

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:

«Онлайн-кінотеатр»

Виконав студент групи ІПЗ-21-3
БАРАБАШ Владислав Віталійович

Керівник роботи:
ЧИЖМОТРИ Олена Геннадіївна

Рецензент:
ЛОБАНЧИКОВА Надія Миколаївна

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: **Барабаш Владислав Віталійович**

Керівник роботи: **Чижмотря Олена Геннадіївна**

Тема роботи: **«Онлайн-кінотеатр»**,

затверджена Наказом закладу вищої освіти від **14.02.2025р., №61/с**

Вихідні дані для роботи: Процес автоматизації організації доступу до медіаконтенту через онлайн-кінотеатр із використанням сучасних інформаційних технологій. Особлива увага приділяється розробці вебзастосунку для зручного перегляду фільмів та серіалів, впровадженню системи рекомендацій, фільтрації контенту за жанрами та рейтингами, а також персоналізації інтерфейсу відповідно до вподобань користувача.

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.Г	20.03.2025	20.03.2025
2	Чижмотря О.Г	20.03.2025	20.03.2025
3	Чижмотря О.Г	20.03.2025	20.03.2025

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з вебзастосунку – онлайн-кінотеатру для перегляду фільмів і серіалів з функціоналом персоналізації – та пояснювальної записки. Пояснювальна записка до роботи містить 82 сторінки, 25 ілюстрацій, 11 таблиць, 1 додаток та список із 31 використаного джерела.

Метою роботи є створення сучасного, зручного та функціонального онлайн-сервісу, що дозволяє користувачам переглядати фільми, створювати особисті списки перегляду, залишати оцінки та рецензії, обговорювати контент з іншими користувачами та отримувати персоналізовані рекомендації. Платформа орієнтована на широку аудиторію глядачів, яка зацікавлена в зручному способі доступу до медіаконтенту та взаємодії в рамках тематичної спільноти.

Об'єктом дослідження є процес створення онлайн-платформи для перегляду медіаконтенту з елементами соціальної взаємодії.

Предметом дослідження є методи розробки адаптивного вебзастосунку з реалізацією функцій відеоплеєра, роботи з базами даних фільмів і серіалів, авторизації користувачів, рейтингування, коментування, пошуку та рекомендаційного механізму.

У роботі застосовано методи об'єктно-орієнтованого програмування, побудови REST API, інтеграції з базами даних, інструменти фронтенд- та бекенд-розробки, адаптивного дизайну, а також механізми забезпечення інтерактивності та взаємодії користувачів. Особливу увагу приділено зручності інтерфейсу, якості відтворення відео та можливості персоналізації контенту.

КЛЮЧОВІ СЛОВА: ОНЛАЙН-КІНОТЕАТР, ВЕБЗАСТОСУНОК, ФІЛЬМИ, СЕРІАЛИ, ПЕРСОНАЛІЗАЦІЯ, ІНТЕРФЕЙС КОРИСТУВАЧА, ОБГОВОРЕННЯ, ВЕБ-ТЕХНОЛОГІЇ.

					ІПЗ.КР.Б – 121 – 25 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		В.В. Барабаш			Онлайн-кінотеатр Пояснювальна записка	Літ.	Арк.	Аркушів
Керівник		О.Г. Чижмотря					3	82
Рецензент		Н.М. Лобанчикова				Житомирська Політехніка, група ІПЗ-21-3		
Зав. кафедри		Т.А. Вакалюк						

ABSTRACT

The bachelor's qualification thesis consists of a web application - an online cinema for watching movies and TV series with personalization functionality - and an explanatory report. The explanatory report contains 82 pages, 25 figures, 11 tables, 1 appendix, and a list of 31 references used.

The aim of the thesis is to develop a modern, user-friendly, and functional online service that allows users to watch movies, create personal watchlists, leave ratings and reviews, discuss content with other users, and receive personalized recommendations. The platform is designed for a broad audience of viewers interested in convenient access to media content and interaction within a themed community.

The object of the research is the process of creating an online platform for accessing media content with elements of social interaction.

The subject of the research is the methods of developing an adaptive web application implementing features such as a video player, interaction with movie and TV series databases, user authentication, rating systems, commenting, content search, and a recommendation mechanism.

The project employs object-oriented programming methods, REST API development, database integration, frontend and backend development tools, responsive design, and mechanisms ensuring interactivity and user engagement. Special attention is given to interface usability, content quality, and personalization capabilities.

KEYWORDS: ONLINE CINEMA, WEB APPLICATION, MOVIES, TV SERIES, PERSONALIZATION, USER INTERFACE, DISCUSSION, WEB TECHNOLOGIES.

		<i>В.В. Барабаш</i>			<i>ІПЗ.КР.Б – 121 – 25 – ПЗ</i>	<i>Арк.</i>
		<i>О.Г. Чижмотря</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		<i>4</i>

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ	9
1.1. Постановка задачі.....	9
1.2. Аналіз аналогів програмного продукту	10
1.3. Вибір архітектури платформи онлайн-кінотеатру.....	15
1.4. Обґрунтування вибору інструментальних засобів.....	16
Висновки до першого розділу.....	20
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ПЛАТФОРМИ ОНЛАЙН- КІНОТЕАТРУ	21
2.1. Визначення варіантів використання та структури системи.....	21
2.2. Розробка бази даних.....	23
2.3. Проєктування та реалізація алгоритмів роботи	33
2.4. Реалізація функціоналу.....	36
Висновки до другого розділу	41
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУ	42
3.1. Порядок встановлення та налаштування системи	42
3.2. Структура інтерфейсу та порядок взаємодії з інтернет-магазином ювелірних виробів.....	43
3.3. Тестування роботи програмного продукту	53
Висновки до третього розділу.....	56
ВИСНОВКИ.....	57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	59
ДОДАТКИ.....	62

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення

ІТ – інформаційні технології

БД – база даних

СУБД – система управління базами даних

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

JS – JavaScript

API – Application Programming Interface

REST – Representational State Transfer

CRUD – Create, Read, Update, Delete

HTTPS – Hyper Text Transfer Protocol Secure

JWT – JSON Web Token

SPA – Single Page Application

ID – Identifier

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. У сучасному світі цифровізація та інтернет-технології змінюють способи споживання медіаконтенту, зокрема кіно. Зростаюча популярність стрімінгових сервісів та онлайн-кінотеатрів свідчить про те, що користувачі все частіше віддають перевагу зручним, доступним та персоналізованим платформам для перегляду фільмів та серіалів. Проте багато існуючих онлайн-кінотеатрів мають ряд обмежень: застарілі інтерфейси, недостатню швидкість завантаження контенту, відсутність гнучкої системи рекомендацій або обмежений функціонал для взаємодії користувачів. Створення сучасного веб-додатку онлайн-кінотеатру, який поєднує у собі зручний інтерфейс, високу продуктивність та широкий набір інструментів для користувачів, є актуальним завданням, що відповідає потребам сучасної аудиторії.

Метою роботи є розробка вебплатформи для перегляду фільмів і серіалів, яка надає користувачам можливість створювати та редагувати персоналізовані списки перегляду, відстежувати прогрес, залишати оцінки й коментарі, а також взаємодіяти з іншими користувачами в зручному онлайн-середовищі. Платформа має забезпечити високий рівень користувацької взаємодії, персоналізації контенту та інтеграції з іншими учасниками спільноти, створюючи комфортну атмосферу для перегляду медіаконтенту.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Проаналізувати сучасні рішення у сфері онлайн-кінотеатрів.
2. Визначити основні вимоги користувачів до функціоналу платформи.
3. Спроекувати архітектуру вебзастосунку та бази даних.
4. Реалізувати функціонал реєстрації, аутентифікації та профілів користувачів.
5. Створити систему рейтингу, коментування, списків перегляду та рекомендацій.
6. Забезпечити адаптивність інтерфейсу та зручність користування.
7. Протестувати додаток і оптимізувати його продуктивність.

		<i>В.В. Барабаш</i>			<i>ІПЗ.КР.Б – 121 – 25 – ПЗ</i>	<i>Арк.</i>
		<i>О.Г. Чижмотря</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		<i>7</i>

Об'єктом дослідження є процес створення веб-додатку онлайн-кінотеатру, який охоплює аналіз потреб користувачів, проектування інтерфейсів, розробку функціоналу та інтеграцію зі зовнішніми сервісами.

Предметом дослідження є сучасні веб-технології, фреймворки та інструменти, необхідні для реалізації ефективного онлайн-кінотеатру, зокрема механізми рекомендацій, системи пошуку контенту, управління користувачами та інші функціональні компоненти.

		<i>В.В. Барабаш</i>			<i>ІПЗ.КР.Б – 121 – 25 – ПЗ</i>	<i>Арк.</i>
		<i>О.Г. Чижмотря</i>				8
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1. Постановка задачі

Сучасна цифрова епоха суттєво трансформувала способи споживання медіаконтенту, що призвело до стрімкого розвитку онлайн-платформ у сфері дозвілля. Онлайн-кінотеатри стали важливою складовою медіаринку, оскільки задовольняють потребу користувачів у швидкому, зручному та персоналізованому доступі до фільмів і серіалів. Зі зростанням конкуренції на ринку розважального контенту та підвищенням вимог з боку аудиторії, необхідність у створенні інноваційних вебрішень стає дедалі актуальнішою.

