розбиває систему на невеликі автономні сервіси з власними бізнес-функціями; попри значну складність налаштування, моніторингу та оркестрації, вона відкриває найширші можливості для горизонтального масштабування та незалежного розвитку кожного модуля.

Беручи до уваги сукупність функціональних та нефункціональних вимог, сформульованих для системи "Кінопошук", а також враховуючи потенційні обмеження у часі та доступних ресурсах, для розробки було зупинено вибір на клієнт-серверній архітектурі з використанням RESTful API як найбільш збалансованому та оптимальному рішенні.

Обґрунтування вибору саме клієнт-серверної архітектури із застосуванням RESTful API ґрунтується на необхідності чіткого розмежування відповідальності між клієнтською (фронтенд) та серверною (бекенд) частинами. Таке розмежування є ключовим для забезпечення гнучкості розробки та можливості ефективного масштабування системи в майбутньому [11]. Цей підхід дозволяє здійснювати незалежне оновлення та модифікацію користувацького інтерфейсу та серверної логіки без ризику негативного впливу на інші компоненти системи. Додатково, застосування REST API як стандартизованого протоколу взаємодії відкриває перспективи для потенційної інтеграції з іншими зовнішніми сервісами та платформами, а також для розробки альтернативних клієнтських додатків (наприклад, мобільних) у майбутньому.

Використання сучасного технологічного стеку, що включає фреймворк Laravel (PHP) для серверної частини [1], бібліотеку React для клієнтської частини та систему управління базами даних MariaDB, дозволяє побудувати надійну, безпечну та високопродуктивну систему. Застосування технології контейнеризації Docker забезпечує стабільність роботи вебсайту незалежно від специфіки середовища розгортання та значно спрощує цей процес.

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

Архітектурна модель вебсайту "Кінопошук" реалізована на принципах клієнт-серверної взаємодії та передбачає логічне розділення системи на наступні ключові компоненти:

1. Frontend (Клієнтська частина): Реалізований як односторінковий застосунок (Single Page Application – SPA) з використанням бібліотеки React. Цей компонент відповідає за всю візуальну складову, взаємодію з кінцевим користувачем, відображення отриманих даних та формування й надсилання запитів до серверної частини через REST API.
2. Backend (Серверна частина): Розроблений на базі PHP-фреймворку Laravel. Цей компонент інкапсулює всю основну бізнес-логіку додатку, обробляє запити від клієнтської частини, здійснює взаємодію з базою даних MariaDB, управляє зберіганням та доступом до файлів (постерів, зображень акторів тощо) безпосередньо у файловій системі операційної системи сервера.
3. База даних (Database): В якості системи управління базами даних (СУБД) використовується MariaDB. Вона призначена для постійного зберігання основних структурованих даних вебсайту, таких як інформація про користувачів, фільми, жанри, акторів, режисерів, виставлені оцінки, залишені відгуки, списки улюблених фільмів, а також шляхи до відповідних медіафайлів у файловій системі.
4. Зберігання файлів (File Storage): Для зберігання різноманітних медіафайлів, таких як зображення постерів фільмів, фотографії акторів, режисерів та інші графічні матеріали, використовується стандартна файлова система операційної системи сервера, на якому функціонує бекенд-додаток. Бекенд відповідає за організацію структури каталогів та управління доступом до цих файлів[20].

Усі перераховані сервіси та компоненти системи розгортаються та функціонують у Docker-контейнерах (або принаймні бекенд та фронтенд). Такий підхід забезпечує їх повну ізоляцію один від одного, значно полегшує процес

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

налаштування робочого оточення, автоматизує розгортання застосунку та спрощує процедури масштабування системи в перспективі [5].

Клієнтська (фронтенд) та серверна (бекенд) частини функціонують як два незалежні сервіси. Обмін даними між ними здійснюється виключно через стандартизовані HTTP-запити до чітко визначених ендпоінтів RESTful API. Таке архітектурне рішення дозволяє значно підвищити гнучкість системи в цілому, спрощує процес модифікації та розвитку окремих її компонентів, а також відкриває широкі можливості для інтеграції з іншими потенційними клієнтськими застосунками або сторонніми сервісами. Обрана архітектура повною мірою відповідає актуальним тенденціям у сфері веброзробки та завданням, що стоять перед інформаційно-пошуковим вебсайтом "Кінопошук".

1.4. Обґрунтування вибору інструментальних засобів для розробки

З огляду на специфічні функціональні потреби, що висувуються до вебсайту "Кінопошук", а також беручи до уваги вимоги щодо гнучкості розробки, забезпечення належного рівня безпеки та потенційних інтеграційних можливостей, було здійснено вибір на користь сучасного та перевіреного набору технологій. Цей технологічний стек спроможний забезпечити ефективну реалізацію усіх запланованих аспектів та функціональних блоків системи.

Для розробки серверної частини (backend) було обрано PHP-фреймворк Laravel. На сьогодні Laravel є одним із найбільш затребуваних та потужних фреймворків у середовищі PHP-розробки. Він надає розробникам багатий та добре структурований набір інструментів для прискореного створення вебзастосунків. Зокрема, Laravel пропонує елегантні рішення для реалізації маршрутизації запитів, вбудовані механізми аутентифікації та авторизації користувачів, зручну роботу з базами даних через об'єктно-реляційний картограф (ORM) Eloquent, ефективну обробку запитів до REST API та гнучку систему проміжних обробників (middleware) [6]. Окрім цього, Laravel відомий своєю архітектурою, що сприяє розробці масштабованих вебрішень. Ця

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

характеристика є критично важливою для інформаційно-пошукового порталу "Кінопошук", який потенційно має обробляти значну кількість запитів від користувачів, пов'язаних із пошуком фільмів, отриманням детальної інформації, переглядом рейтингів та відгуків. Laravel також забезпечує належні інструменти для управління файлами (наприклад, постерами фільмів, зображеннями акторів), що зберігатимуться у файловій системі сервера.

Клієнтська частина (frontend) вебсайту реалізована з використанням JavaScript-бібліотеки React. Такий вибір дозволяє створити сучасний односторінковий застосунок (Single Page Application – SPA), що характеризується динамічним, інтерактивним та відгукливим користувацьким інтерфейсом. React забезпечує високу швидкодію інтерфейсу завдяки використанню концепції віртуального DOM (Document Object Model). До того ж, React пропагує компонентний підхід до розробки, що значно полегшує структурування коду, його повторне використання, а також подальше розширення та технічне обслуговування інтерфейсу користувача [21]. Вибір React обґрунтований необхідністю створення гнучкого, інтуїтивно зрозумілого та зручного у використанні інтерфейсу. Це дозволить користувачам швидко та ефективно здійснювати пошук фільмів за різноманітними критеріями, переглядати детальну інформацію, взаємодіяти з системою рейтингів та відгуків, а також керувати власними профілями та списками улюблених фільмів.

В якості системи управління базами даних (СУБД) було обрано MariaDB. Це надійна, перевірена часом та широко використовувана реляційна СУБД, яка гарантує стабільну роботу застосунку, підтримує ACID-транзакції, забезпечує високий рівень сумісності з мовою програмування PHP та фреймворком Laravel, а також демонструє хорошу масштабованість для ефективної обробки значних обсягів даних [9]. MariaDB дозволяє оптимально зберігати структуровану інформацію, таку як дані про користувачів, детальну інформацію про фільми (назви, описи, роки випуску, країни виробництва), жанри, акторський склад, режисерів, користувацькі оцінки, текстові відгуки, списки улюблених фільмів, а

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

також шляхи до відповідних медіафайлів, що зберігаються у файловій системі сервера.

Для адміністрування бази даних MariaDB та зручної роботи з нею використовується вебзастосунок phpMyAdmin. Це популярне рішення надає графічний вебінтерфейс для виконання широкого спектра операцій з даними, включаючи створення та модифікацію таблиць, налаштування зв'язків між ними, виконання довільних SQL-запитів, а також здійснення процедур резервного копіювання та відновлення даних [13].

Невід'ємною та важливою частиною сучасної інфраструктури розроблюваного вебсайту є застосування технології контейнеризації Docker. Ця технологія використовується для розгортання всіх ключових компонентів системи. Кожен сервіс (зокрема, бекенд на Laravel, фронтенд на React, база даних MariaDB, інструмент адміністрування phpMyAdmin, а також потенційно SMTP-сервер, якщо він локальний) розгортається в окремому ізольованому Docker-контейнері. Такий підхід забезпечує ізоляцію сервісів один від одного, гарантує відтворюваність робочого середовища на різних машинах та значно полегшує масштабування системи в майбутньому. Docker дозволяє стандартизувати процес налаштування інфраструктури, спрощує оновлення окремих компонентів та розгортання всього застосунку на різноманітних хостингових платформах [3].

Таким чином, обраний комплекс інструментальних засобів повною мірою відповідає специфічним вимогам, що висуваються до інформаційно-пошукового вебсайту "Кінопошук". Він забезпечує створення комплексного програмного рішення з високим рівнем продуктивності, надійності, безпеки та масштабованості, а також підтримує сучасні стандарти та найкращі практики у сфері веброзробки.

Висновки до першого розділу

Дослідження, проведене в рамках першого розділу даної кваліфікаційної роботи, надало можливість здійснити системну оцінку ключових передумов,

					ІІЗк.КР.Б – 121 – 25 – ІЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

актуальних потреб користувачів та сучасних тенденцій у сфері онлайн-сервісів, присвячених кінематографу. Ці фактори сукупно формують комплекс вимог до розроблюваного вебсайту "Кінопошук". Було здійснено аналіз як загальносвітових процесів цифровізації доступу до інформації про кіно, так і специфічних викликів, пов'язаних із необхідністю агрегації великих обсягів даних, забезпечення зручного пошуку та надання користувачам інтерактивних можливостей.

Результати проведеного аналізу переконливо демонструють, що сучасний користувач інформаційних ресурсів про кіно очікує від цифрових сервісів не лише базових функцій, таких як пошук фільмів за назвою. Існує чіткий запит на розширені можливості, зокрема, на детальний пошук за множиною критеріїв (жанр, рік, актори, режисери, рейтинг), доступ до вичерпної інформації про кінострічки, а також на інструменти для оцінювання, коментування та формування персоналізованих списків. Багато існуючих платформ, хоча й пропонують значний обсяг контенту, не завжди забезпечують оптимальний користувацький досвід через перевантаженість інтерфейсів або обмежену гнучкість пошукових механізмів. Це створює об'єктивну потребу в інноваційних рішеннях, які б поєднували високий рівень юзабіліті, зручність взаємодії та багатство функціоналу.

Аналіз функціональних можливостей вебсайтів-аналогів, таких як IMDb, Rotten Tomatoes та Letterboxd, показав, що, незважаючи на їхній потужний та усталений функціонал, кожен із них має певні недоліки або особливості, які можуть впливати на зручність користування, повноту чи об'єктивність представленої інформації. Зокрема, було виявлено обмеження щодо гнучкості фільтрації при складному пошуку, різний ступінь деталізації інформації про фільми та персон, а також відмінності в підходах до систем рейтингування та соціальної взаємодії. Ці аспекти чітко вказують на наявність вільної ніші для розробки нових рішень, здатних запропонувати користувачам збалансоване

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

поєднання інформаційної насиченості, гнучкості інструментів взаємодії та сучасного користувацького досвіду.

У підсумку виконаної роботи було чітко сформульовано перелік основних завдань, вирішення яких є необхідним для створення конкурентоспроможного вебсайту "Кінопошук", що відповідатиме актуальним запитам сучасної аудиторії. Особливий акцент було зроблено на необхідності реалізації розширеного пошуку та фільтрації, забезпечення повноти та достовірності інформації про фільми, інтеграції зручної системи оцінювання та коментування, а також надання користувачам інструментів для персоналізації (профілі, списки улюблених) та адміністраторам – ефективних засобів для управління контентом.

Також у першому розділі було детально обґрунтовано вибір архітектурного підходу для майбутнього вебсайту. Було прийнято рішення на користь клієнт-серверної моделі з чітким розділенням на фронтенд (клієнтську частину) та бекенд (серверну частину), які взаємодіятимуть між собою через RESTful API. Така архітектура забезпечує необхідну гнучкість для масштабування системи, можливість інтеграції з потенційними сторонніми сервісами та загальну зручність технічного супроводу й подальшого розвитку проєкту. Використання сучасного та перевіреного технологічного стеку, що включає Docker для контейнеризації, Laravel (PHP) для бекенду, MariaDB для бази даних та React для фронтенду, дозволяє створити надійну, продуктивну та безпечну систему, добре адаптовану до потреб різних категорій користувачів та здатну ефективно вирішувати поставлені завдання.

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБСАЙТУ «КІНОПОШУК»

2.1. Визначення варіантів використання та структури системи

Для ефективної розробки вебсайту "Кінопошук" критично важливим є точне окреслення сценаріїв взаємодії різних категорій користувачів із платформою, а також формування її чіткої логічної структури. Такий підхід дозволить не лише забезпечити відповідність функціоналу очікуванням кінцевих користувачів, але й закласти основи для створення гнучкої, масштабованої та продуктивної архітектури майбутнього інформаційного ресурсу.

Процес проєктування будь-якого інформаційного ресурсу, яким є і "Кінопошук", традиційно розпочинається з детального визначення його функціонального призначення та способів, якими користувачі будуть взаємодіяти з наданими можливостями. Одним із найбільш дієвих та поширених методів для опису таких взаємодій є концепція варіантів використання (use cases). Варіанти використання являють собою формалізований опис сценаріїв, які наочно демонструють, як різні типи користувачів (що в термінології UML називаються акторами або дійовими особами) взаємодіють із системою для досягнення своїх конкретних цілей та завдань [15]. Вони відображають функціональність системи саме «з погляду користувача», дозволяючи чітко ідентифікувати очікувані дії, передбачувані результати та необхідні умови для успішного виконання тих чи інших операцій.

Застосування діаграм варіантів використання на стадії проєктування має суттєве значення, адже це дозволяє застосувати системний та структурований підхід до процесу створення програмного забезпечення. Такий підхід надає низку переваг, зокрема:

- дозволяє визначити повний спектр необхідних функцій системи відповідно до специфічних ролей та потреб користувачів;
- допомагає уникнути можливих пропусків у функціоналі або, навпаки, його надмірного дублювання;

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

- спрощує комунікацію та взаєморозуміння між командою розробників та потенційними кінцевими користувачами завдяки чіткому та зрозумілому опису принципів роботи системи;
- закладає міцну основу для подальшого формування технічного завдання, планування етапів розробки та тестування.

Вебсайт "Кінопошук" розробляється для задоволення потреб декількох груп користувачів, кожна з яких матиме власні цілі, рівні доступу та набір доступних функцій. У рамках системи передбачається три ключові ролі:

1. Неавторизований користувач (Гість)
2. Авторизований користувач
3. Адміністратор.

Кожна з цих ролей буде взаємодіяти з системою відповідно до своїх специфічних потреб та функціонального призначення. Для кожної ролі визначено типові сценарії взаємодії, які згодом будуть візуалізовані у вигляді відповідних діаграм варіантів використання.

Неавторизований користувач (Гість) – це будь-яка особа, яка відвідує вебсайт "Кінопошук" без проходження процедури реєстрації та авторизації. Основна мета такого користувача – ознайомлення з контентом сайту та пошук інформації про фільми.

Основні функції Неавторизованого користувача (Гостя) (рис. 2.1):

- Перегляд головної сторінки (доступ до списків нових, популярних, рекомендованих фільмів).
- Використання пошукового рядка для базового пошуку фільмів.
- Розширений пошук фільмів за різними критеріями (назва, режисер, актор, жанр, рік випуску).
- Фільтрація результатів пошуку (наприклад, за рейтингом, жанрами).
- Перегляд детальної сторінки фільму (опис, постер, трейлер, рік, країна, жанри, рейтинг, актори, режисер, відгуки інших користувачів).

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		



Рисунок 2.1 – Діаграма варіантів використання для Неавторизованого користувача (Гостя)

Авторизований користувач – це зареєстрований учасник платформи "Кінопошук", який успішно пройшов процедуру автентифікації (входу в систему). Ця роль передбачає доступ до всіх функцій Гостя, а також до додаткових інтерактивних можливостей, пов'язаних з персоналізацією досвіду та участю у житті спільноти сайту.

Основні функції Авторизованого користувача (рис. 2.2):

- Реєстрація нового облікового запису в системі.
- Авторизація (вхід) у систему з використанням існуючих облікових даних.
- Усі функції, доступні Неавторизованому користувачеві (Гостю).
- Можливість оцінити фільм (виставлення рейтингової оцінки).
- Додавання власних відгуків та коментарів до фільмів.
- Редагування та видалення власних відгуків/коментарів.
- Формування та управління особистим списком "Улюблені фільми"
- Перегляд власного профілю користувача
- Редагування даних власного профілю

Змн.	Арк.	№ докум.	Підпис	Дата



Рисунок 2.2 – Діаграма варіантів використання для Авторизованого користувача

Адміністратор – це спеціалізований користувач з найвищим рівнем повноважень у системі "Кінопошук". Адміністратор відповідає за управління контентом сайту, модерацію активності користувачів та забезпечення загальної стабільної та коректної роботи платформи.

Основні функції Адміністратора (рис. 2.3):

- Авторизація в адміністративній панелі системи.
- Управління каталогом фільмів: додавання нових фільмів (введення назви, опису, завантаження постера, зазначення року, режисера, посилання на трейлер), редагування інформації про існуючі фільми, видалення фільмів з бази даних.
- Управління довідником жанрів: додавання нових жанрів, редагування назв існуючих жанрів, видалення жанрів.
- Управління інформацією про акторів та режисерів: додавання нових персон (акторів, режисерів), редагування біографічної інформації, фотографій, видалення інформації про персон.
- Модерація контенту, створеного користувачами: видалення відгуків/коментарів, що порушують правила сайту.

					ІПЗк.КР.Б – 121 – 25 – ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		



Рисунок 2.3 – Діаграма варіантів використання для Адміністратора

Змн.	Арк.	№ докум.	Підпис	Дата

2.2. Проектування та розробка бази даних вебдодатку

Етап проектування та подальшої розробки структури бази даних посідає центральне місце у процесі створення будь-якого сучасного інформаційного вебсайту, і "Кінопошук" не є винятком. Саме база даних (БД) формує фундамент для логічного структурування, зберігання та ефективної обробки всієї інформації, з якою буде оперувати програмне забезпечення. Якісно та продумано спроектована база даних є запорукою стабільної роботи всієї системи, забезпечує оперативний доступ до необхідних відомостей та спрощує подальше обслуговування й розвиток проєкту.[27]

Раціональний підхід до проектування структури бази даних суттєво підвищує її адаптивність та закладає потенціал для майбутнього масштабування. Завдяки цьому в перспективі можна буде відносно безперешкодно розширювати функціональні можливості вебсайту "Кінопошук" відповідно до змін у користувацьких вимогах, технологічних трендах чи ринкових умовах його експлуатації.

Для реалізації бази даних у межах даного проєкту було обрано систему управління базами даних (СУБД) MariaDB. Таке рішення обумовлене низкою її вагомих переваг. По-перше, MariaDB є однією з найпоширеніших реляційних СУБД у світі, вона активно використовується в проєктах абсолютно різного масштабу – від невеликих вебсайтів до великих корпоративних систем. Активна підтримка з боку великої спільноти розробників гарантує її стабільність, надійність та постійний розвиток.

MariaDB відома своєю високою продуктивністю, особливо при роботі з операціями читання, що є типовим для інформаційних порталів, та хорошою здатністю до масштабування, що є важливим для вебсайтів, які потенційно працюватимуть зі значними обсягами даних та великою кількістю користувацьких запитів [7]. Крім того, MariaDB надає розробникам гнучкі та розвинені засоби для управління правами доступу до даних, що дозволяє забезпечити належний рівень безпеки інформації, яка зберігається. Усі ці

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

фактори в сукупності роблять MariaDB оптимальним вибором для створення надійної, масштабованої та безпечної бази даних для вебсайту "Кінопошук".

У процесі проєктування структури БД були враховані всі ключові функціональні вимоги, що висувуються до вебсайту "Кінопошук". Це дозволило сформувати логічно впорядковану, нормалізовану та добре організовану структуру бази даних, яка буде представлена у вигляді ER-діаграми (рис. 2.4).

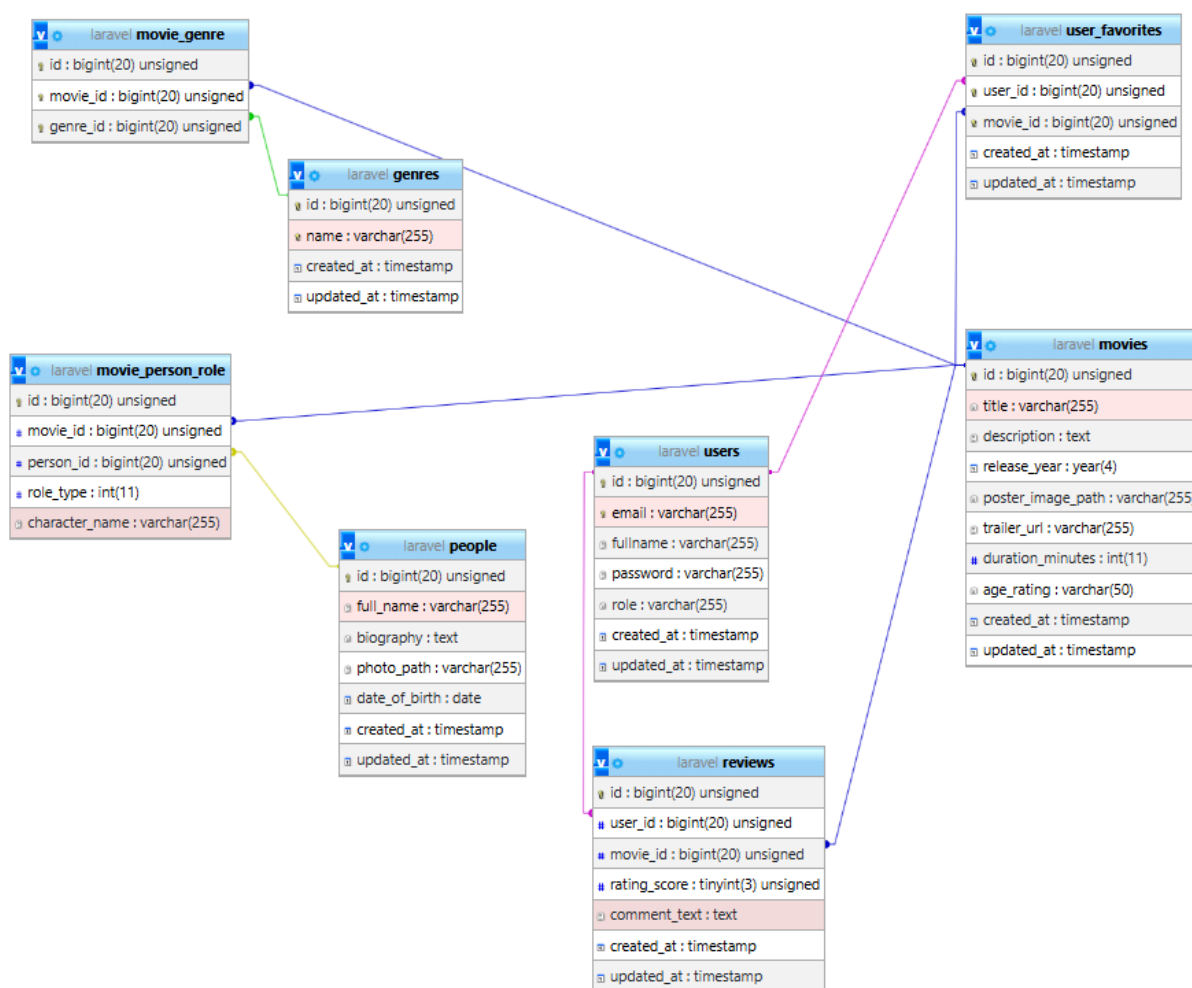


Рисунок 2.4 – Структура бази даних вебдодатку

Таблиця users відповідає за зберігання облікових записів користувачів вебсайту (табл. 2.1). Вона містить дані про електронну пошту, хешований пароль, ім'я користувача, його роль у системі а також службові позначки часу.

Таблиця 2.1

Таблиця users

Назва поля	Тип даних	РК	FK	Примітка
id	int	+	-	унікальний ідентифікатор
email	varchar(255)	-	-	електронна пошта
fullname	varchar(255)	-	-	прізвище та ім'я
password	varchar(255)	-	-	пароль
role	varchar(255)	-	-	роль
created_at	timestamp	-	-	коли створено користувача
updated_at	timestamp	-	-	час останнього оновлення

Таблиця movies містить детальну інформацію про фільми, представлені на сайті (табл. 2.2).

Таблиця 2.2

Таблиця movies

Назва поля	Тип даних	РК	FK	Примітка
id	int	+	-	унікальний ідентифікатор
title	varchar(255)	-	-	назва фільму
description	text	-	-	детальний опис фільму
release_year	int	-	-	рік випуску фільму
poster_image_path	varchar(255)	-	-	шлях до файлу постера
trailer_url	varchar(255)	-	-	посилання на трейлер
duration_minutes	int	-	-	Тривалість в хвилинах
age_rating	varchar(50)	-	-	віковий рейтинг
created_at	timestamp	-	-	час створення
updated_at	timestamp	-	-	час останнього оновлення

Таблиця genres слугує довідником для всіх жанрів фільмів (табл. 2.3).

Таблиця 2.3

Таблиця genres

Назва поля	Тип даних	РК	FK	Примітка
id	int	+	-	унікальний ідентифікатор
name	varchar(255)	-	-	назва жанру
created_at	timestamp	-	-	час створення
updated_at	timestamp	-	-	час останнього оновлення

Таблиця people призначена для зберігання інформації про персон, пов'язаних з кіно (табл. 2.4).

Таблиця 2.4

Таблиця people

Назва поля	Тип даних	РК	FK	Примітка
id	int	+	-	унікальний ідентифікатор
full_name	varchar(255)	-	-	повне ім'я персони
biography	text	-	-	біографія
photo_path	varchar(255)	-	-	шлях до фотографії
date_of_birth	date	-	-	дата народження
created_at	timestamp	-	-	час створення
updated_at	timestamp	-	-	час останнього оновлення

Таблиця movier_genre є проміжною (зв'язуючою) таблицею для реалізації відношення "багато-до-багатьох" між фільмами та жанрами (табл. 2.5).

Таблиця 2.5

Таблиця movie_genre

Назва поля	Тип даних	РК	FK	Примітка
id	int	+	-	унікальний ідентифікатор
movie_id	int	-	+	ідентифікатор фільму
genre_id	int	-	+	ідентифікатор жанру

Таблиця movie_person_role є проміжною таблицею для реалізації відношення "багато-до-багатьох" між фільмами та персонами, із зазначенням ролі персони у фільмі (табл. 2.6).

Таблиця 2.6

Таблиця movie_person_role

Назва поля	Тип даних	РК	FK	Примітка
id	int	+	-	унікальний ідентифікатор
movie_id	int	-	+	ідентифікатор фільму
person_id	int	-	+	ідентифікатор персони
role_type	int	-	-	тип ролі персони у фільмі
character_name	varchar(255)	-	-	ім'я персонажа

Таблиця reviews акумулює відгуки та оцінки користувачів про фільми (табл. 2.7).

Таблиця 2.7

Таблиця reviews

Назва поля	Тип даних	РК	FK	Примітка
id	int	+	-	унікальний ідентифікатор
user_id	int	-	+	ідентифікатор користувача
movie_id	int	-	+	ідентифікатор фільму
rating_score	tinyint	-	-	оцінка фільму
comment_text	text	-	-	текстовий коментар
created_at	timestamp	-	-	час створення
updated_at	timestamp	-	-	час останнього оновлення

Таблиця user_favorites зберігає інформацію про фільми, які користувачі додали до свого списку улюблених (табл. 2.8).