Розробка онлайн-кінотеатру передбачає створення повноцінної вебплатформи, що поєднує в собі сучасний дизайн, інтуїтивно зрозумілий інтерфейс, зручну навігацію по каталогу контенту, механізми персоналізації перегляду, інтеграцію з базами даних фільмів та реалізацію функцій взаємодії між користувачами. Важливим аспектом є забезпечення системи реєстрації, збереження прогресу перегляду, можливості формування власних списків, оцінювання контенту, додавання рецензій та коментарів.

Окрему увагу варто приділити адміністративній частині системи, яка дозволяє керувати наповненням каталогу, оновлювати інформацію про медіапродукти, модерацію контенту та аналіз статистики взаємодії користувачів з платформою. Для підвищення ефективності обслуговування та зворотного зв'язку можна також впровадити засоби повідомлень або чат-боти для оперативної взаємодії з користувачами.

Таким чином, створення онлайн-кінотеатру є комплексним проєктом, що охоплює не лише технічну реалізацію функціональних можливостей платформи, а й врахування особливостей поведінки глядацької аудиторії, вимог до зручності, швидкодії, візуальної привабливості та стабільності роботи системи в умовах значного навантаження.

		<i>В.В. Барабаш</i>			<i>ІПЗ.КР.Б – 121 – 25 – ПЗ</i>	<i>Арк.</i>
		<i>О.Г. Чижмотря</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		<i>9</i>

1.2. Аналіз аналогів програмного продукту

Для розробки ефективного веб-додатку онлайн-кінотеатру необхідно провести аналіз існуючих аналогів, щоб визначити їхні переваги та недоліки, а також виявити ключові функціональні можливості, які слід врахувати при створенні власного продукту. У цьому розділі буде проведено порівняльний аналіз декількох популярних онлайн-кінотеатрів, таких як Netflix, Amazon Prime Video, Disney+ та HBO Max [1-4], з метою визначення їхніх основних характеристик та виявлення напрямків для покращення.

Netflix

Netflix є одним із найбільш популярних онлайн-кінотеатрів у світі. Платформа пропонує широкий каталог фільмів, серіалів, документальних проєктів та оригінального контенту (рис.1.1). Серед ключових переваг Netflix варто відзначити персоналізовану систему рекомендацій, яка завдяки алгоритмам машинного навчання пропонує користувачам індивідуальні добірки фільмів та серіалів відповідно до їхніх вподобань. Платформа забезпечує високу якість перегляду, підтримуючи стрімінг до 4K та адаптивну передачу даних, що дозволяє зберігати стабільність перегляду навіть за умов нестабільного інтернет-з'єднання. Додатковою перевагою є мультиплатформова підтримка: доступ до сервісу можливий з різних пристроїв, включаючи смартфони, планшети, Smart TV та ігрові консолі. Крім того, користувачі мають змогу завантажувати контент для офлайн-перегляду. Водночас, Netflix має і деякі недоліки. Зокрема, відсутність реклами, хоча й є перевагою для користувачів, обмежує можливості монетизації для самої платформи. Також варто зазначити, що підписка на Netflix є однією з найдорожчих серед онлайн-кінотеатрів, що може бути стримувальним чинником для частини потенційних користувачів.

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижморя				10
Змн.	Арк.	№ докум.	Підпис	Дата		

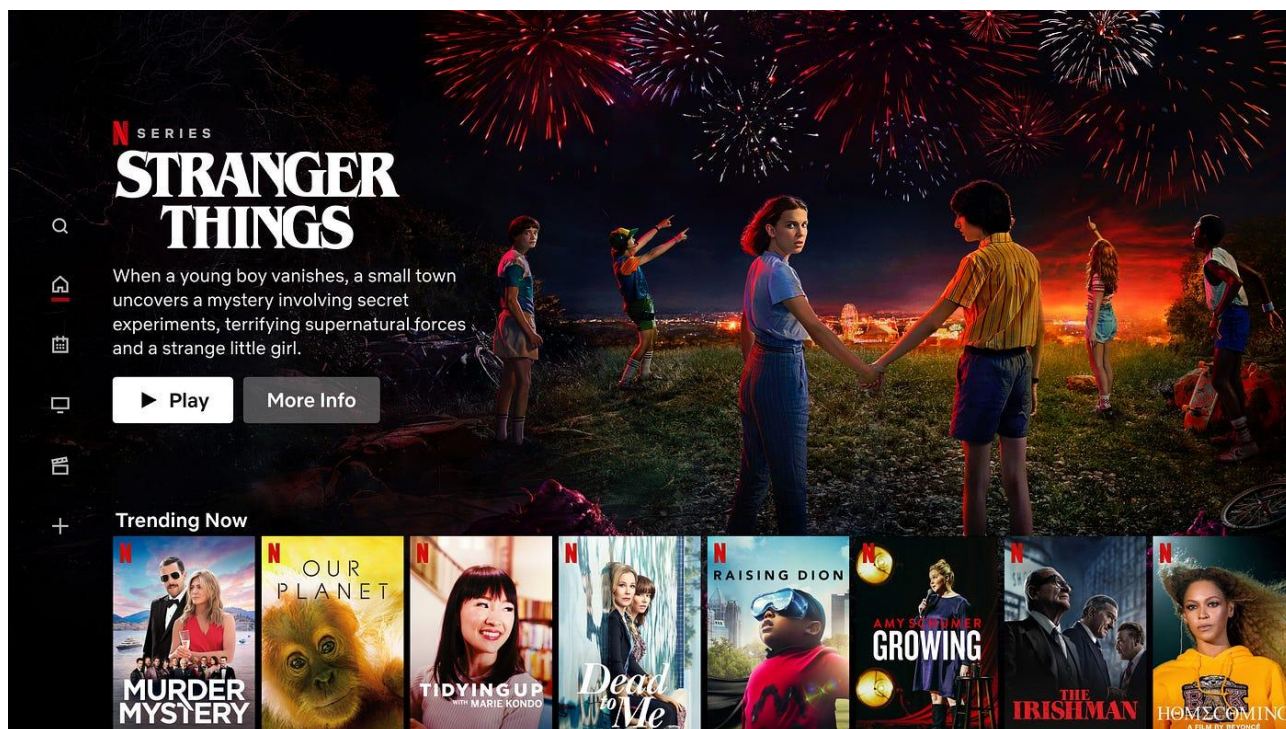


Рис.1.1 – Інтерфейс сайту Netflix

Amazon Prime Video

Amazon Prime Video є частиною екосистеми Amazon і пропонує великий каталог фільмів та серіалів, включаючи ексклюзивний оригінальний контент (рис.1.2.). Однією з ключових переваг платформи є інтеграція з Amazon, що дозволяє користувачам зручно купувати або орендувати додаткові фільми через Amazon Store, розширюючи можливості монетизації для сервісу. Також сервіс вирізняється широким вибором контенту, зокрема фільмів, серіалів і регіонального відео, що дозволяє охопити широку аудиторію. Amazon Prime Video підтримує стрімінг у високій якості, включаючи Ultra HD та технології HDR, що забезпечує покращене зображення та комфортний перегляд. Водночас платформа має й певні недоліки. Зокрема, деякі користувачі відзначають складність і неінтуїтивність інтерфейсу, що може ускладнювати навігацію. Крім того, система персоналізації контенту поступається конкурентам – рекомендації не завжди точно відповідають інтересам користувача, що знижує ефективність залучення до нового контенту.

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				11
Змн.	Арк.	№ докум.	Підпис	Дата		