Таблиця user_favorites

Назва поля	Тип даних	РК	FK	Примітка
id	int	+	-	унікальний ідентифікатор
user_id	int	-	+	ідентифікатор користувача
movie_id	int	-	+	ідентифікатор фільму
created_at	timestamp	-	-	час створення

2.3. Реалізація функціоналу роботи вебдодатку для вебсайту «Кінопошук»

Після детального визначення варіантів використання системи, формування вичерпного переліку вимог, проєктування структури бази даних та попереднього аналізу ключових алгоритмів, що лежатимуть в основі роботи вебсайту "Кінопошук", було розпочато етап безпосередньої практичної реалізації програмного продукту. У цьому підрозділі докладно розглядається процес створення та налаштування основних функціональних модулів системи, що забезпечують її ключові можливості.

Для забезпечення функціоналу завантаження, зберігання та відображення медіафайлів, таких як постери до фільмів та фотографії акторів/режисерів, у системі "Кінопошук" реалізовано спеціалізований сервіс, умовно названий FileManagementService. Цей сервіс інкапсулює логіку взаємодії з файловою системою сервера, де розгорнуто бекенд-додаток. Він відповідає за безпечне завантаження файлів, генерацію унікальних імен для уникнення колізій, збереження файлів у визначених директоріях, формування публічно доступних URL-адрес для їх відображення на клієнтській частині, а також за видалення файлів, коли вони більше не потрібні (наприклад, при видаленні фільму або оновленні постера).

Основний клас сервісу має наступну структуру:

```
class FileManagementService
{
```

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        public function uploadImage(UploadedFile $file, string $directory):
?string;
        public function getFileUrl(?string $path): ?string;
        public function deleteImage(?string $path): bool;
    }

```

Метод `uploadImage()` приймає завантажений файл (наприклад, об'єкт `UploadedFile` у Laravel), валідує його генерує унікальне ім'я, зберігає у відповідну піддиректорію та повертає відносний шлях до збереженого файлу або інформацію про успіх/помилку операції:

```

        public function uploadImage(UploadedFile $file, string $directory):
?string
        {
            $filename = uniqid() . '_' . time() . '.' . $file-
>getClientOriginalExtension();
            $path = $file->storeAs('public/' . $directory, $filename);
            if (!$path) {
                return null;
            }
            return $directory . '/' . $filename;
        }

```

Метод `getFileUrl()` приймає відносний шлях до файлу та генерує повний URL, за яким цей файл буде доступний через вебсервер для відображення у браузері користувача:

```

        public function getFileUrl(?string $path): ?string
        {
            if (!$path) {
                return null;
            }
            return Storage::url($path);
        }

```

Метод `deleteImage()` приймає шлях до файлу, перевіряє його існування у файловій системі та видаляє його, повертаючи результат операції:

```

        public function deleteImage(?string $path): bool
        {
            if (!$path) {
                return true;
            }
            $fullPath = 'public/' . $path;
            if (Storage::exists($fullPath)) {
                return Storage::delete($fullPath);
            }
            return true;
        }

```

Таким чином, сервіс `FileManagementService` слугує централізованим інтерфейсом для всіх операцій, пов'язаних із медіафайлами, забезпечуючи

					ІІЗк.КР.Б – 121 – 25 – ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

уніфікований підхід та спрощуючи управління контентом у системі "Кінопошук".

Механізми аутентифікації (перевірки особи користувача) та авторизації (надання прав доступу) у системі "Кінопошук" реалізовані через спеціалізований контролер, наприклад, AuthController. Цей контролер відповідає за обробку запитів, пов'язаних зі створенням нових облікових записів користувачів, їх входом до системи, отриманням інформації про поточного аутентифікованого користувача та процедурою виходу із системи. Для управління сесіями та безпекою використовується механізм JWT (JSON Web Tokens). [25]

Метод register() приймає дані нового користувача (ім'я, email, пароль), валідує їх, створює новий запис у таблиці users з роллю 'user', хешує пароль. Після успішного створення облікового запису генерується JWT-токен, який повертається клієнту разом з інформацією про користувача.

```
public function register(StoreUserRequest $request): JsonResponse
{
    $validated = $request->validated();
    $user = User::create([
        'fullname' => $validated['fullname'],
        'email' => $validated['email'],
        'password' => $validated['password'],
        'role' => 'user',
    ]);
    $token = Auth::guard('api')->login($user);
    return $this->respondWithToken($token, $user);
}
```

Метод login() приймає облікові дані, перевіряє їх відповідність збереженим у базі даних. У разі успішної автентифікації генерується JWT-токен, який містить інформацію про користувача і повертається клієнту. У випадку невірних даних повертається відповідна помилка:

```
public function login(Request $request): JsonResponse
{
    $request->validate([
        'email' => 'required|email',
        'password' => 'required|string',
    ]);
    $credentials = $request->only('email', 'password');
    if (!$token = Auth::guard('api')->attempt($credentials)) {
        return response()->json(['error' => 'Unauthorized'], 401);
    }
    $user = Auth::guard('api')->user();
}
```

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return $this->respondWithToken($token, $user);
    }

```

Метод `getUser()` доступний для аутентифікованих користувачів, використовує JWT-токен, надісланий у заголовок запиту, для ідентифікації користувача та повертає інформацію про його профіль з бази даних:

```

public function getUser(): JsonResponse
{
    /** @var \App\Models\User $user */
    $user = Auth::guard('api')->user();

    $user->load([
        'favoriteMovies' => function ($query) {
            $query->with('genres')-
>latest('user_favorites.created_at');
        },
        'reviews' => function ($query) {
            $query->with('movie:id,title,poster_image_path')-
>latest();
        }
    ]);

    return response()->json($user);
}

```

Для адміністраторів системи реалізовано набір функцій для управління основним контентом сайту – каталогом фільмів. Ці функції доступні через спеціальні ендпоінти API, захищені перевіркою ролі адміністратора, і реалізовані, у контролері `MovieController`.

Метод `store()` - Метод приймає дані про новий фільм. Відбувається валідація даних, збереження інформації в таблицю `movies` та пов'язані таблиці (`movie_genre`, `movie_person_role`), а також завантаження постера за допомогою `FileManagementService`.

```

public function store(StoreMovieRequest $request): JsonResponse
{
    $data = $request->validated();
    $posterPath = null;

    if ($request->hasFile('poster')) {
        $posterPath = $this->fileService->uploadImage($request-
>file('poster'), 'posters');
    }

    $movie = DB::transaction(function () use ($data, $posterPath) {
        $movie = Movie::create([
            'title' => $data['title'],
            'description' => $data['description'],

```

```

        'age_rating' => $data['age_rating'],
        'release_year' => $data['release_year'],
        'duration_minutes' => $data['duration_minutes'],
        'trailer_url' => $data['trailer_url'] ?? null,
        'poster_image_path' => $posterPath,
    ]);

    $movie->genres()->sync($data['genres']);

    $peopleData = collect($data['people'])->mapWithKeys(function
($person) {
        return [
            $person['id'] => [
                'role_type' => $person['role_type'],
                'character_name' => $person['character_name'] ??
null
            ]
        ];
    });
    $movie->people()->sync($peopleData);

    return $movie;
});

return response()->json($movie->load('genres', 'people'), 201);
}

```

Метод update() - дозволяє оновити інформацію про існуючий фільм, включаючи його атрибути, пов'язані жанри, персон та постер.

```

public function update(UpdateMovieRequest $request, Movie $movie):
JsonResponse
{
    $data = $request->validated();

    if ($request->hasFile('poster')) {
        $this->fileService->deleteImage($movie->poster_image_path);
        $data['poster_image_path'] = $this->fileService-
>uploadImage($request->file('poster'), 'posters');
    }

    DB::transaction(function () use ($movie, $data) {
        $movie->update($data);

        if (isset($data['genres'])) {
            $movie->genres()->sync($data['genres']);
        }

        if (isset($data['people'])) {
            $peopleData = collect($data['people'])->
mapWithKeys(function ($person) {
                return [
                    $person['id'] => [
                        'role_type' => $person['role_type'],

```

```

        'character_name' => $person['character_name']
    ?? null
    ];
    });
    $movie->people()->sync($peopleData);
}
});

return response()->json($movie->load('genres', 'people'));
}

```

Метод `destroy()` - видаляє запис про фільм з бази даних, а також пов'язані з ним записи.

```

public function destroy(Movie $movie): JsonResponse
{
    DB::transaction(function () use ($movie) {
        $this->fileService->deleteImage($movie->poster_path);
        $movie->delete();
    });

    return response()->json(null, 204);
}

```

Для авторизованих користувачів реалізовано функціонал, що дозволяє взаємодіяти з фільмами, висловлювати свою думку та персоналізувати досвід користування сайтом.

Метод `store()` в `ReviewController` дає користувачу можливість поставити фільму оцінку та написати текстовий відгук. Система перевіряє, чи існує вже відгук від цього користувача для даного фільму. Якщо так – оновлює, якщо ні – створює новий запис у таблиці `reviews`.

```

public function store(Request $request): JsonResponse
{
    $data = $request->validate([
        'movie_id' => 'required|integer|exists:movies,id',
        'rating_score' => 'required|integer|min:1|max:10',
        'comment_text' => 'nullable|string',
    ]);

    $user = Auth::guard('api')->user();

    $review = Review::updateOrCreate(
        ['user_id' => $user->id, 'movie_id' => $data['movie_id']],
        ['rating_score' => $data['rating_score'], 'comment_text' =>
    $data['comment_text'] ?? null]
    );

    return response()->json($review, $review->wasRecentlyCreated ? 201
: 200);}

```

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

Метод addToFavorites() в UserFavoriteController дозволяє користувачу додати фільм до свого персонального списку улюблених. Створюється запис у таблиці user_favorites

```
public function addToFavorites(Movie $movie): JsonResponse
{
    $user = Auth::guard('api')->user();
    $user->favoriteMovies()->syncWithoutDetaching([$movie->id]);

    return response()->json(['message' => 'Movie added to favorites.']);
}
```

Метод removeFromFavorites() видаляє фільм зі списку улюблених користувача.

```
public function removeFromFavorites(Movie $movie): JsonResponse
{
    $user = Auth::guard('api')->user();
    $user->favoriteMovies()->detach($movie->id);

    return response()->json(['message' => 'Movie removed from favorites.']);
}
```

Ключовий функціонал сайту – це пошук та відображення інформації про фільми, у контролері MovieController.

Метод search() обробляє пошукові запити користувачів, дозволяючи шукати за різними параметрами. Реалізує фільтрацію та сортування результатів. Повертає список фільмів з пагінацією

```
public function search(Request $request): JsonResponse
{
    $query = Movie::query()->with('genres');

    if ($request->filled('title')) {
        $query->where('title', 'like', '%' . $request->input('title') . '%');
    }

    if ($request->filled('genres')) {
        $genreIds = explode(',', $request->input('genres'));
        $query->whereHas('genres', function ($q) use ($genreIds) {
            $q->whereIn('genres.id', $genreIds);
        });
    }

    if ($request->filled('year_from')) {
        $query->whereYear('release_year', '>=', $request->input('year_from'));
    }
}
```

```

        if ($request->filled('year_to')) {
            $query->whereYear('release_year', '<=', $request->
>input('year_to'));
        }

        if ($request->filled('person_id')) {
            $query->whereHas('people', function ($q) use ($request) {
                $q->where('people.id', $request->input('person_id'));
            });
        }

        switch ($request->input('sort_by')) {
            case 'release_year':
                $query->orderBy('release_year', $request->
>input('sort_dir', 'desc'));
                break;

            case 'rating':
                $query->withAvg('reviews as average_rating',
'rating_score')
                    ->orderByDesc('average_rating');
                break;

            default:
                $query->latest('created_at');
                break;
        }
    }

```

Метод `show()` повертає повну інформацію про конкретний фільм, включаючи його атрибути, список жанрів, акторський склад, режисерів, середній рейтинг та список відгуків.

```

public function show(Movie $movie): JsonResponse
{
    $movie->load(['genres', 'people', 'reviews']);

    $movie->average_rating = round($movie->reviews()-
>avg('rating_score'), 1);

    return response()->json($movie);
}

```

Висновки до другого розділу

У другому розділі спроектовано вебсайт «Кінопошук». Визначено ключові ролі — гість, авторизований користувач і адміністратор — та їхні сценарії взаємодії, що лягли в основу моделі контролю доступу. Створено схему БД MariaDB (користувачі, фільми, жанри, персони, відгуки, оцінки, обране) з

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		45

чіткими зв'язками й обґрунтовано вибір СУБД завдяки її продуктивності, надійності та масштабованості.

Окреслено концепти ключових модулів: FileManagementService для завантаження й зберігання медіафайлів; AuthController із JWT-автентифікацією для реєстрації, входу й керування сесіями; а також контролери адміністрування контенту (фільми, жанри, персони) і взаємодії користувачів (відгуки, оцінки, списки улюбленого).

					ІІЗк.КР.Б – 121 – 25 – ПЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК ВЕБСАЙТУ «КІНОПОШУК»

3.1. Порядок встановлення та налаштування системи

Розгортання й налаштування «Кінопошуку» повністю ґрунтується на контейнерній архітектурі. Спочатку клонують репозиторій із вихідним кодом, де містяться каталоги backend і frontend разом із файлом docker-compose.yaml, після чого створюють файл .env і прописують у ньому конфіденційні параметри середовища: облікові дані для бази даних, робочі порти та ключі доступу. Команда docker compose build, поєднана з docker compose up, збирає образи, запускає міграції та формує структуру сховища; перший запуск одночасно компілює React-клієнт і піднімає його в окремому контейнері. Конфігурація, описана у docker-compose.yaml, автоматизує старт усієї екосистеми й гарантує відтворюване розгортання: фронтенд на React збирається за власним Dockerfile у каталозі docker/frontend, бекенд на Laravel використовує контекст docker/api з під'єднаним вихідним кодом, nginx 1.17 читає налаштування з каталогу configs/nginx і слухає порт, указаний у змінній NGINX_PORT, MariaDB 11.3 відповідає за зберігання даних, а phpMyAdmin, підключений через внутрішню мережу за параметрами PMA_HOST та PMA_PORT, спрощує адміністрування. Усі контейнери працюють у спільній bridge-мережі, комунікують за внутрішніми DNS-іменами й не відкривають зайвих портів назовні, що одночасно підвищує безпеку та полегшує подальший супровід.