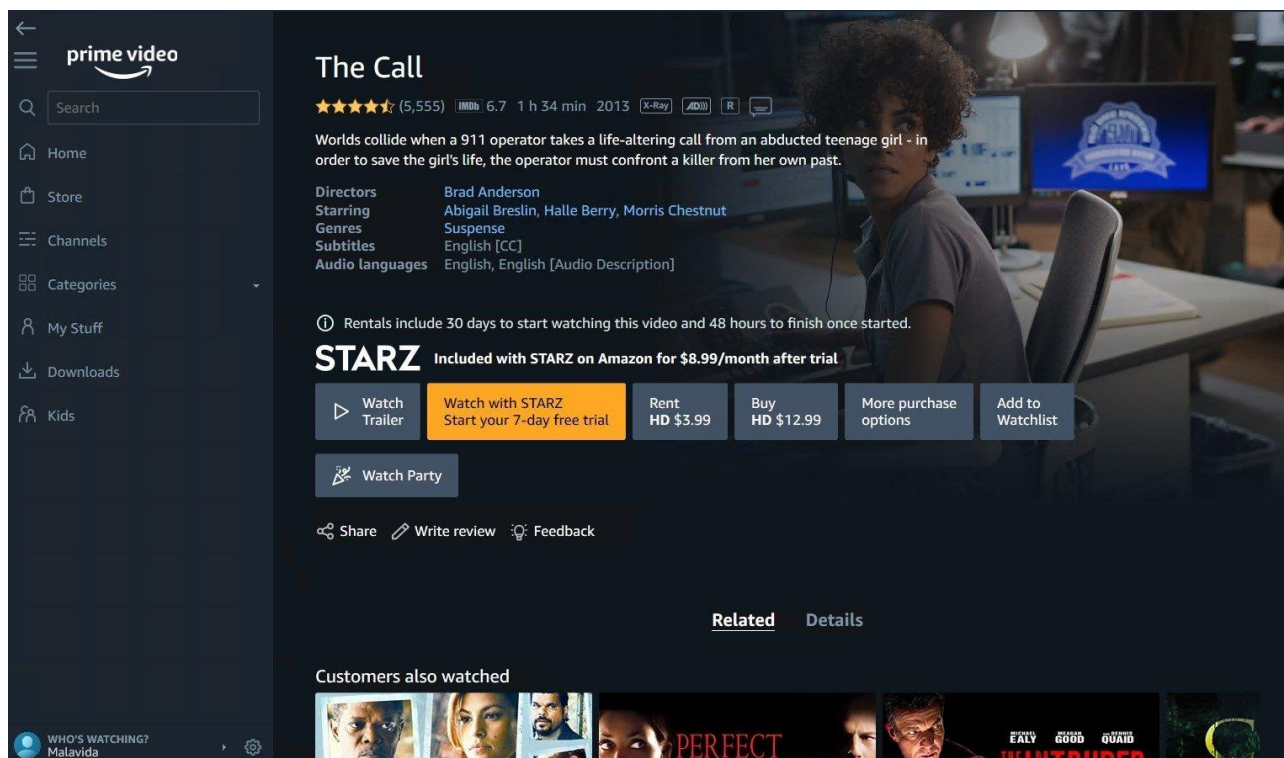


Рис. 1.2 – Інтерфейс сайту Amazon Prime Video

Disney+

Disney+ є відносно новою платформою на ринку онлайн-кінотеатрів, яка за короткий час здобула широку популярність завдяки великій кількості оригінального контенту від відомих брендів, таких як Marvel, Pixar, Lucasfilm та інших (рис 1.3). Однією з головних переваг Disney+ є ексклюзивність – сервіс пропонує фільми та серіали, які недоступні на інших платформах, що робить його привабливим для фанатів конкретних франшиз. Крім того, платформа має чітко виражену сімейну орієнтацію: контент адаптований для спільного перегляду всією родиною, що значно розширює її цільову аудиторію. Ще однією конкурентною перевагою є доступна ціна підписки, яка робить сервіс привабливим навіть для користувачів із обмеженим бюджетом. Водночас Disney+ має і деякі обмеження. Зокрема, через фокус на власному контенті платформа не може похизуватися таким різноманіттям фільмів і серіалів, як Netflix або Amazon Prime Video. Також, як і у випадку з Netflix, відсутність реклами хоча й є перевагою для користувачів, обмежує платформу у використанні додаткових джерел прибутку.

		V.B. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				12
Змн.	Арк.	№ докум.	Підпис	Дата		

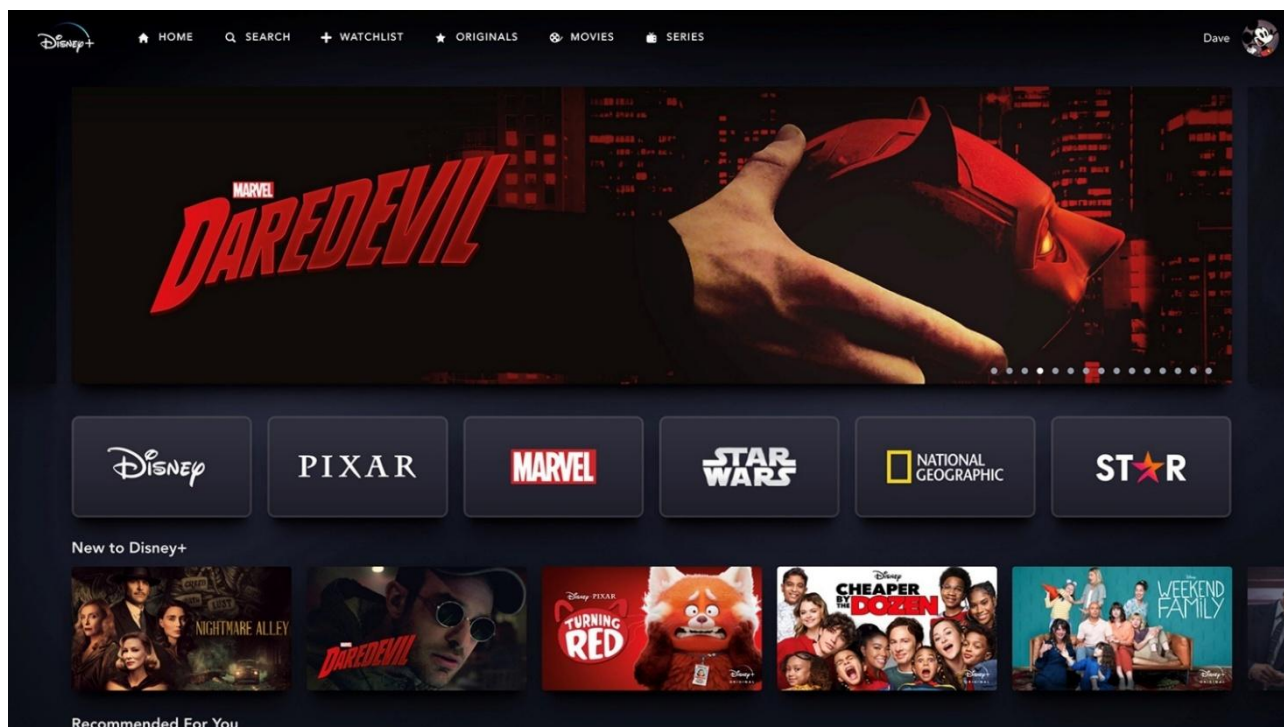


Рис. 1.3 –Сторінка товару Інтерфейс сайту Disney+

HBO Max

HBO Max є платформою, яка об'єднує контент від таких відомих брендів, як HBO, Warner Bros., DC та інших, і пропонує користувачам велику бібліотеку оригінальних серіалів, фільмів, документальних проєктів та анімації (рис 1.4.). Однією з ключових переваг сервісу є високоякісний контент, що став фірмовою ознакою бренду HBO: платформа відома серіалами з глибокими сюжетами, продуманою режисурою та високим рівнем акторської гри. Крім того, HBO Max вирізняється жанровим розмаїттям – користувачам доступний широкий спектр фільмів, серіалів, документальних стрічок і анімаційних проєктів, що дозволяє охопити інтереси різних категорій глядачів. Платформа підтримує стрімінг у високій якості, зокрема Ultra HD, що забезпечує комфортний перегляд. Водночас сервіс має і певні недоліки. Зокрема, підписка на HBO Max є однією з найдорожчих серед онлайн-кінотеатрів, що може бути бар'єром для деяких користувачів. Окрім того, платформа має обмежену географічну доступність і офіційно працює не в усіх країнах, що ускладнює доступ до контенту для міжнародної аудиторії.

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				13
Змн.	Арк.	№ докум.	Підпис	Дата		