3.2. Структура інтерфейсу вебсайту «кінопошук» та порядок роботи

Інтерфейс користувача вебсайту "Кінопошук" відіграє провідну роль у досягненні ключової мети проєкту – забезпечення зручної, інтуїтивно зрозумілої та ефективної взаємодії між кінцевими користувачами (глядачами, кіноманами) та адміністраторами системи. Саме візуальна частина вебсайту формує перше враження про ресурс, визначає легкість орієнтування в його структурі, а також

					ІПЗк.КР.Б – 121 – 25 – ІЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

впливає на швидкість, з якою користувачі здатні знаходити потрібну інформацію про фільми, читати відгуки, залишати власні оцінки чи керувати своїм профілем.

Особливу увагу у процесі проєктування інтерфейсу було приділено логіці навігації, швидкому доступу до ключових функціональних елементів та відповідності сучасним принципам UX/UI-дизайну (користувацького досвіду та інтерфейсу користувача)[29]. У цьому підпункті детально проаналізовано основні компоненти інтерфейсу вебсайту "Кінопошук", зокрема: головну сторінку, сторінки пошуку та фільтрації фільмів, сторінку детальної інформації про фільм, сторінки реєстрації та авторизації, профіль користувача, а також адміністративну панель. Детальний опис прийнятих інтерфейсних рішень дозволить чітко з'ясувати, яким чином побудовано взаємодію користувача із системою.

Головна сторінка (рис. 3.1) є візитівкою вебсайту "Кінопошук". Вона надає користувачеві швидкий доступ до основного меню з ключовими розділами сайту. Центральна частина сторінки містить списки нових фільмів. Також на головній сторінці розташований помітний пошуковий рядок для швидкого пошуку фільмів за назвою. У верхній частині сторінки знаходяться посилання для авторизації/реєстрації або переходу до профілю користувача, якщо він вже авторизований.

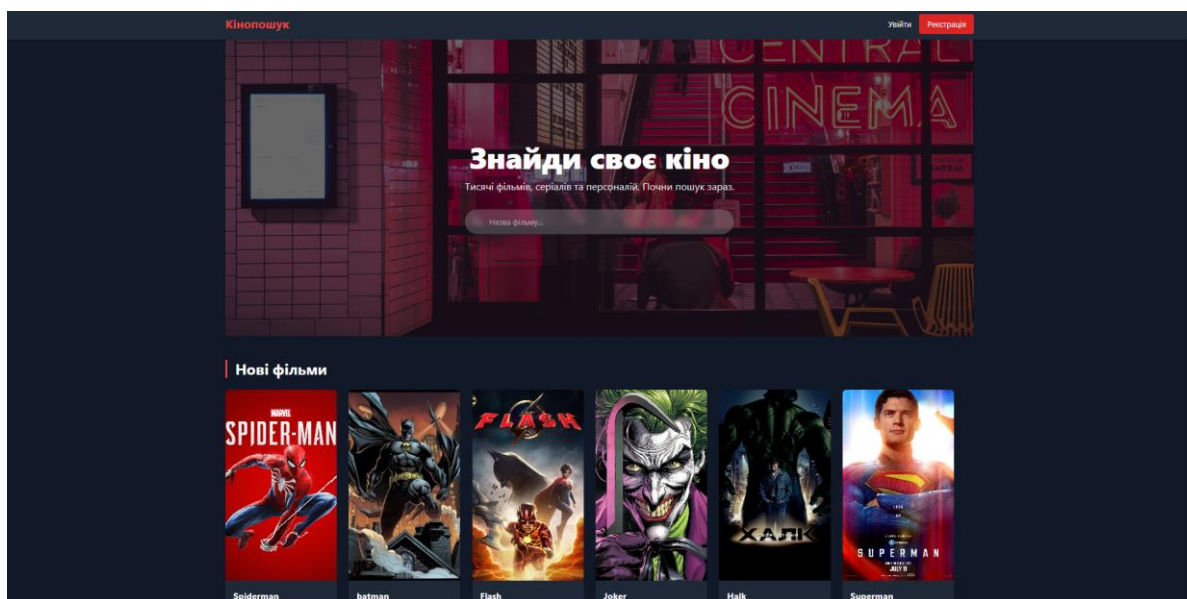


Рисунок 3.1 – Головна сторінка

Змн.	Арк.	№ докум.	Підпис	Дата

З головної сторінки може перейти до сторінки розширеного пошуку фільмів (рис. 3.2). Ця сторінка надає потужні інструменти для знаходження фільмів за різноманітними критеріями розташована форма з полями для введення параметрів пошуку: назва фільму, жанр, рік випуску. Також тут доступні фільтри за рейтингом, новизною та датою випуску. Основна частина сторінки відображає результати пошуку у вигляді карток, де для кожного фільму показана мінімальна інформація: постер, назва, рік випуску, жанри.

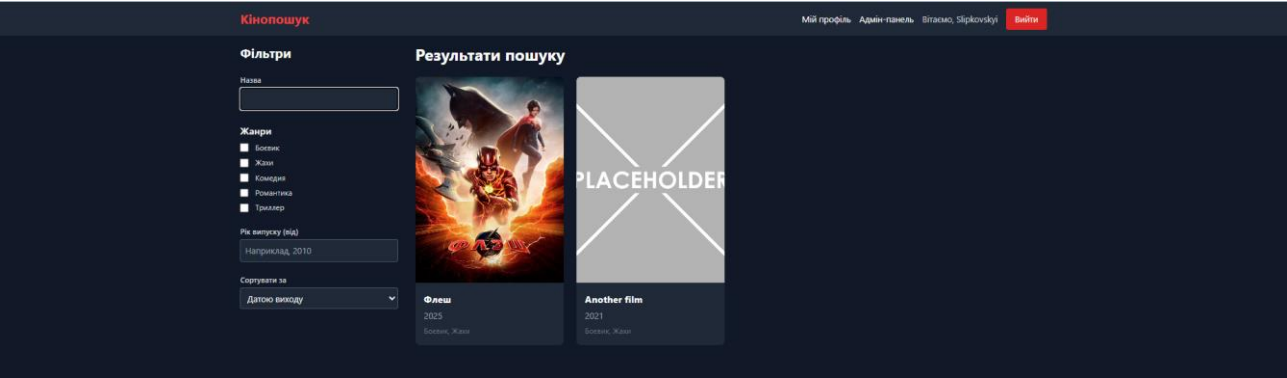


Рисунок 3.2 – Сторінка розширеного пошуку фільмів

Кожен користувач має можливість налаштувати фільтри пошуку таким чином, щоб знайти фільми, які максимально відповідають його індивідуальним уподобанням та критеріям. Приклад сторінки з результатами пошуку після застосування декількох фільтрів зображено на рисунку 3.3.

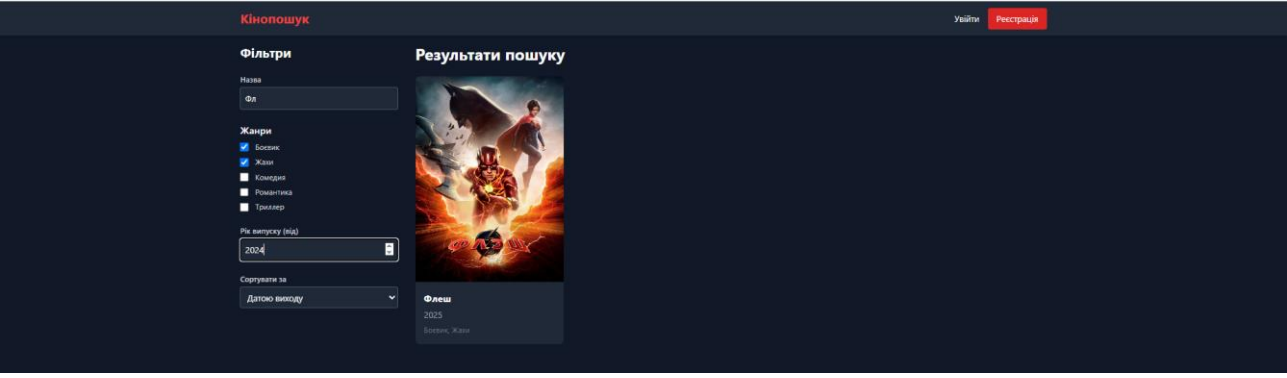


Рисунок 3.3 – Сторінка пошуку фільмів із застосованими фільтрами

Після того, як користувач знайде цікавий для нього фільм у списку результатів пошуку або на головній сторінці, він може перейти до сторінки детальної інформації про фільм (рис. 3.4), натиснувши на картку фільму. Ця сторінка надає вичерпну інформацію про обрану кінострічку. Тут представлено:

великий постер, повна назва, рік випуску, перелік жанрів, детальний опис сюжету, посилання на трейлер (якщо є), інформація про тривалість та віковий рейтинг. Також на цій сторінці відображається середній рейтинг фільму, список акторів та режисерів, а також секція з відгуками та оцінками інших користувачів. Авторизовані користувачі тут же мають можливість оцінити фільм та додати свій відгук.

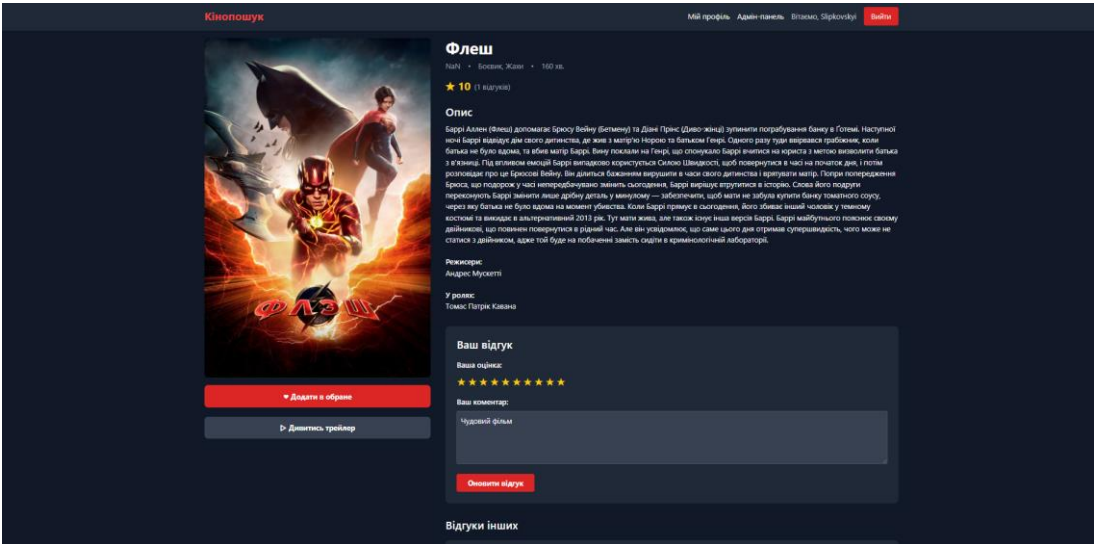


Рисунок 3.4 – Сторінка детальної інформації про фільм

Для того, щоб мати можливість оцінювати фільми, залишати відгуки, формувати список улюблених та користуватися іншими персоналізованими функціями, користувачеві необхідно авторизуватися в системі. На сторінці авторизації (рис. 3.5) користувачеві пропонується ввести свою електронну пошту та пароль, вказані під час реєстрації.

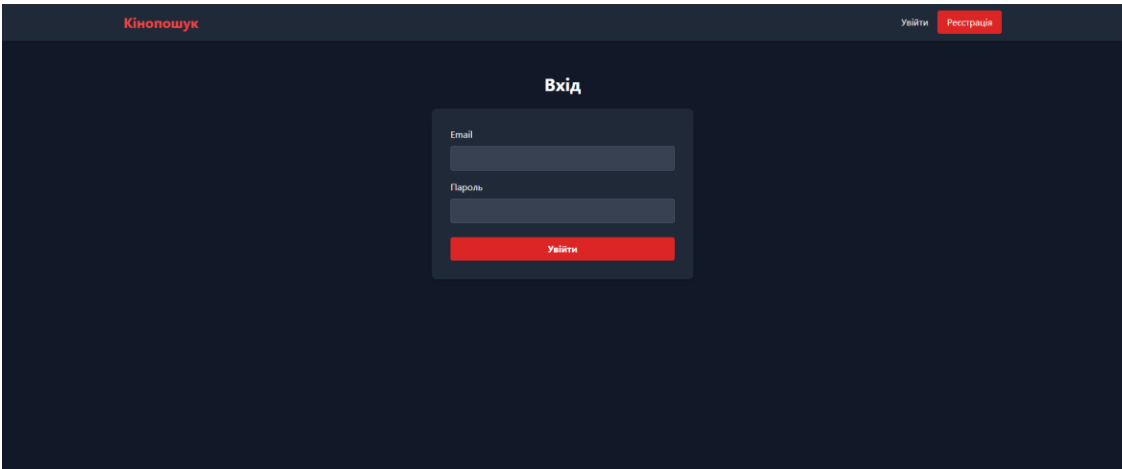


Рисунок 3.5 – Сторінка авторизації в систему

Якщо користувач ще не має облікового запису на вебсайті "Кінопошук", він може легко його створити, перейшовши на сторінку реєстрації (рис. 3.6). Форма реєстрації зазвичай вимагає введення основних даних: ім'я користувача (логін), електронна пошта та пароль (з підтвердженням пароля).

Рисунок 3.6 – Сторінка реєстрації нового користувача

Після успішної авторизації користувач отримує доступ до свого профілю користувача (рис. 3.7). На цій сторінці відображається інформація про його активність на сайті: перелік залишених відгуків, а також список улюблених фільмів.

Рисунок 3.7 – Профіль користувача зі списком активності

					ІІЗк.КР.Б – 121 – 25 – ІЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

Адміністративна панель (рис. 3.8, 3.9, 3.10) призначена для користувачів з роллю «Адміністратор» і надає їм розширені можливості для управління контентом та функціоналом вебсайту "Кінопошук". Доступ до адміністративної панелі зазвичай здійснюється через окрему URL-адресу та вимагає додаткової авторизації.

Наприклад, адміністратор має доступ до розділу управління фільмами (рис. 3.8). Тут він може: додавати нові фільми, редагувати інформацію про існуючі фільми та видаляти фільми з бази даних.

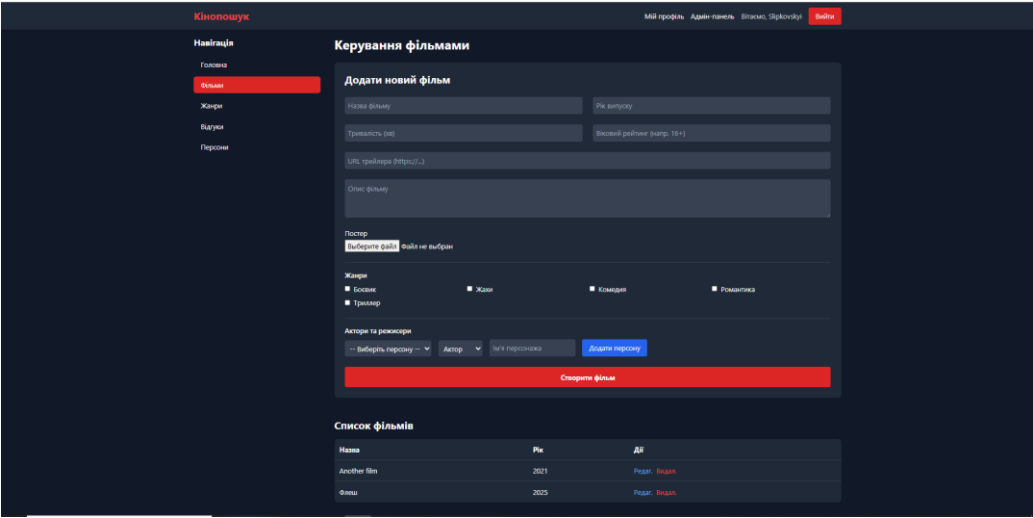


Рисунок 3.8 – Адміністративна панель: розділ управління фільмами

Адміністратор може керувати довідником жанрів (рис. 3.9) та персонами (рис. 3.10).

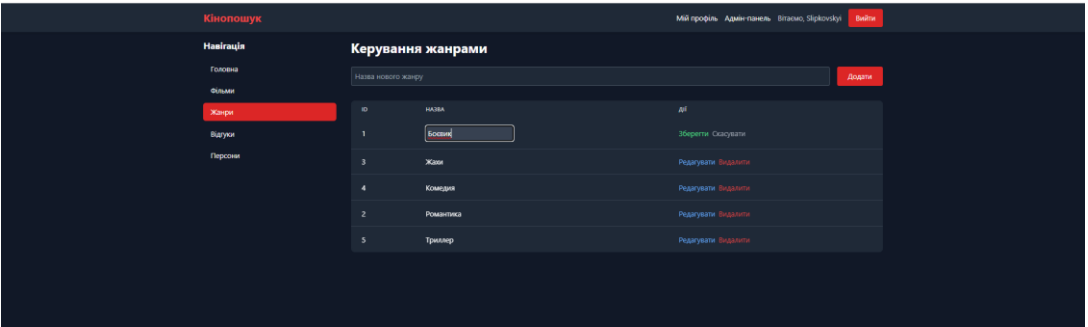


Рисунок 3.9 – Адміністративна панель: розділ управління жанрами

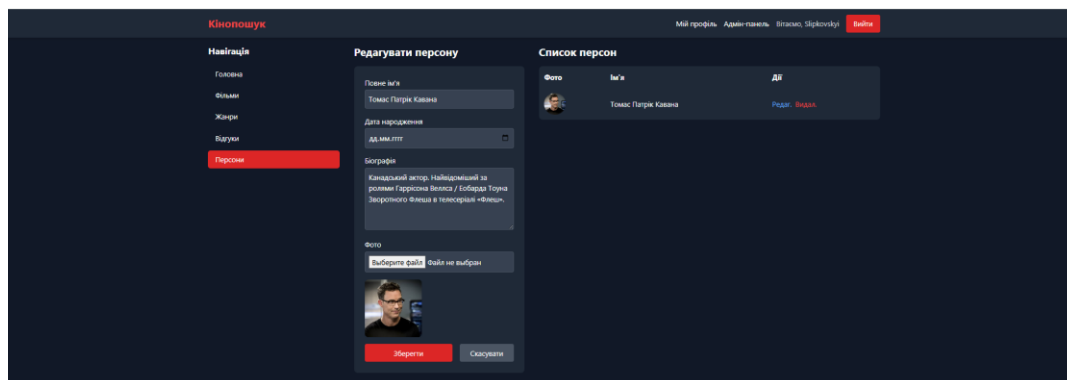


Рисунок 3.10 – Адміністративна панель: розділ управління персонами

Однією з важливих функцій адміністратора є модерація контенту, створеного користувачами, зокрема відгуків (рис. 3.11). Адміністратор може видаляти ті, що порушують правила спільноти (наприклад, містять спам, образи або нецензурну лексику).

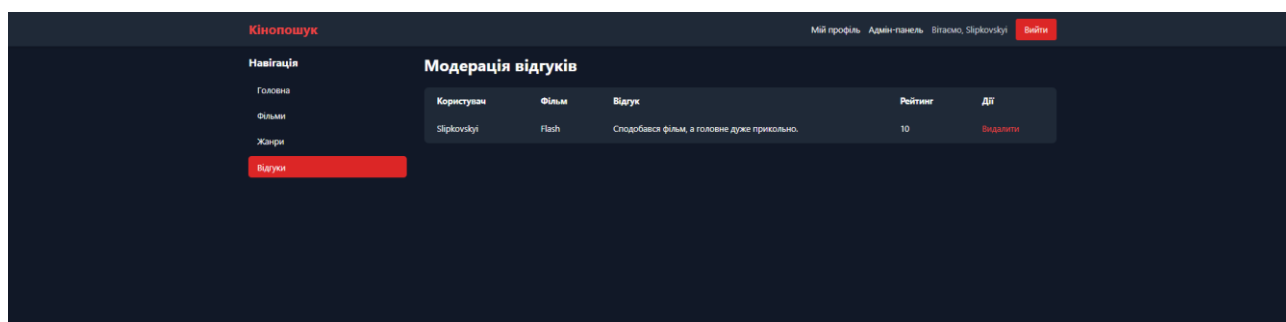


Рисунок 3.11 – Адміністративна панель: розділ модерації відгуків

Такий структурований підхід до проектування інтерфейсу користувача вебсайту "Кінопошук" спрямований на створення інтуїтивно зрозумілого, функціонального та візуально привабливого середовища для всіх категорій користувачів, що сприятиме їх ефективній взаємодії з платформою та позитивному досвіду користування.

3.3. Тестування роботи вебдодатку

Процес тестування відіграє фундаментальну роль у життєвому циклі розробки будь-якого програмного продукту, і вебсайт "Кінопошук" не є винятком. Саме на цьому етапі здійснюється ретельна перевірка відповідності реалізованої системи раніше визначеним функціональним та нефункціональним вимогам. У рамках розробки вебсайту "Кінопошук", що надає користувачам

інформацію про фільми, можливість їх оцінювання та обговорення, тестування дозволяє виявити потенційні технічні помилки у роботі ключових модулів, проаналізувати надійність та коректність взаємодії між окремими компонентами системи (фронтом, бекендом, базою даних), а також об'єктивно оцінити логічність, інтуїтивність та загальну зручність розробленого інтерфейсу з позиції кінцевого користувача.

Для всебічної перевірки працездатності, стабільності та коректності функціонування розробленого вебсайту "Кінопошук" було проведено мануальне (ручне) тестування основних користувацьких сценаріїв. Особливий акцент під час тестування було зроблено на перевірці обробки системою різноманітних виняткових ситуацій та помилок [30]. Зокрема, тестування охоплювало перевірку коректності відображення інформативних повідомлень про помилки у випадках порушення умов доступу до певних ресурсів (наприклад, спроба неавторизованого користувача залишити відгук), неправильного введення даних у форми (реєстрації, авторизації, додавання відгуку), або звернення до неіснуючих сторінок чи ресурсів сайту. Під час цього етапу було навмисно змодельовано типові помилки, які можуть траплятися під час активного користування вебсайтом.

1. Спроби доступу до неіснуючого маршруту або ресурсу. Наприклад, при переході за URL-адресою, що містить помилку, або при спробі звернутися до сторінки фільму, який був видалений з бази даних. У процесі тестування здійснювалися навмисні спроби звернення до таких неіснуючих маршрутів. У результаті система коректно ідентифікувала такі запити та відображала спеціально розроблену сторінку з кодом помилки 404 (Not Found) (рис. 3.12). Дизайн сторінки помилки 404 передбачає чітке та зрозуміле повідомлення для користувача про те, що запитуваний ресурс не знайдено, а також надає зручні посилання для швидкого повернення на головну сторінку сайту або до інших ключових розділів.

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

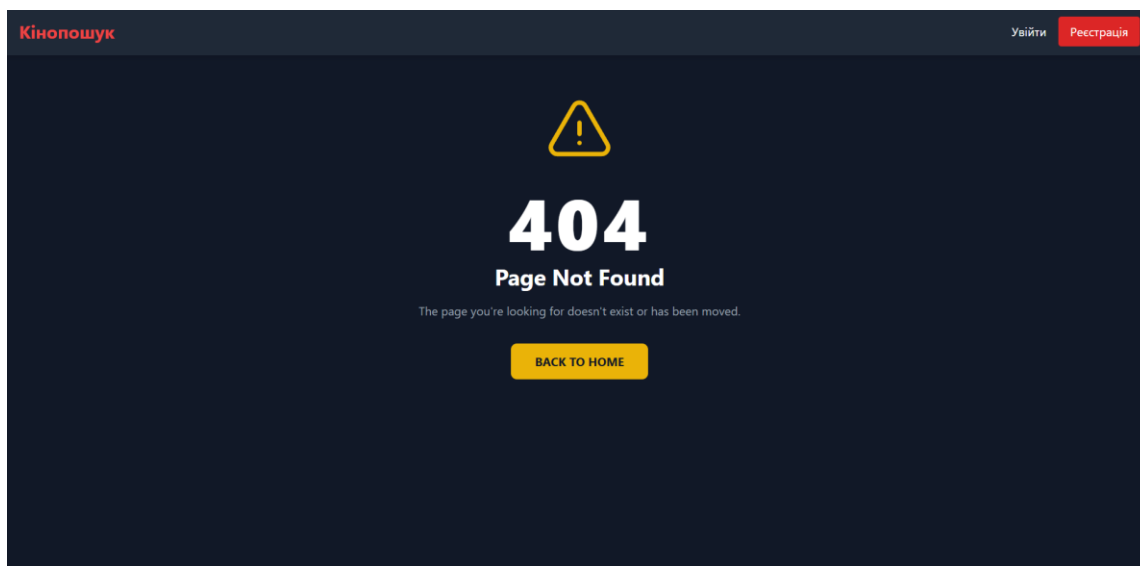


Рисунок 3.12 – Сторінка помилки 404

2. Спроби несанкціонованого доступу до ресурсів або виконання дій. Наприклад, ситуація, коли неавторизований користувач намагається залишити оцінку фільму, або звичайний авторизований користувач намагається отримати доступ до адміністративної панелі. Під час тестування було перевірено, що система безпеки належним чином перехоплює такі неавторизовані запити. У відповідь користувачеві відображається сторінка з кодом помилки 403 (Forbidden) (рис. 3.13), яка інформує про відмову в доступі до запитуваного ресурсу або дії. Сторінка також може містити рекомендацію повернутися на головну сторінку або авторизуватися (якщо користувач не увійшов до системи).

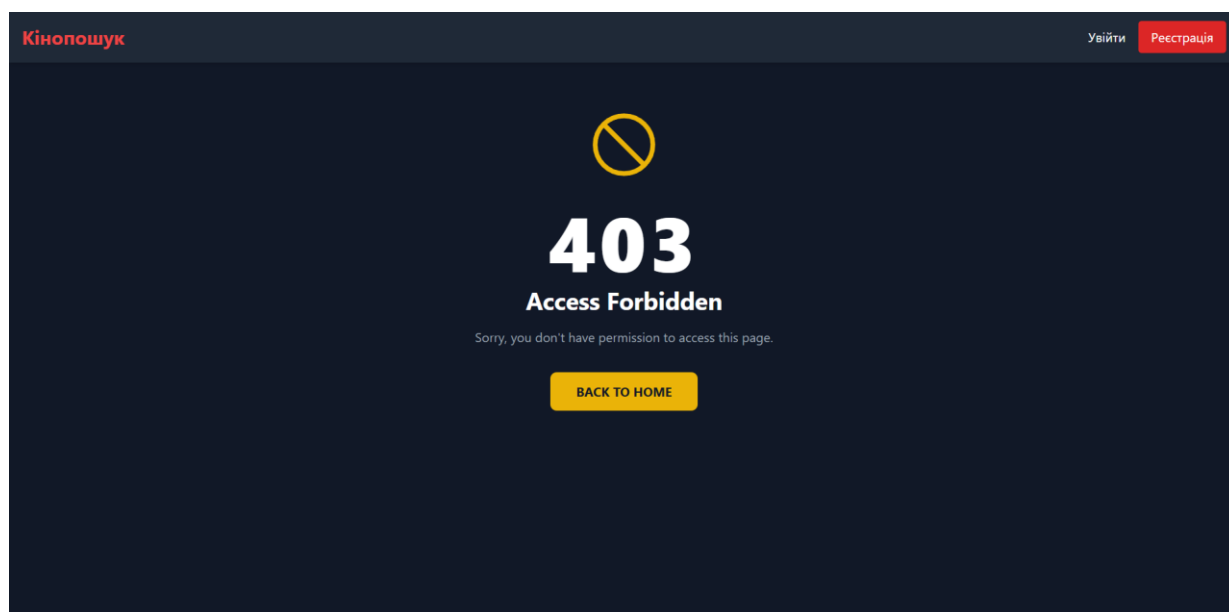


Рисунок 3.13 – Сторінка помилки 403

					ІІЗк.КР.Б – 121 – 25 – ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

3. Критичні помилки на рівні серверної логіки (внутрішні помилки сервера). Було змодельовано ситуації, які потенційно могли б викликати непередбачену внутрішню помилку на стороні сервера (наприклад, помилка підключення до бази даних, непередбачений виняток у коді бекенду). У результаті таких змодельованих збоїв користувачеві коректно відображалась спеціальна сторінка з повідомленням про внутрішню помилку сервера (код 500 - Internal Server Error) (рис. 3.14). Такий підхід дозволяє уникнути відображення "сирих" технічних повідомлень про помилки або трасувань стеку, які можуть бути незрозумілими для користувача та потенційно містити чутливу інформацію. Натомість, забезпечується базовий рівень інформативності та зручності навіть у критичних ситуаціях.