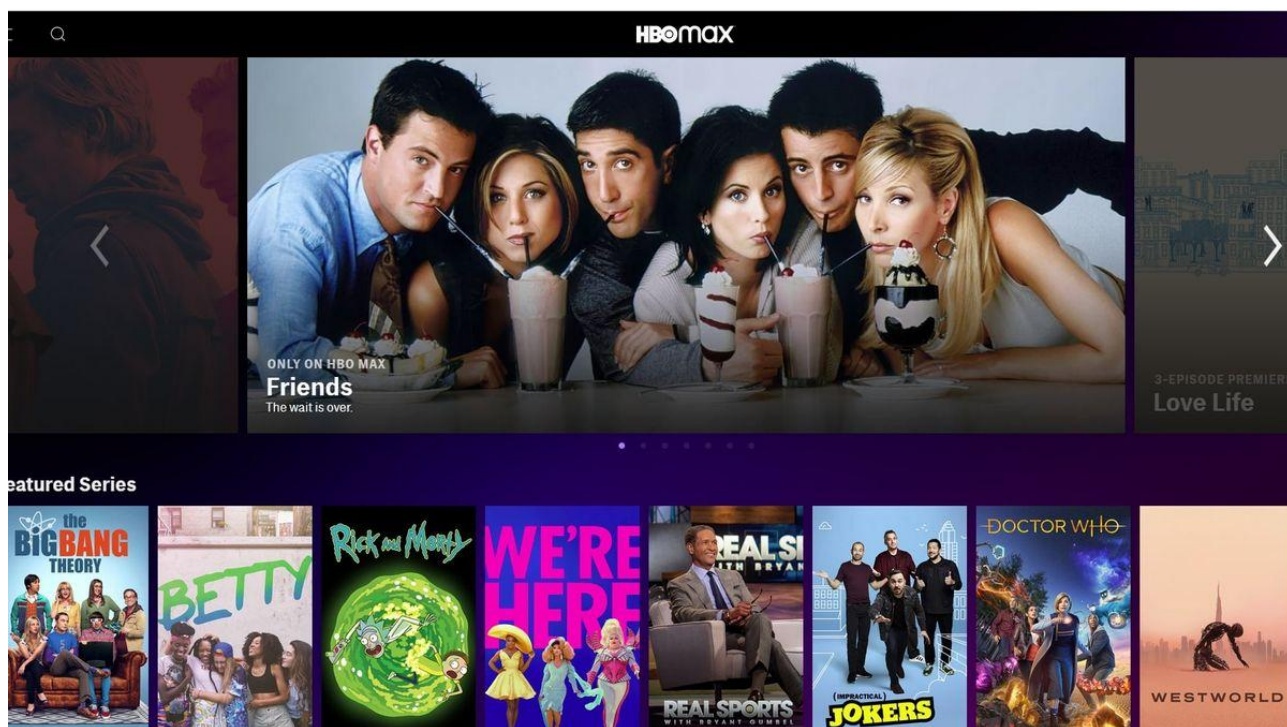


Рис.1.4 – Інтерфейс сайту HBO Max

На основі аналізу існуючих аналогів можна виділити низку ключових функціональних можливостей, які варто врахувати під час розробки власного веб-додатку онлайн-кінотеатру. Однією з основних є персоналізована система рекомендацій, що сприяє покращенню користувацького досвіду завдяки адаптації контенту до інтересів і поведінки конкретного глядача. Важливим також є забезпечення високої якості відтворення відео, включаючи підтримку роздільної здатності до 4K та впровадження адаптивної передачі даних, що дозволяє зберігати стабільність перегляду незалежно від швидкості інтернет-з'єднання. Додатковою функціональністю, яка може значно підвищити зручність користування платформою, є можливість офлайн-перегляду, що особливо актуально для користувачів, які часто перебувають у зонах з обмеженим доступом до мережі. Таким чином, аналіз аналогічних сервісів дає змогу окреслити основні напрями розвитку та сформулювати перелік функцій, необхідних для створення сучасного, зручного та конкурентоспроможного веб-застосунку онлайн-кінотеатру.

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				14
Змн.	Арк.	№ докум.	Підпис	Дата		

1.3. Вибір архітектури платформи онлайн-кінотеатру

Для реалізації вебплатформи онлайн-кінотеатру було обрано клієнт-серверну архітектуру з чітким розділенням фронтенд- та бекенд-частин. Такий підхід забезпечує гнучкість, масштабованість і спрощення розробки, дозволяючи незалежно розвивати інтерфейс користувача та серверну логіку. У якості клієнтської частини обрано бібліотеку React [5], що дозволяє створювати сучасні, швидкі та інтерактивні інтерфейси. Для розробки серверної частини використовується платформа Node.js з вебфреймворком Express, який забезпечує обробку HTTP-запитів, взаємодію з базою даних, автентифікацію користувачів та реалізацію бізнес-логіки.

Клієнтська частина відповідає за рендеринг основних сторінок платформи, включаючи головну, каталог фільмів, детальні сторінки медіапродуктів, особистий кабінет, форму входу/реєстрації тощо. Користувацькі дії, такі як фільтрація контенту, додавання у список улюбленого, оцінювання чи залишення коментаря, викликають HTTP-запити до серверної частини через REST API. Реалізований API забезпечує передачу, обробку та зберігання даних у базі, а також відповідає за виконання логіки авторизації, рекомендацій та управління контентом.

Серверна частина на базі Express [6] реалізує обробку запитів типу GET, POST, PUT і DELETE, а також виконує перевірку прав доступу, валідацію даних, створення й редагування профілів, управління відгуками та обробку статистики. Уся необхідна інформація зберігається у базі даних, з якою сервер взаємодіє через відповідні ORM-засоби або драйвери.

Клієнт-серверна модель дозволяє легко масштабувати додаток, додаючи нові модулі (наприклад, адмінпанель чи аналітичний модуль) або інтегруючи сторонні сервіси – такі як платіжні системи, чат-платформи чи сервіси аналітики. Вона також сприяє підвищенню безпеки, оскільки обробка критичних операцій відбувається на сервері, а на клієнтському рівні реалізується лише візуалізація та взаємодія з користувачем.

		<i>В.В. Барабаш</i>			<i>ІПЗ.КР.Б – 121 – 25 – ПЗ</i>	<i>Арк.</i>
		<i>О.Г. Чижмоторя</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		<i>15</i>

Таким чином, обрана клієнт-серверна архітектура з React на фронтенді та Express на бекенді є ефективним, надійним і сучасним рішенням для розробки онлайн-кінотеатру, яке забезпечує високий рівень продуктивності, безпеки й масштабованості платформи.

На рис. 1.5 представлено загальний принцип взаємодії між клієнтською та серверною частинами у межах клієнт-серверної архітектури вебзастосунку онлайн-кінотеатру. З клієнтського інтерфейсу, створеного на React, надсилаються запити до бекенду на Express через REST API. Сервер обробляє ці запити, взаємодіє з базою даних, виконує авторизацію, зберігає або змінює дані й повертає відповіді на клієнт.

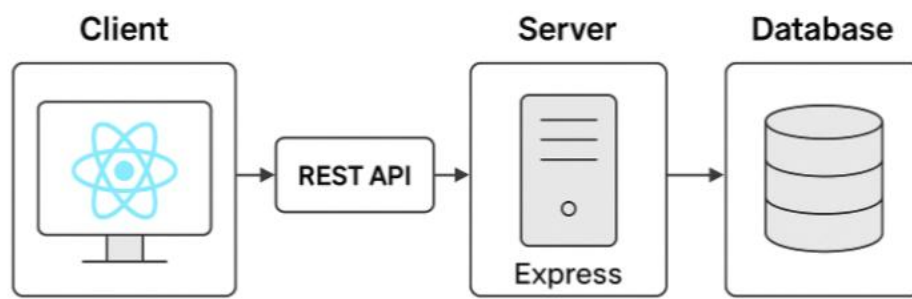


Рис. 1.5 – Загальна схема архітектури онлайн-кінотеатру з клієнт-серверною взаємодією

1.4. Обґрунтування вибору інструментальних засобів

Для реалізації онлайн-кінотеатру було використано сучасний набір інструментальних засобів, що забезпечують гнучкість, високу продуктивність, безпеку та зручність у розробці як клієнтської, так і серверної частин застосунку. Архітектура проєкту передбачає чітке розділення на фронтенд і бекенд, з комунікацією через REST API.

Клієнтська частина створена з використанням бібліотеки React, яка дозволяє будувати динамічний, модульний та інтуїтивно зрозумілий інтерфейс. Для стилізації використовується SASS [6], що забезпечує гнучкість у побудові адаптивного дизайну. Бібліотека Material-UI [7] застосовується для створення зручного і сучасного UI з готовими компонентами та іконками. Навігація між сторінками реалізована за допомогою React Router DOM [8], що дозволяє

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				
Змн.	Арк.	№ докум.	Підпис	Дата		16

будувати SPA-застосунок. Для роботи з API використовується бібліотека Axios, яка забезпечує просту та ефективну реалізацію HTTP-запитів. Компонент react-youtube дозволяє вбудовувати трейлери та інші відеоматеріали безпосередньо на сторінки фільмів.

Бекенд-частина побудована на базі Node.js [9] з використанням легкого і гнучкого фреймворку Express, який забезпечує маршрутизацію, обробку запитів і реалізацію REST API. Для роботи з базою даних використовується Mongoose – обгортка над MongoDB [10], що дозволяє зручно описувати схеми даних і працювати з документно-орієнтованою структурою. Бібліотека dotenv використовується для безпечного зберігання середовищних змінних, а morgan – для логування запитів у процесі розробки.

Для автентифікації та авторизації застосовується jsonwebtoken (JWT) [11], що дозволяє створювати захищені маршрути й контролювати доступ до контенту. Захист та шифрування чутливих даних реалізовано за допомогою crypto-js. Для обробки файлів (наприклад, зображень постерів чи профільних аватарів) використовується multer у поєднанні з multer-storage-cloudinary. Сам Cloudinary [12] забезпечує зберігання, трансформацію та оптимізацію мультимедійного контенту в хмарі. Для відправки електронних повідомлень (наприклад, підтвердження реєстрації чи відновлення паролю) використовується Nodemailer.

У межах реалізації функціоналу оплати впроваджено інтеграцію з Stripe – надійним сервісом для обробки онлайн-платежів, що підтримує різноманітні платіжні методи та гарантує високий рівень безпеки транзакцій [13]. Це забезпечує можливість реалізації платного доступу до преміум-контенту чи підписки.

Для автоматичного перезапуску сервера під час розробки використовується Nodemon [14], що підвищує ефективність і пришвидшує цикл розробки.

Таким чином, обрані технології повністю відповідають вимогам сучасного вебзастосунку для онлайн-кінотеатру. Вони дозволяють реалізувати

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				17
Змн.	Арк.	№ докум.	Підпис	Дата		

масштабовану, безпечну, продуктивну та зручну в користуванні платформу з широким функціоналом – від реєстрації й перегляду фільмів до взаємодії з іншими користувачами та онлайн-оплати.

Під час вибору оптимальної системи керування базами даних для онлайн-кінотеатру було проаналізовано кілька поширених рішень, серед яких – MongoDB, PostgreSQL [15] та MySQL [16]. Кожна з цих СУБД має свої сильні сторони та обмеження, які по-різному впливають на ефективність розробки та масштабованість застосунку. Для порівняння було розглянуто ключові технічні параметри: обсяг збережених даних, підтримку транзакцій, а також можливості роботи з вкладеними структурами та напівструктурованою інформацією.