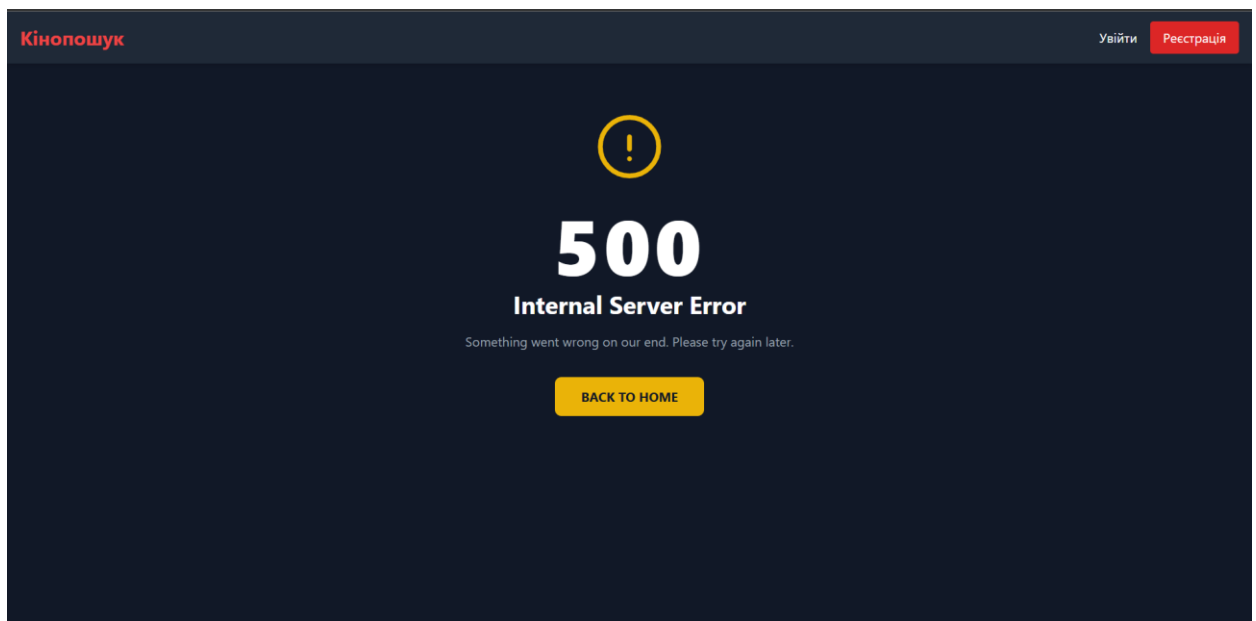


Рисунок 3.14 – Сторінка помилки 500

У межах тестування вебсайту "Кінопошук" було також проведено ретельну перевірку надійності та коректності реалізації механізмів валідації вхідних даних, особливо під час процесу реєстрації нового користувача та додавання відгуків/оцінок. Основна увага була приділена перевірці того, як система реагує на неправильне, неповне або некоректне заповнення обов'язкових полів у формах, а також наскільки чіткими та інформативними є повідомлення про помилки, що відображаються користувачеві (рис. 3.15 – приклад валідації на формі реєстрації).

					ІІЗк.КР.Б – 121 – 25 – ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

Рисунок 3.15 – Перевірка валідації вхідних даних

Зокрема, було змодельовано кілька ситуацій із навмисним порушенням вимог до форматів:

1. Неправильний формат електронної пошти: Під час введення адреси електронної пошти без символу “@”, з некоректним доменом (наприклад, без доменної зони типу .com чи .ua) або з іншими порушеннями стандартного формату, система автоматично блокувала відправку форми та виводила чітке, локалізоване повідомлення про помилку поруч із відповідним полем, інформуючи про необхідність ввести коректну email-адресу.

2. Недостатня довжина або невідповідність вимогам до пароля: Було перевірено введення пароля, що не відповідав встановленим критеріям безпеки (наприклад, занадто короткий пароль, відсутність цифр або спеціальних символів, якщо такі вимоги були задані). Результатом стало відображення інформативного повідомлення про необхідність створити більш складний та надійний пароль, що відповідає вимогам системи.

3. Порожні обов'язкові поля: Спроба відправити форму (наприклад, реєстрації або додавання відгуку) без заповнення одного чи декількох обов'язкових полів призводила до відображення відповідних повідомлень про помилку біля кожного незаповненого поля.

Загалом, результати проведеного мануального тестування переконливо підтвердили, що механізми валідації вхідних даних на вебсайті «Кінопошук» розроблені відповідно до найкращих сучасних практик веброзробки. Валідація відбувається як на стороні клієнта для оперативного зворотного зв'язку і мінімізації навантаження на сервер, так і на стороні сервера — для остаточного контролю цілісності даних, запобігання ін'єкціям та інших потенційних уразливостей. Під час тестування система безвідмовно перехоплювала некоректні, неповні або зловмисно сформовані запити ще до проходження критичних логічних гілок бекенда, забезпечуючи цілісність БД та стабільність роботи всього застосунку. Окрім цього, повідомлення про помилки формувалися чітко, інформативно.

Висновки до третього розділу

У третьому розділі кваліфікаційної роботи детально описано практичну реалізацію вебсайту «Кінопошук». Спочатку пояснено порядок розгортання проєкту за допомогою Docker — контейнеризація дала змогу ізолювати компоненти, стандартизувати процес інсталяції та гарантувати відтворюваність запуску на будь-якій операційній системі. Використання Docker Compose забезпечило централізовану конфігурацію і автоматичний запуск усіх служб: бекенду на Laravel, фронтенду на React, MariaDB, phpMyAdmin та Nginx. Далі проаналізовано логіку користувацького інтерфейсу: навігація продумана так, щоб шлях від головної сторінки до детальної інформації про фільм і подальших дій — авторизації, оцінювання чи формування списків — залишався послідовним та інтуїтивним. Рольова модель доступу розділяє можливості гостей, зареєстрованих користувачів і адміністраторів, що спрощує керування контентом і підтримує безпеку. Завершальним етапом став цикл ручних тестів, під час якого імітувалися типові та виняткові ситуації, включно зі спробами доступу до заборонених маршрутів, переходами на неіснуючі сторінки й виникненням внутрішніх помилок сервера. Перевірка підтвердила стабільність

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

роботи модулів, коректність обробки помилок із кодами 403, 404 та 500 і належний рівень користувацького досвіду завдяки зрозумілим повідомленням. Отримані результати демонструють, що реалізований вебсайт відповідає сучасним вимогам до кіно-орієнтованих інформаційно-пошукових сервісів і готовий до подальшого розвитку.

					ІІЗк.КР.Б – 121 – 25 – ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У результаті виконання даної кваліфікаційної роботи було комплексно вирішено завдання розробки вебсайту "Кінопошук", що надає користувачам зручний інструмент для пошуку інформації про фільми, їх оцінювання, обговорення та персоналізації кінематографічного досвіду, і відповідає сучасним потребам користувачів інформаційних онлайн-ресурсів.

Проведений на початковому етапі аналіз тенденцій у сфері онлайн-кінопорталів та сервісів, присвячених кінематографу, дозволив чітко сформулювати актуальні вимоги до подібних систем. Було визначено ключові фактори, що забезпечують зручність взаємодії користувачів з платформою та сприяють формуванню активної спільноти кіноманів. Особливу увагу було приділено врахуванню потенційних недоліків існуючих популярних платформ, таких як обмежені можливості глибокої фільтрації, не завжди інтуїтивний інтерфейс або недостатня увага до певних аспектів кіноінформації.

Здійснений аналіз вебсайтів-аналогів підтвердив необхідність створення ресурсу, який би поєднував інформаційну повноту та достовірність з гнучким та потужним функціоналом пошуку, а також надавав інтерактивні можливості для користувачів. Використання обґрунтованого архітектурного підходу, заснованого на клієнт-серверній моделі з чітким розподілом відповідальності між фронтенд- та бекенд-частинами, що взаємодіють через стандартизований RESTful API, дозволило побудувати гнучку, масштабовану та легко підтримувану систему, готову до подальшого розвитку та потенційних інтеграцій.

Етап проєктування забезпечив створення чіткої логічної структури вебсайту "Кінопошук". Це включало розробку деталізованої моделі бази даних із ретельною специфікацією всіх необхідних таблиць, їх полів та взаємозв'язків між ними, а також ідентифікацію та опис основних сценаріїв використання системи для різних категорій користувачів (Неавторизований гість, Авторизований користувач, Адміністратор). Такий системний підхід заклав

					ІІЗк.КР.Б – 121 – 25 – ІЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

міцний фундамент для подальшої реалізації ефективних механізмів контролю доступу, забезпечення стабільності роботи вебсайту та безпеки зберігання користувацьких та контентних даних.

Розроблене програмне рішення побудоване на базі сучасних та перевірених технологій, таких як Docker для контейнеризації, PHP-фреймворк Laravel для розробки бекенд-логіки, JavaScript-бібліотека React для створення динамічного фронтенд-інтерфейсу та реляційна СУБД MariaDB для зберігання даних. Вибір даного технологічного стеку забезпечив не лише повну функціональну завершеність проєкту, але й його надійність, продуктивність, зручність технічного супроводу та адаптивність до можливих змін у робочому середовищі. Зокрема, запровадження контейнеризації за допомогою Docker та Docker Compose дозволило значно спростити процедуру розгортання вебсайту та забезпечити прогнозовану та стабільну роботу системи в будь-яких умовах експлуатації.

Детально опрацьований та реалізований інтерфейс користувача вебсайту "Кінопошук" відповідає сучасним стандартам та принципам UX/UI-дизайну. Він забезпечує логічний, послідовний та інтуїтивно зрозумілий шлях користувача по сайту – від пошуку фільмів та ознайомлення з детальною інформацією про них до можливості оцінювання, залишення відгуків, формування персональних списків улюблених фільмів та управління власним профілем. Адміністративна панель надає зручні інструменти для ефективного управління контентом сайту.

Проведене на завершальному етапі ручне (мануальне) тестування ключових функціональних можливостей вебсайту підтвердило стабільність роботи розробленої платформи у типових користувацьких сценаріях, включаючи коректне реагування системи на різноманітні помилки та виняткові ситуації (наприклад, спроби доступу до неіснуючих ресурсів або несанкціоновані дії). Тестування механізмів валідації вхідних даних у формах (реєстрації, авторизації, додавання відгуків) продемонструвало високий рівень захисту системи від

					ІІЗк.КР.Б – 121 – 25 – ІЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

некоректних або неповних даних, що додатково підвищує її надійність та зручність у практичному застосуванні.

Таким чином, отримані в ході виконання кваліфікаційної роботи результати свідчать про успішну реалізацію поставленої мети – розробки функціонального та сучасного вебсайту "Кінопошук". Створений програмний продукт відповідає усім сформульованим на початку роботи вимогам та готовий до подальшого розвитку та потенційного використання.

					ІІЗк.КР.Б – 121 – 25 – ПЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. An Expert Guide to PHP Framework Popularity in 2024 - 2025 [Електронний ресурс] – Режим доступу до ресурсу: <https://devabit.com/blog/php-framework-popularity-2024/>
2. Docker Compose: опис та використання [Електронний ресурс] – Режим доступу до ресурсу: <https://freehost.com.ua/ukr/faq/articles/docker-compose/>
3. Docker Deep Dive / Nigel Poulton, 2017. – 249 p.
4. IMDb [Електронний ресурс] – Режим доступу до ресурсу: <https://www.imdb.com/>
5. Kane S. P., Matthias K. Docker: Up & Running: Shipping Reliable Containers in Production. 3rd Edition / O'Reilly Media, 2023. – 416 p.
6. Laravel Tutorials [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel-news.com/category/tutorials>
7. Learning MySQL and MariaDB: Heading in the Right Direction with MySQL and MariaDB / O'Reilly Media, 2015. – 408 p.
8. Letterboxd [Електронний ресурс] – Режим доступу до ресурсу: <https://letterboxd.com/>
9. MariaDB. Документація [Електронний ресурс] – Режим доступу до ресурсу: <https://mariadb.com/kb/en/library/>
10. Microservices vs. monolithic architecture [Електронний ресурс] – Режим доступу до ресурсу: <https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith>
11. REST API Design Rulebook / O'Reilly Media, 2011. – 112 p.
12. Rotten Tomatoes [Електронний ресурс] – Режим доступу до ресурсу: <https://www.rottentomatoes.com/>
13. SQL QuickStart Guide: The Simplified Beginner's Guide to Managing, Analyzing, and Manipulating Data With SQL (Coding & Programming – QuickStart Guides) / ClydeBank Media LLC, 2019 – 242 p.

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

- 14.Ullman L. PHP Advanced and Object-Oriented Programming: Visual QuickPro Guide, 3rd ed. / Peachpit Press, 2012. – 504 p.
- 15.Use Case Diagram – Unified Modeling Language (UML) [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/uml/uml_use_case_diagram.htm
- 16.What is UML? [Електронний ресурс] – Режим доступу до ресурсу: <https://www.edrawmax.com/uml/>
- 17.What is a REST API? [Електронний ресурс] – Режим доступу до ресурсу: <https://www.redhat.com/en/topics/api/what-is-a-rest-api>
- 18.What is a RESTful API? [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mulesoft.com/api/rest/what-is-rest-api>
- 19.Zandstra M. PHP 8 Objects, Patterns, and Practice: Mastering OO Enhancements, Design Patterns, and Essential Development Tools / Apress, 2021. – 833 p.
- 20.Де зберігати файли: база даних чи файлове сховище? [Електронний ресурс] – Режим доступу до ресурсу: <https://master-spring-ter.medium.com/to-blob-or-not-to-blob-storing-files-in-your-database-vs-ff7b8d0cbee1>
- 21.Документація React [Електронний ресурс] – Режим доступу до ресурсу: <https://react.dev/learn>
- 22.Нефункціональні вимоги: приклади, типи, підходи [Електронний ресурс] – Режим доступу до ресурсу: <https://training.qatestlab.com/blog/technical-articles/non-functional-requirements-examples-types-approaches/>
- 23.Основні типи архітектури програмного забезпечення [Електронний ресурс] – Режим доступу до ресурсу: <https://www.artofba.com/uk/post/main-types-of-software-architecture>
- 24.Офіційна документація Docker [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.docker.com/>

25. Плюси та мінуси JWT: короткий огляд тонкощів цієї технології [Електронний ресурс] – Режим доступу до ресурсу: <https://highload.tech/uk/plyusy-i-minusy-jwt-kratkij-obzor/>
26. Файлові системи Linux: типи та особливості [Електронний ресурс] – Режим доступу до ресурсу: <https://www.ufsexplorer.com/uk/articles/linux-file-systems/>
27. Що таке база даних? Визначення, типи, особливості [Електронний ресурс] – Режим доступу до ресурсу: <https://university.sigma.software/what-is-database/>
28. Що таке Docker і навіщо він? [Електронний ресурс] – Режим доступу до ресурсу: <https://qagroup.com.ua/publications/shcho-take-docker-i-navishcho-vin>
29. Як створити інтуїтивно зрозумілий додаток для охорони здоров'я: головні принципи UX/UI [Електронний ресурс] – Режим доступу до ресурсу: <https://wezom.com.ua/ua/blog/yak-stvoriti-intuyitivno-zrozumiliy-dodatok-dlya-ohoroni-zdorovya-golovni-printsipi-uxui>
30. Як тестувати веб-сайт: основні етапи і поради [Електронний ресурс] – Режим доступу до ресурсу: <https://brainlab.com.ua/uk/blog-uk/yak-testuvati-veb-sayt-osnovn-etapi-poradi/>

ДОДАТКИ

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

Файл docker-compose.yaml

```

version: "3.9"

services:
  api:
    build:
      context: .
      dockerfile: api/Dockerfile
    volumes:
      - ./api:/var/www
      - /var/www/vendor
    environment:
      APP_ENV: local
      DB_HOST: mysql
      DB_DATABASE: ${MYSQL_DATABASE}
      DB_USERNAME: ${MYSQL_USER}
      MARIADB_ALLOW_EMPTY_ROOT_PASSWORD: 1
    networks: [internal]

  frontend:
    build:
      context: .
      dockerfile: frontend/Dockerfile
    command: sh -c "npm run dev -- --host 0.0.0.0"
    volumes:
      - ./frontend:/app
      - /app/node_modules
    ports:
      - "5173:5173"
    networks: [internal]

  nginx:
    image: nginx:1.27-alpine
    volumes:
      - ./docker/nginx/default.conf:/etc/nginx/conf.d/default.conf:ro
      - ./frontend/dist:/usr/share/nginx/html:ro
      - ./api/public:/var/www/public:ro
    ports:
      - "${NGINX_PORT-80}:80"
    depends_on: [api, frontend]
    networks: [internal]

  mysql:
    image: mariadb:11.3
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE:      ${MYSQL_DATABASE}
      MYSQL_USER:           ${MYSQL_USER}
      MYSQL_PASSWORD:       ${MYSQL_PASSWORD}
      TZ: Europe/Kyiv
    volumes:
      - mysql_data:/var/lib/mysql

```

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

```
ports:
  - "3306:3306"
networks: [internal]
```

```
phpmyadmin:
  image: phpmyadmin:latest
  environment:
    PMA_HOST: mysql
    PMA_PORT: 3306
  ports:
    - "8080:80"
  depends_on: [mysql]
  networks: [internal]
```

```
networks:
  internal:
    driver: bridge
```

```
volumes:
  mysql_data:
```

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

AuthController.php

```

<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Http\Requests\Api\StoreUserRequest;
use App\Models\User;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class AuthController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth:api', ['except' => ['login',
'register']] );
    }

    public function register(StoreUserRequest $request): JsonResponse
    {
        $validated = $request->validated();

        $user = User::create([
            'fullname' => $validated['fullname'],
            'email' => $validated['email'],
            'password' => $validated['password'],
            'role' => 'user',
        ]);

        $token = Auth::guard('api')->login($user);
        return $this->respondWithToken($token, $user);
    }

    public function login(Request $request): JsonResponse
    {
        $request->validate([
            'email' => 'required|email',
            'password' => 'required|string',
        ]);

        $credentials = $request->only('email', 'password');

        if (!$token = Auth::guard('api')->attempt($credentials)) {
            return response()->json(['error' => 'Unauthorized'], 401);
        }

        $user = Auth::guard('api')->user();
        return $this->respondWithToken($token, $user);
    }
}

```

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		69

```

public function getUser(): JsonResponse
{
    /** @var \App\Models\User $user */
    $user = Auth::guard('api')->user();

    $user->load([
        'favoriteMovies' => function ($query) {
            $query->with('genres')-
>latest('user_favorites.created_at');
        },
        'reviews' => function ($query) {
            $query->with('movie:id,title,poster_image_path')-
>latest();
        }
    ]);

    return response()->json($user);
}

public function logout(): JsonResponse
{
    Auth::guard('api')->logout();
    return response()->json(['message' => 'Successfully logged out']);
}

protected function respondWithToken(string $token, User $user):
JsonResponse
{
    return response()->json([
        'access_token' => $token,
        'token_type' => 'bearer',
        'expires_in' => Auth::guard('api')->factory()->getTTL() * 60,
        'user' => $user
    ]);
}
}

```

MovieController.php

```

<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Http\Requests\Api\StoreMovieRequest;
use App\Http\Requests\Api\UpdateMovieRequest;
use App\Models\Movie;
use App\Services\FileManagementService;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class MovieController extends Controller
{
    protected FileManagementService $fileService;

    public function __construct(FileManagementService $fileService)
    {
        $this->fileService = $fileService;
    }

    public function store(StoreMovieRequest $request): JsonResponse
    {
        $data = $request->validated();
        $posterPath = null;

        if ($request->hasFile('poster')) {
            $posterPath = $this->fileService->uploadImage($request-
>file('poster'), 'posters');
        }

        $movie = DB::transaction(function () use ($data, $posterPath) {
            $movie = Movie::create([
                'title' => $data['title'],
                'description' => $data['description'],
                'age_rating' => $data['age_rating'],
                'release_year' => $data['release_year'],
                'duration_minutes' => $data['duration_minutes'],
                'trailer_url' => $data['trailer_url'] ?? null,
                'poster_image_path' => $posterPath,
            ]);

            $movie->genres()->sync($data['genres']);

            $peopleData = collect($data['people'])->mapWithKeys(function
($person) {
                return [
                    $person['id'] => [
                        'role_type' => $person['role_type'],
                        'character_name' => $person['character_name'] ??
null

```

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        ]
    ];
    });
    $movie->people()->sync($peopleData);

    return $movie;
});

return response()->json($movie->load('genres', 'people'), 201);
}

public function update(UpdateMovieRequest $request, Movie $movie):
JsonResponse
{
    $data = $request->validated();

    if ($request->hasFile('poster')) {
        $this->fileService->deleteImage($movie->poster_image_path);
        $data['poster_image_path'] = $this->fileService-
>uploadImage($request->file('poster'), 'posters');
    }

    DB::transaction(function () use ($movie, $data) {
        $movie->update($data);

        if (isset($data['genres'])) {
            $movie->genres()->sync($data['genres']);
        }

        if (isset($data['people'])) {
            $peopleData = collect($data['people']->mapWithKeys(function
($person) {
                return [
                    $person['id'] => [
                        'role_type' => $person['role_type'],
                        'character_name' => $person['character_name']
                    ]
                ];
            }));
            $movie->people()->sync($peopleData);
        }
    });

    return response()->json($movie->load('genres', 'people'));
}

public function destroy(Movie $movie): JsonResponse
{
    DB::transaction(function () use ($movie) {
        $this->fileService->deleteImage($movie->poster_image_path);
        $movie->delete();
    });

    return response()->json(null, 204);
}

```

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    public function search(Request $request): JsonResponse
    {
        $query = Movie::query()->with('genres');

        if ($request->filled('title')) {
            $query->where('title', 'like', '%' . $request->input('title')
. '%');
        }

        if ($request->filled('genres')) {
            $genreIds = explode(',', $request->input('genres'));
            $query->whereHas('genres', function ($q) use ($genreIds) {
                $q->whereIn('genres.id', $genreIds);
            });
        }

        if ($request->filled('year_from')) {
            $query->whereYear('release_year', '>=', $request-
>input('year_from'));
        }
        if ($request->filled('year_to')) {
            $query->whereYear('release_year', '<=', $request-
>input('year_to'));
        }

        // Фильтрация по актеру
        if ($request->filled('person_id')) {
            $query->whereHas('people', function ($q) use ($request) {
                $q->where('people.id', $request->input('person_id'));
            });
        }

        switch ($request->input('sort_by')) {
            case 'release_year':
                $query->orderBy('release_year', $request->input('sort_dir',
'desc'));
                break;

            case 'rating':
                $query->withAvg('reviews as average_rating', 'rating_score')
->orderByDesc('average_rating');
                break;

            default:
                $query->latest('created_at');
                break;
        }

        $movies = $query->paginate(20)->withQueryString();

        return response()->json($movies);
    }

```

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public function show(Movie $movie): JsonResponse
{
    $movie->load(['genres', 'people', 'reviews']);

    $movie->average_rating = round($movie->reviews()-
>avg('rating_score'), 1);

    return response()->json($movie);
}

public function getNewMovies(): JsonResponse
{
    $newMovies = Movie::query()
        ->with('genres')
        ->latest()
        ->get();

    return response()->json($newMovies);
}
}

```

					ПЗк.КР.Б – 121 – 25 – ПЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

FileManagementService.php

```

<?php

namespace App\Services;

use Illuminate\Http\UploadedFile;
use Illuminate\Support\Facades\Storage;

class FileManagementService
{
    public function uploadImage(UploadedFile $file, string $directory):
    ?string
    {
        $filename = uniqid() . '_' . time() . '.' . $file-
        >getClientOriginalExtension();
        $path = $file->storeAs('public/' . $directory, $filename);

        if (!$path) {
            return null;
        }

        return $directory . '/' . $filename;
    }

    public function getFileUrl(?string $path): ?string
    {
        if (!$path) {
            return null;
        }
        return Storage::url($path);
    }

    public function deleteImage(?string $path): bool
    {
        if (!$path) {
            return true;
        }

        $fullPath = 'public/' . $path;

        if (Storage::exists($fullPath)) {
            return Storage::delete($fullPath);
        }

        return true;
    }
}

```

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

UserFavoriteController.php

```

<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\Movie;
use Illuminate\Http\JsonResponse;
use Illuminate\Support\Facades\Auth;

class UserFavoriteController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth:api');
    }

    public function addToFavorites(Movie $movie): JsonResponse
    {
        $user = Auth::guard('api')->user();
        $user->favoriteMovies()->syncWithoutDetaching([$movie->id]);

        return response()->json(['message' => 'Movie added to favorites.']);
    }

    public function removeFromFavorites(Movie $movie): JsonResponse
    {
        $user = Auth::guard('api')->user();
        $user->favoriteMovies()->detach($movie->id);

        return response()->json(['message' => 'Movie removed from
favorites.']);
    }
}

```

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						76
Змн.	Арк.	№ докум.	Підпис	Дата		

ReviewController.php

```

<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\Review;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class ReviewController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth:api')->only('store');
        $this->middleware(['auth:api', 'is.admin'])->except('store');
    }

    public function store(Request $request): JsonResponse
    {
        $data = $request->validate([
            'movie_id' => 'required|integer|exists:movies,id',
            'rating_score' => 'required|integer|min:1|max:10',
            'comment_text' => 'nullable|string',
        ]);

        $user = Auth::guard('api')->user();

        $review = Review::updateOrCreate(
            ['user_id' => $user->id, 'movie_id' => $data['movie_id']],
            ['rating_score' => $data['rating_score'], 'comment_text' =>
            $data['comment_text'] ?? null]
        );

        return response()->json($review, $review->wasRecentlyCreated ? 201
: 200);
    }

    public function index(): JsonResponse
    {
        $reviews = Review::with('user:id,fullname', 'movie:id,title')->latest()->paginate(20);
        return response()->json($reviews);
    }

    public function destroy(Review $review): JsonResponse
    {
        $review->delete();
        return response()->json(null, 204);
    }
}

```

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

SearchPage.jsx