Як видно з таблиці 1.1, усі три СУБД здатні обробляти значні обсяги даних. MongoDB підтримує горизонтальне масштабування через шардинг, що робить її придатною для високонавантажених систем. PostgreSQL також добре справляється з великими таблицями та має інструменти для розподіленого зберігання. MySQL у цьому плані трохи поступається через більшу залежність від файлової системи та типу таблиць.

Таблиця 1.1

Можливості зберігання даних у різних СУБД

СУБД	Орієнтовний ліміт даних	Технічні зауваження
MongoDB	До 1 ТБ на колекцію (у 32-бітній системі)	Завдяки шардингу обробляє петабайти; ліміт залежить від файлової системи
PostgreSQL	До 64 ТБ на таблицю (у 64-бітній системі)	Підтримує розподілення великих таблиць, добре оптимізується
MySQL	Теоретично необмежено, практично – кілька ТБ	Реальні обмеження залежать від InnoDB і файлової системи

У таблиці 1.2 наведено характеристики транзакційної моделі. У PostgreSQL транзакції підтримуються на повному рівні, включаючи вкладені транзакції та контроль цілісності. У MongoDB така функціональність з'явилася

порівняно нещодавно, проте вона вже дозволяє працювати з ACID-гарантіями в кількох колекціях. У MySQL усе залежить від обраного типу таблиць – повна підтримка можлива тільки при використанні InnoDB.

Таблиця 1.2

СУБД	Наявність транзакцій	Особливості реалізації
MongoDB	Так (з версії 4.0+)	Працює між колекціями, але з деякими обмеженнями
PostgreSQL	Повна підтримка ACID	Надійна реалізація з контролем зв'язності й вкладеністю
MySQL	Так, за умови InnoDB	Умовна підтримка, не однакова для всіх типів таблиць

Ще один важливий аспект – гнучкість у роботі з вкладеними структурами. В онлайн-кінотеатрі активно використовуються масиви жанрів, списки перегляду, коментарі, рейтинги – тобто дані з вкладеною або напівструктурованою структурою. У MongoDB така модель зберігання реалізується природно через JSON-подібні документи. У випадку з PostgreSQL та MySQL реалізація таких структур потребує складної системи зовнішніх ключів і додаткових таблиць.

Таблиця 1.3

Гнучкість роботи з вкладеними даними

СУБД	Робота з вкладеними структурами	Придатність для напівструктурованих даних
MongoDB	Вбудована підтримка	Висока – зберігає вкладені масиви, об'єкти тощо
PostgreSQL	Через JSONB або окремі таблиці	Можливо, але менш гнучко і з нижчою продуктивністю
MySQL	Обмежена	Обробка складних структур ускладнена або неможлива

З огляду на динамічну структуру даних, гнучку модель взаємодії користувачів з фільмами та потребу у швидкій розробці функціоналу, MongoDB є найбільш відповідним варіантом для реалізації даного онлайн-кінотеатру. Вона дозволяє зберігати складні об'єкти без зайвої нормалізації, швидко масштабувати систему при збільшенні кількості контенту та користувачів, а також забезпечує просту інтеграцію з Node.js через бібліотеку mongoose.

Крім того, MongoDB дозволяє зосередитись на бізнес-логіці проєкту, а не на складній побудові реляційної схеми, що є великою перевагою при розробці застосунку з великою кількістю взаємодій та змін у структурі даних.

Таким чином, MongoDB було обрано як оптимальне рішення, яке забезпечує баланс між гнучкістю, продуктивністю та масштабованістю в контексті створення сучасного онлайн-кінотеатру.

Висновки до першого розділу

У першому розділі проаналізовано актуальність створення онлайн-кінотеатру та вимоги користувачів до зручного і функціонального вебзастосунку. Розглянуто приклади відомих платформ, що дозволило визначити ключові функції: персоналізація, каталогізація, рейтингування, коментарі тощо.

Обґрунтовано вибір клієнт-серверної архітектури з фронтендом на React та бекендом на Express, що забезпечує гнучкість, масштабованість і зручну інтеграцію з REST API. Вибрані інструменти дозволяють швидко розробляти адаптивний і стабільний застосунок.

Проведене порівняння СУБД засвідчило доцільність використання MongoDB – завдяки її гнучкості, підтримці вкладених структур і ефективній інтеграції з Node.js. Це створює надійну технічну основу для реалізації онлайн-кінотеатру.

		<i>В.В. Барабаш</i>			<i>ІПЗ.КР.Б – 121 – 25 – ПЗ</i>	<i>Арк.</i>
		<i>О.Г. Чижмотря</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		20

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ПЛАТФОРМИ ОНЛАЙН-КІНОТЕАТРУ

2.1. Визначення варіантів використання та структури системи

Платформу онлайн-кінотеатру реалізовано як багаторівневу систему з поділом функціональності залежно від ролі користувача. Основу застосунку становлять модулі, які відповідають за перегляд медіакаталогу, керування списками перегляду, залишення рецензій, фільтрацію контенту, а також адміністрування ресурсу. Вся інформація зберігається в структурованій базі даних, що забезпечує узгодженість та доступність даних незалежно від стану користувача в системі.

Гість – це відвідувач, який не має облікового запису або не увійшов у систему. Йому доступний перегляд головної сторінки, ознайомлення з каталогом фільмів та серіалів, фільтрація за жанрами, перегляд детальної інформації про контент, перегляд рейтингу, а також можливість перейти до реєстрації чи авторизації.

Зареєстрований користувач після входу в систему отримує доступ до персонального кабінету, можливості додавати фільми до списку перегляду, залишати оцінки, писати коментарі, отримувати персоналізовані рекомендації та переглядати історію взаємодії з контентом. Це дозволяє створити індивідуальний досвід користування сервісом.

Адміністратор має розширені права доступу та виконує управління всіма складовими платформи: додавання і редагування фільмів, контроль за відгуками, модерація коментарів, керування обліковими записами користувачів, формування добірок, аналітика активності та контроль актуальності медіаінформації. Роль адміністратора критично важлива для підтримки якості та безперебійної роботи платформи.

На рис. 2.1 наведено діаграму варіантів використання платформи онлайн-кінотеатру. У ній відображено, які функціональні можливості доступні для кожного типу користувача: від гостя до адміністратора. Така візуалізація

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				21
Змн.	Арк.	№ докум.	Підпис	Дата		

дозволяє чітко зрозуміти взаємозв'язки між ролями та функціями, які вони виконують у системі.

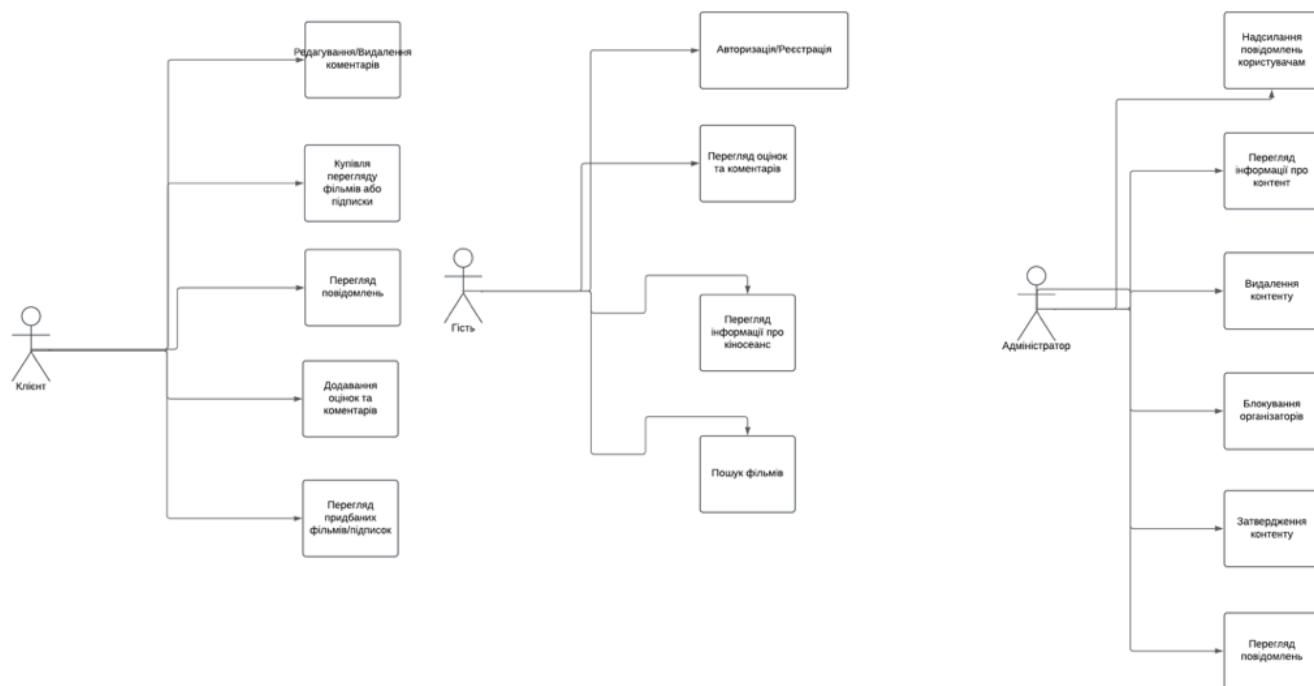


Рис. 2.1 – Діаграма варіантів використання онлайн-кінотеатру

Запропонована модель взаємодії користувачів із платформою онлайн-кінотеатру забезпечує чіткий розподіл доступу до функцій залежно від ролі, дозволяє гнучко керувати контентом та гарантує зручний користувацький досвід. Щоб система працювала стабільно та відповідала очікуванням аудиторії, було визначено набір функціональних і нефункціональних вимог, які лягли в основу розробки.