```

import { useState, useEffect } from 'react';
import { useSearchParams } from 'react-router-dom';
import api from '../services/api';
import MovieCard from '../components/MovieCard';
import Spinner from '../components/Spinner';

const SearchPage = () => {
  const [searchParams, setSearchParams] = useSearchParams();
  const [movies, setMovies] = useState([]);
  const [genres, setGenres] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  // Состояния фильтров
  const [filters, setFilters] = useState({
    title: searchParams.get('title') || '',
    genres: searchParams.getAll('genres') || [],
    year_from: searchParams.get('year_from') || '',
    sort_by: searchParams.get('sort_by') || 'release_year',
  });

  // Загрузка данных для фильтров (жанры)
  useEffect(() => {
    api.get('/genres')
      .then(res => setGenres(res.data))
      .catch(err => console.error("Failed to load genres", err));
  }, []);

  // Загрузка фильмов при изменении фильтров
  useEffect(() => {
    const fetchMovies = async () => {
      setLoading(true);
      setError(null);

      // Формируем параметры для API
      const params = {
        title: filters.title,
        genres: filters.genres.join(','),
        year_from: filters.year_from,
        sort_by: filters.sort_by,
      };

      // Обновляем URL
      setSearchParams(params, { replace: true });

      try {
        const response = await api.get('/movies/search', { params
      setMovies(response.data.data);
    } catch (err) {
      setError('Помилка завантаження фільмів.');
```

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        console.error(err);
    } finally {
        setLoading(false);
    }
};

fetchMovies();
}, [filters, setSearchParams]));

const handleFilterChange = (e) => {
    const { name, value } = e.target;
    setFilters(prev => ({ ...prev, [name]: value }));
};

const handleGenreChange = (e) => {
    const { value, checked } = e.target;
    setFilters(prev => ({
        ...prev,
        genres: checked
            ? [...prev.genres, value]
            : prev.genres.filter(id => id !== value)
    }));
};

return (
    <div className="flex flex-col md:flex-row gap-8">
        {/* --- Левая панель с фильтрами --- */}
        <aside className="w-full md:w-1/4 lg:w-1/5 space-y-6">
            <h2 className="text-2xl font-bold">Фільтри</h2>

            {/* Поиск по названию */}
            <div>
                <label htmlFor="title" className="block text-sm font-medium text-gray-300 mb-1">Назва</label>
                <input type="text" name="title" id="title" value={filters.title} onChange={handleFilterChange} className="w-full p-2 rounded bg-gray-800 border border-gray-600" />
            </div>

            {/* Фильтр по жанрам */}
            <div>
                <h3 className="text-lg font-semibold mb-2">Жанри</h3>
                <div className="space-y-2 max-h-60 overflow-y-auto">
                    {genres.map(genre => (
                        <div key={genre.id} className="flex items-center">
                            <input type="checkbox" id={`genre-${genre.id}`} value={genre.id} checked={filters.genres.includes(String(genre.id))} onChange={handleGenreChange} className="h-4 w-4 rounded border-gray-300 text-red-600 focus:ring-red-500" />

```

```

                                <label      htmlFor={`genre-${genre.id}`}
className="ml-3 text-sm text-gray-300">{genre.name}</label>
                                </div>
                            )}}
                        </div>
                    </div>

                    {/* Фільтр по року */}
                    <div>
                        <label      htmlFor="year_from"      className="block text-sm
font-medium text-gray-300 mb-1">Рік випуску (від)</label>
                        <input      type="number"      name="year_from"      id="year_from"
placeholder="Наприклад, 2010"
                                value={filters.year_from}
onChange={handleFilterChange}
                                className="w-full p-2 rounded bg-gray-800
border border-gray-600" />
                    </div>

                    {/* Сортировка */}
                    <div>
                        <label      htmlFor="sort_by"      className="block text-sm
font-medium text-gray-300 mb-1">Сортувати за</label>
                        <select      name="sort_by"      id="sort_by"
value={filters.sort_by} onChange={handleFilterChange}
                                className="w-full p-2 rounded bg-gray-800
border border-gray-600">
                            <option value="release_year">Датою виходу</option>
                            <option value="rating">Рейтингом</option>
                            <option value="created_at">Новизною</option>
                        </select>
                    </div>
                </aside>

                {/* --- Правая часть с результатами --- */}
                <main className="w-full md:w-3/4 lg:w-4/5">
                    <h1      className="text-3xl font-bold mb-6">Результати
пошуку</h1>

                    {loading ? <Spinner /> :
                    error ? <p className="text-red-400">{error}</p> :
                    movies.length > 0 ? (
                        <div      className="grid grid-cols-1 sm:grid-cols-2
lg:grid-cols-4 gap-6">
                            {movies.map(movie => <MovieCard key={movie.id}
movie={movie} />)}
                        </div>
                    ) : (
                        <p>За вашими критеріями нічого не знайдено.</p>
                    )}
                </main>
            </div>
        );
    };

    export default SearchPage;

```

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

                                ProfilePage.jsx
import { useState, useEffect } from 'react';
import useAuth from '../hooks/useAuth';
import api from '../services/api';
import Spinner from '../components/Spinner';
import MovieCard from '../components/MovieCard';

const ProfilePage = () => {
  const { user: authUser } = useAuth();
  const [profileData, setProfileData] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    api.get('/me')
      .then(response => {
        setProfileData(response.data);
      })
      .catch(error => console.error(error))
      .finally(() => setLoading(false));
  }, []);

  if (loading) {
    return <Spinner />;
  }

  if (!profileData) {
    return <p className="text-center">Не вдалося завантажити
профіль.</p>;
  }

  return (
    <div>
      <h1 className="text-4xl font-bold mb-8">Профіль користувача:
{profileData.fullname}</h1>

      </* Секция избранных фильмов */>
      <section className="mb-12">
        <h2 className="text-2xl font-semibold mb-4">Обрані
фільми</h2>
        {profileData.favorite_movies.length > 0 ? (
          <div className="grid grid-cols-2 sm:grid-cols-3
md:grid-cols-4 lg:grid-cols-5 gap-6">
            {profileData.favorite_movies.map(movie =>
              <MovieCard key={movie.id} movie={movie} />)}
          </div>
        ) : (
          <p className="text-gray-400">Ви ще не додали жодного
фільму в обране.</p>
        )}
      </section>

      </* Секция отзывов */>
      <section>

```

```

        <h2 className="text-2xl font-semibold mb-4">Ваші
відгуки</h2>
        {profileData.reviews.length > 0 ? (
          <div className="space-y-4">
            {profileData.reviews.map(review => (
              <div key={review.id} className="bg-gray-800 p-
4 rounded-lg">
                <h3 className="text-lg font-bold text-
white mb-1">
                  До фільму: <span className="text-red-
400">{review.movie.title}</span>
                </h3>
                <p className="text-yellow-400 font-bold
mb-2">★ {review.rating_score} / 10</p>
                <p
                  className="text-gray-300
italic">"{review.comment_text}"</p>
              </div>
            )
          )
        ) : (
          <p className="text-gray-400">Ви ще не залишили жодного
відгуку.</p>
        )
      </section>
    </div>
  );
};

export default ProfilePage;

```

MovieDetailPage.jsx

```

import { useState, useEffect } from 'react';
import { useParams } from 'react-router-dom';
import api from '../services/api';
import useAuth from '../hooks/useAuth';
import Spinner from '../components/Spinner';

const StarRating = ({ rating, setRating }) => {
  return (
    <div className="flex space-x-1">
      {[...Array(10)].map((_, index) => {
        const ratingValue = index + 1;
        return (
          <button
            type="button"
            key={ratingValue}
            className={ratingValue <= rating ? "text-yellow-400" : "text-gray-500"}
            onClick={() => setRating(ratingValue)}
          >
            <span className="text-2xl">★</span>
          </button>
        );
      })}
    </div>
  );
};

const MovieDetailPage = () => {
  const { isAuthenticated, user } = useAuth();
  const { id } = useParams();

  const [movie, setMovie] = useState(null);
  const [loading, setLoading] = useState(true);
  const [isFavorite, setIsFavorite] = useState(false);

  const [userReview, setUserReview] = useState(null);
  const [rating, setRating] = useState(0);
  const [comment, setComment] = useState('');
  const [reviewError, setReviewError] = useState('');

  const fetchMovieData = () => {
    api.get(`/movies/${id}`)
      .then(response => {
        const movieData = response.data;
        setMovie(movieData);

        if (user) {
          const review = movieData.reviews.find(r => r.user_id === user.id);

          if (review) {

```

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						83
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        setUserReview(review);
        setRating(review.rating_score);
        setComment(review.comment_text || '');
    }
    api.get('/me').then(profileRes => {
        const isFav =
profileRes.data.favorite_movies.some(favMovie => favMovie.id === movieData.id);
        setIsFavorite(isFav);
    });
    }
    })
    .catch(error => console.error(error))
    .finally(() => setLoading(false));
};

useEffect(() => {
    setLoading(true);
    fetchMovieData();
}, [id, user]);

const handleFavoriteToggle = () => {
    const request = isFavorite
        ? api.delete(`/favorites/${movie.id}`)
        : api.post(`/favorites/${movie.id}`);

    request
        .then(() => setIsFavorite(!isFavorite))
        .catch(err => console.error("Favorite toggle error", err));
};

const handleReviewSubmit = (e) => {
    e.preventDefault();
    if (rating === 0) {
        setReviewError('Будь ласка, поставте оцінку.');
```

```
        return;
```

```
    }
    setReviewError('');
```

```
    api.post('/reviews', {
        movie_id: movie.id,
        rating_score: rating,
        comment_text: comment,
    })
```

```
    .then(() => {
        fetchMovieData();
    })
```

```
    .catch(err => {
        setReviewError('Не вдалося відправити відгук.');
```

```
    });
```

```
    if (loading) {
        return <Spinner />;
    }
}
```

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						84
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    if (!movie) {
      return <p>Фільм не знайдено.</p>;
    }

    const directors = movie.people.filter(p => p.pivot.role_type === 2);
    const actors = movie.people.filter(p => p.pivot.role_type === 1);

    return (
      <div className="flex flex-col md:flex-row gap-8">
        <div className="md:w-1/3">
          <img
            src={movie.poster_url}
            alt={movie.title} className="rounded-lg w-full" />
          {isAuthenticated && (
            <button
              onClick={handleFavoriteToggle}
              className={`w-full mt-4 py-3 rounded-lg font-bold
transition-colors ${
                isFavorite
                  ? 'bg-red-800 text-red-300 hover:bg-red-
900'
                  : 'bg-red-600 text-white hover:bg-red-700'
              }}`>
              {isFavorite ? '✓ В обраному' : '♥ Додати в обране'}
            </button>
          )}
          {movie.trailer_url && (
            <a
              href={movie.trailer_url}
              target="_blank"
              rel="noopener noreferrer"
              className="w-full block text-center py-3 rounded-lg
font-bold transition-colors bg-gray-700 text-white hover:bg-gray-600 mt-5"
            >
              ▷ Дивитись трейлер
            </a>
          )}
        </div>
        <div className="md:w-2/3">
          <h1 className="text-4xl font-bold mb-2">{movie.title}</h1>
          <div className="flex flex-wrap items-center gap-x-4 mb-4
text-gray-400">
            <span>{new
Date(movie.release_date).getFullYear()}</span>
            <span>•</span>
            <span>{movie.genres.map(g => g.name).join(',
')}</span>
            <span>•</span>
            <span>{movie.duration_minutes} хв.</span>
          </div>
        </div>
      </div>
    )
  }
}

```

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<div className="flex items-center mb-6">
  <span className="text-2xl font-bold text-yellow-400
mr-2">★ {movie.average_rating || 'N/A'}</span>
  <span className="text-gray-
400">({movie.reviews.length} відгуків)</span>
</div>

<h2 className="text-2xl font-semibold mb-2">Опис</h2>
<p className="mb-6">{movie.description}</p>

<div className="mb-6">
  <h3 className="font-semibold">Режисери:</h3>
  <p>{directors.map(d => d.full_name).join(', ')}</p>
</div>
<div>
  <h3 className="font-semibold">У ролях:</h3>
  <p>{actors.map(a => a.full_name).join(', ')}</p>
</div>

{isAuthenticated && (
  <div className="mt-8 bg-gray-800 p-6 rounded-lg">
    <h2 className="text-2xl font-semibold mb-4">
      {userReview ? 'Ваш відгук' : 'Залишити відгук'}
    </h2>
    <form onSubmit={handleReviewSubmit}>
      <div className="mb-4">
        <label className="block mb-2 font-
medium">Ваша оцінка:</label>
        <StarRating rating={rating}
setRating={setRating} />
      </div>
      <div className="mb-4">
        <label htmlFor="comment" className="block
mb-2 font-medium">Ваш коментар:</label>
        <textarea id="comment" value={comment}
onChange={e => setComment(e.target.value)}
rows="4" placeholder="Поділіться
своїми враженнями..."
className="w-full p-2 rounded
bg-gray-700 border border-gray-600 focus:outline-none focus:ring-2 focus:ring-
red-500"
        ></textarea>
      </div>
      {reviewError && <p className="text-red-400 mb-
4">{reviewError}</p>}
      <button type="submit" className="bg-red-600
hover:bg-red-700 text-white font-bold py-2 px-6 rounded">
        {userReview ? 'Оновити відгук' :
'Відправити відгук'}
      </button>
    </form>
  </div>
)}

<div className="mt-8">

```

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						86
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        <h2 className="text-2xl font-semibold mb-4">Відгуки
інших</h2>
        <div className="space-y-4">
            {movie.reviews.length > 0 ?
movie.reviews.map(review => (
                <div key={review.id} className="bg-gray-800
p-4 rounded-lg">
                    <div className="flex items-center mb-2">
                        <p className="font-bold mr-
4">{review.user.fullname}</p>
                        <p className="text-yellow-400 font-
bold">★ {review.rating_score}</p>
                    </div>
                    <p className="text-gray-
300">{review.comment_text}</p>
                </div>
            )) : <p>Відгуків ще немає.</p>}
        </div>
    </div>
</div>
);
};

export default MovieDetailPage;

```

					ІІЗк.КР.Б – 121 – 25 – ІІЗ	Арк.
						87
Змн.	Арк.	№ докум.	Підпис	Дата		