Функціональні вимоги охоплюють ключові можливості платформи. Авторизація користувачів передбачає реєстрацію, вхід і присвоєння ролей – гість, зареєстрований користувач або адміністратор. Гості можуть лише переглядати публічний каталог, знайомитись із фільмами та серіалами, користуватись пошуком і фільтрами. Після входу користувач отримує доступ до особистого кабінету, може купувати доступ до перегляду контенту, залишати оцінки, писати коментарі, формувати власні добірки. Адміністратор має інструменти для керування контентом, модерації користувацьких дій та аналізу активності на платформі.

Функціонал перегляду реалізовано через вбудований відеоплеєр із базовим набором можливостей: перемотування, пауза, регулювання гучності. Модуль купівлі дозволяє оформити замовлення на перегляд контенту з вибором способу оплати. Особистий кабінет користувача включає історію покупок, збережені фільми, базові налаштування профілю.

Адміністративна частина дає змогу додавати або редагувати фільми, видаляти застарілий або некоректний контент, переглядати скарги та статистику, а також керувати обліковими записами користувачів.

Окрім прикладної логіки, система повинна відповідати нефункціональним критеріям. Основними серед них є стабільність роботи, швидкість відгуку та зручність інтерфейсу. Платформа повинна коректно обробляти запити навіть при високому навантаженні, підтримувати щонайменше кілька сотень одночасних сесій, а сторінки мають відкриватись не довше ніж за 2 секунди. Інтерфейс повинен бути простим у використанні, без складних елементів або заплутаної навігації. Для технічного обслуговування допускається обмежений відсоток часу простою протягом місяця.

Загальна архітектура платформи побудована з урахуванням реальних потреб користувачів і дозволяє забезпечити стабільний доступ до відеоконтенту, підтримку персоналізованого досвіду та ефективне адміністрування всіх елементів системи.

Аналіз функціональних вимог дозволив виокремити ключові сутності, необхідні для реалізації бізнес-логіки платформи онлайн-кінотеатру. До них належать користувачі, фільми, жанри, коментарі, оцінки, списки перегляду, замовлення доступу, платіжні записи та повідомлення. На основі цих сутностей сформовано модель даних, реалізовану з використанням ORM-бібліотеки Mongoose, яка відображає структуру об'єктів, їх атрибути та зв'язки, необхідні для коректної роботи системи.

2.2. Розробка бази даних

Ефективне функціонування онлайн-кінотеатру неможливе без продуманої бази даних, яка має відповідати сучасним вимогам до стабільності, швидкодії,

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				23
Змн.	Арк.	№ докум.	Підпис	Дата		

масштабованості та зручності супроводу. У межах цього проєкту було прийнято рішення використовувати документоорієнтовану базу даних MongoDB, яка добре підходить для динамічних застосунків зі складною структурою документів. Для моделювання даних обрано ORM-бібліотеку Mongoose, яка надає зручний інструментарій для створення схем, валідації полів, визначення зв'язків між сутностями, а також оптимізації запитів і індексації.

У процесі проєктування було виокремлено низку ключових моделей, які становлять основу логіки системи. Сутність User зберігає облікові дані користувачів, такі як ім'я, електронна пошта, пароль, роль у системі, статус активності та аватар. Крім цього, модель містить інформацію про активність користувача: історію переглядів, перелік придбаних фільмів, обране та дату останнього входу. Це дозволяє реалізувати індивідуальний підхід до кожного користувача, впровадити обмеження доступу відповідно до ролі та аналізувати поведінку для подальшої персоналізації сервісу.

Завдяки гнучкості структури MongoDB, база даних підтримує як прості операції з користувацькими обліками, так і складні зв'язки між моделями – наприклад, між фільмами, коментарями, категоріями, жанрами та платежами. Таким чином, база забезпечує цілісність інформації, легкість розширення системи новими функціями та ефективне обслуговування зростаючої кількості запитів зі сторони клієнтів.

Сутність Movie є центральною у структурі бази даних, оскільки вона представляє основну одиницю контенту – фільм або серіал. У її складі зберігається повний набір характеристик: назва, детальний опис, постер, посилання на трейлер та повне відео, тривалість, рік випуску, країна походження, мова, рейтинг за віковою категорією, а також вартість перегляду. Передбачено також зберігання інформації про режисера, акторський склад і тип контенту (наприклад, «фільм» або «серіал»). Вартість контенту реалізується через вкладений об'єкт pricing, що дозволяє задавати як ціну для покупки, так і позначку про безкоштовний доступ.

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				24
Змн.	Арк.	№ докум.	Підпис	Дата		

Модель підтримує можливість прив'язки до кількох жанрів і категорій, що реалізовано за допомогою масивів з посиланнями на відповідні колекції Genre та Category. Це дає змогу фільму належати до кількох тематичних груп одночасно, що сприяє кращому структурованому пошуку для користувачів. Система також дозволяє сортувати та фільтрувати контент за різними критеріями, включаючи рік випуску, країну, тип, наявність безкоштовного доступу тощо.

Для організації й класифікації фільмів використовуються окремі сутності Genre та Category. Genre дозволяє згрупувати контент за художніми ознаками, а Category – за призначенням, типом платформи або маркетинговими критеріями. Обидві моделі мають індекси за назвами та активністю, що забезпечує швидке виконання пошукових і фільтраційних запитів.

Функціональність взаємодії між користувачами реалізована через модель Comment. Вона дає змогу залишати відгуки та коментарі до переглянутих фільмів. Кожен коментар містить текстове повідомлення, дату створення, а також прив'язку до конкретного фільму та автора. Це дозволяє формувати соціальний контекст довкола контенту, а також реалізувати сортування за часом або іншими критеріями. Такий підхід сприяє підвищенню залученості користувачів і формуванню довіри до платформи.

Окрему увагу в рамках проєктування було зосереджено на реалізації платіжної логіки, яка є критично важливою для функціонування онлайн-кінотеатру з платним доступом до контенту. Моделі Payment та Purchase відповідають за збереження повної історії фінансових транзакцій, фіксуючи суму платежу, валюту, статус операції, спосіб оплати, а також унікальні ідентифікатори транзакцій. Також передбачено поля, що відображають, чи було надано доступ до відео після оплати та коли саме закінчується термін доступу.

У доповнення до фінансових даних, у цих моделях зберігається технічна інформація про сеанс користувача – зокрема IP-адреса, user-agent та інформація про пристрій. Це дозволяє не лише проводити базову аналітику транзакцій, а й виявляти потенційно підозрілі або шахрайські дії, наприклад, багаторазові спроби оплати з одного пристрою або географічно аномальні запити.

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				25
Змн.	Арк.	№ докум.	Підпис	Дата		

Завдяки активному використанню віртуальних полів (virtual fields) у Mongoose забезпечено гнучке зв'язування документів між собою без дублювання даних. Це дозволяє ефективно формувати агреговані запити – наприклад, отримання історії покупок користувача з повною інформацією про відео – без додаткових витрат на продуктивність. Індексація ключових полів, таких як `userId`, `movieId`, `transactionId`, `status`, додатково прискорює операції фільтрації, сортування та статистичного аналізу даних.

На рисунку 2.2 представлено логічну структуру бази даних, яка відображає основні сутності системи, їхні атрибути та типи зв'язків між ними. Така організація дозволяє створити стабільну, масштабовану архітектуру з підтримкою майбутнього розширення – як за рахунок нових функцій (наприклад, підписок або промокодів), так і за рахунок збільшення навантаження зі сторони користувачів.

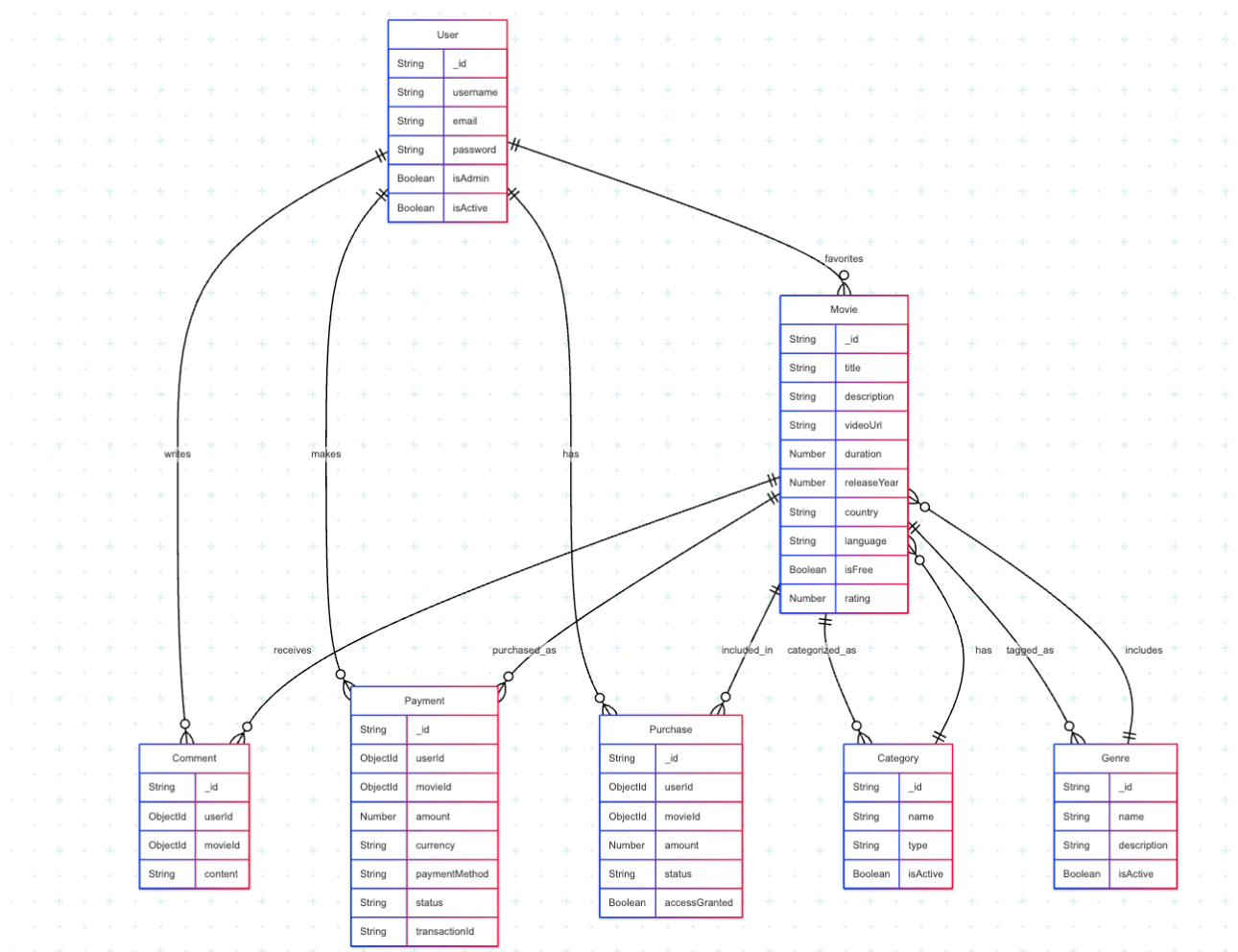


Рис. 2.2 Структура бази даних

Після побудови загальної логічної структури бази даних (рис. 2.2) доцільно розглянути детальні характеристики кожної моделі, що використовуються в системі. У цьому підрозділі подано табличний опис основних сутностей, де для кожної моделі наведено перелік полів, типи даних, вказано наявність первинних (PK) та зовнішніх (FK) ключів, а також коротко охарактеризовано функціональне призначення кожного атрибуту. Такий формат дозволяє чітко структурувати інформацію та забезпечити зручність подальшої реалізації й супроводу системи.

Таблиця 2.1

Структура сутності "User"

Назва поля	Тип даних	PK	FK	Опис
_id	ObjectId	+	-	Унікальний ідентифікатор користувача
username	String	-	-	Унікальне ім'я користувача
email	String	-	-	Адреса електронної пошти
password	String	-	-	Хешований пароль
firstName	String	-	-	Ім'я користувача
lastName	String	-	-	Прізвище користувача
profilePicture	String	-	-	Посилання на аватар
isAdmin	Boolean	-	-	Ознака, чи є користувач адміністратором
isActive	Boolean	-	-	Активність облікового запису
passwordResetToken	String	-	-	Токен для відновлення пароля
passwordResetEndTime	Date	-	-	Термін дії токена
lastLogin	Date	-	-	Дата останнього входу
purchasedMovies	Array	-	+	Масив фільмів, які придбав користувач

Продовження таблиці 2.1

favoriteMovies	Array	-	+	Список фільмів, доданих у обране
createdAt	Date	-	-	Дата створення запису
updatedAt	Date	-	-	Дата останнього оновлення запису

Таблиця 2.2

Структура сутності " Movie"

Назва поля	Тип даних	PK	FK	Опис
_id	ObjectId	+	-	Унікальний ідентифікатор фільму
title	String	-	-	Назва фільму
description	String	-	-	Опис фільму
posterImage	String	-	-	Посилання на зображення постера
trailerUrl	String	-	-	Посилання на трейлер фільму
videoUrl	String	-	-	Посилання на відеофайл фільму
duration	Number	-	-	Тривалість фільму у хвилинах
releaseYear	Number	-	-	Рік випуску
country	String	-	-	Країна виробництва
film_language	String	-	-	Мова озвучення або субтитрів
ageRating	Enum (G/PG/PG-13/...)	-	-	Віковий рейтинг
genres	Array	-	+	Жанри фільму

Продовження таблиці 2.2

director	String	-	-	Режисер
cast	Array	-	-	Основні актори
type	Enum (movie)	-	-	Тип медіа
pricing.buyPrice	Number	-	-	Ціна за покупку
pricing.isFree	Boolean	-	-	Ознака безкоштовного доступу
categories	Array	-	+	Категорії фільму
views	Number	-	-	Кількість переглядів
likes	Number	-	-	Кількість вподобань
rating	Number (0–10)	-	-	Середній рейтинг фільму
createdAt	Date	-	-	Дата створення запису
updatedAt	Date	-	-	Дата останнього оновлення запису

Таблиця 2.3

Структура сутності "Category"

Назва поля	Тип даних	PK	FK	Опис
_id	ObjectId	+	-	Унікальний ідентифікатор категорії
name	String	-	-	Назва категорії
description	String	-	-	Опис категорії
type	String	-	-	Тип категорії
isActive	Boolean	-	-	Ознака активності категорії
createdAt	Date	-	-	Дата створення запису
updatedAt	Date	-	-	Дата останнього оновлення запису

Таблиця 2.4

Структура сутності "Genre"

Назва поля	Тип даних	PK	FK	Опис
_id	ObjectId	+	-	Унікальний ідентифікатор жанру
name	String	-	-	Назва жанру (наприклад, драма, комедія, трилер)
description	String	-	-	Текстовий опис жанру
content	Array	-	-	Додатковий контент або мета-дані, пов'язані з жанром
isActive	Boolean	-	-	Ознака активності жанру, використовується для фільтрації
createdAt	Date	-	-	Дата створення запису
updatedAt	Date	-	-	Дата останнього оновлення запису

Таблиця 2.5

Структура сутності "Comment"

Назва поля	Тип даних	PK	FK	Опис
_id	ObjectId	+	-	Унікальний ідентифікатор коментаря
userId	ObjectId	-	+	Посилання на користувача, який залишив коментар
movieId	ObjectId	-	+	Посилання на фільм, до якого належить коментар
content	String	-	-	Текст повідомлення користувача
createdAt	Date	-	-	Дата та час створення коментаря
updatedAt	Date	-	-	Дата останнього редагування (якщо було)

Структура сутності "Payment"

Назва поля	Тип даних	PK	FK	Опис
_id	ObjectId	+	-	Унікальний ідентифікатор платежу
userId	ObjectId	-	+	Посилання на користувача, який здійснив платіж
movieId	ObjectId	-	+	Посилання на фільм, за який проведено оплату
amount	Number	-	-	Сума транзакції
currency	String	-	-	Валюта оплати
paymentMethod	String	-	-	Метод оплати
transactionId	String	-	-	Унікальний ідентифікатор транзакції
status	String	-	-	Статус оплати
paymentProvider	String	-	-	Провайдер оплати
providerTransactionId	String	-	-	Ідентифікатор транзакції від провайдера
accessGranted	Boolean	-	-	Ознака надання доступу до фільму
accessExpiryDate	Date	-	-	Дата закінчення доступу
refundReason	String	-	-	Причина повернення коштів
refundDate	Date	-	-	Дата повернення коштів
metadata.userAgent	String	-	-	Інформація про браузер/пристрій користувача

Продовження таблиці 2.6

metadata.ip Address	String	-	-	IP-адреса користувача
metadata.deviceInfo	String	-	-	Технічна інформація про пристрій
createdAt	Date	-	-	Дата створення запису
updatedAt	Date	-	-	Дата останнього оновлення

Таблиця 2.7

Структура сутності "Purchase"

Назва поля	Тип даних	PK	FK	Опис
_id	ObjectId	+	-	Унікальний ідентифікатор покупки
userId	ObjectId	-	+	Посилання на користувача, який здійснив покупку
movieId	ObjectId	-	+	Посилання на фільм, який було придбано
amount	Number	-	-	Сума покупки
Currency	String	-	-	Валюта покупки (за замовчуванням – UAH)
paymentMethod	String	-	-	Метод оплати
transactionId	String	-	-	Унікальний ідентифікатор транзакції
status	String	-	-	Статус покупки
accessGranted	Boolean	-	-	Ознака того, чи було надано доступ до перегляду
createdAt	Date	-	-	Дата створення запису
updatedAt	Date	-	-	Дата останнього оновлення

Таким чином, спроектована база даних онлайн-кінотеатру охоплює всі основні аспекти функціонування системи – від управління користувачами, фільмами та категоріями до обробки оплат, перегляду придбаного контенту, збереження вподобань і взаємодії через коментарі. Кожна модель виконує чітко визначену роль у бізнес-логіці, що дозволяє підтримувати структурованість і узгодженість даних. Завдяки використанню ORM-бібліотеки Mongoose досягнуто зручного опису схем, реалізації зв'язків між документами та оптимізації запитів за допомогою індексів і віртуальних полів. Обрана архітектура забезпечує масштабованість, гнучкість і ефективність роботи системи навіть при зростанні кількості користувачів та обсягів контенту. Це створює надійну основу для подальшого розширення функціоналу та інтеграції нових можливостей у межах онлайн-платформи.

2.3. Проектування та реалізація алгоритмів роботи

Архітектура системи онлайн-кінотеатру побудована за клієнт-серверною моделлю з розподіленням відповідальностей між фронтендом та бекендом. Такий підхід дозволяє досягти високої гнучкості, масштабованості та розділення логіки, що полегшує супровід і подальший розвиток системи.

Серверна частина реалізована за допомогою фреймворку Express.js, що забезпечує створення RESTful API для взаємодії з клієнтським інтерфейсом. Усі запити обробляються через чітко визначені маршрути, які відповідають за функціональність, пов'язану з автентифікацією, керуванням фільмами, коментарями, оплатами та користувачами. Для роботи з базою даних використовується ORM-бібліотека Mongoose, яка дозволяє гнучко описувати схеми, створювати зв'язки між моделями та формувати складні запити до MongoDB.

Зберігання медіаконтенту реалізовано через інтеграцію з Cloudinary, що дозволяє не лише надійно зберігати файли, а й оптимізувати їх для швидкого завантаження на клієнтській стороні. Завантаження виконується за допомогою бібліотек multer та multer-storage-cloudinary.

		<i>В.В. Барабаш</i>			<i>ІПЗ.КР.Б – 121 – 25 – ПЗ</i>	<i>Арк.</i>
		<i>О.Г. Чижмотря</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		<i>33</i>

Для обробки оплат інтегровано Stripe API. Після підтвердження транзакції на фронтенді, відповідний запит надсилається на сервер, де перевіряється токен платежу, зберігаються деталі транзакції в базі даних і надається доступ до перегляду контенту відповідному користувачеві. Усі операції проходять через захищені ендпоінти з перевіркою JWT-токена, що генерується через бібліотеку jsonwebtoken.

Користувацький інтерфейс реалізовано за допомогою бібліотеки React у поєднанні з React Router для маршрутизації. Візуальні компоненти створені на основі Material-UI, що дозволяє швидко створювати адаптивний і сучасний дизайн. Комунікація з бекендом здійснюється через HTTP-запити бібліотеки axios.

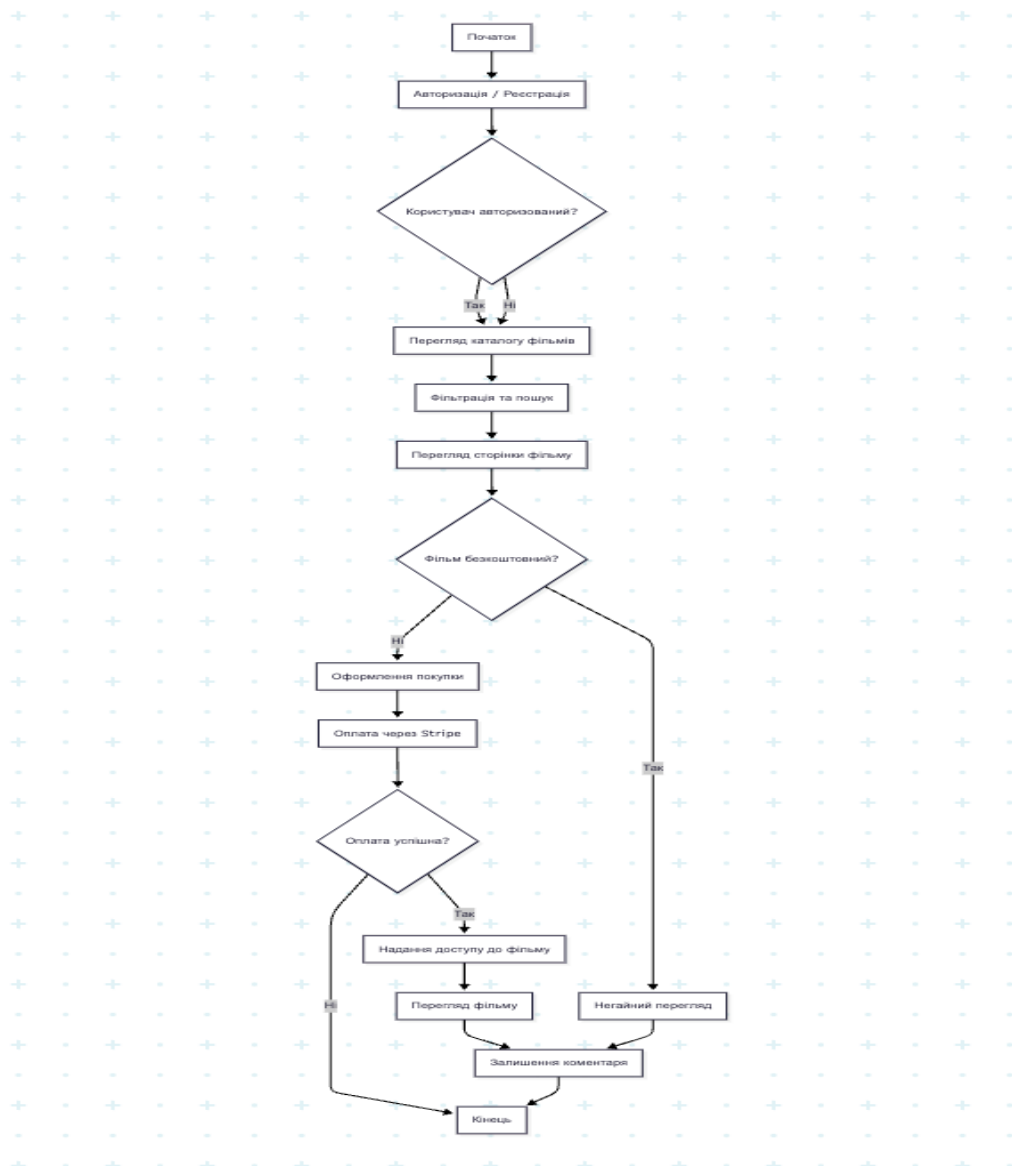


Рис. 2.3 – Блок-схема загального алгоритму

Розглянемо ключові сценарії взаємодії користувача. Гість має можливість переглядати каталог фільмів, здійснювати сортування та фільтрацію за категоріями, жанрами, роком випуску, країною та іншими параметрами. Авторизований користувач може додавати фільми до обраного, переглядати історію покупок, отримувати доступ до придбаного контенту та залишати коментарі. Адміністратор здійснює керування фільмами, жанрами та категоріями, модерує коментарі, а також переглядає звіти щодо оплат і користувацької активності.

На рисунку 2.3 подано загальний алгоритм функціонування системи: від входу користувача та завантаження фільмів до здійснення оплати й перегляду відео. Завдяки розділенню ролей, кожен тип користувача взаємодіє лише з дозволеним йому функціоналом, що підвищує безпеку та контроль над платформою.

Для кращого розуміння логіки взаємодії користувача з платформою в процесі придбання фільму було створено діаграму активності, яка ілюструє ключові етапи сценарію купівлі та взаємодії з платіжною системою (рис 2.4). У ній розмежовано ролі користувача та зовнішнього платіжного сервісу, що дозволяє наочно відобразити послідовність подій, перевірок і рішень.

Сценарій розпочинається з перегляду сторінки конкретного фільму, після чого користувач може авторизуватись або зареєструватись. У випадку вже існуючого акаунта – система дозволяє перейти безпосередньо до оплати. Наступним кроком є вибір методу оплати, введення платіжних даних і передача запиту до зовнішньої платіжної системи. Після обробки транзакції система отримує відповідь – підтвердження успішної оплати або відмову. У першому випадку користувачу надається доступ до перегляду фільму, у другому – платформа дозволяє повторити спробу або змінити спосіб оплати.

Завдяки такій структурі процесу забезпечується безперервна й безпечна взаємодія між користувачем і системою, з урахуванням потенційних помилок, необхідності підтвердження транзакцій та подальшого контролю доступу. Ця діаграма стала основою для реалізації клієнтського та серверного алгоритмів, а

		<i>В.В. Барабаш</i>			<i>ІПЗ.КР.Б – 121 – 25 – ПЗ</i>	<i>Арк.</i>
		<i>О.Г. Чижмотря</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		<i>35</i>

також сприяла побудові захищеного, зручного й передбачуваного процесу купівлі цифрового контенту.

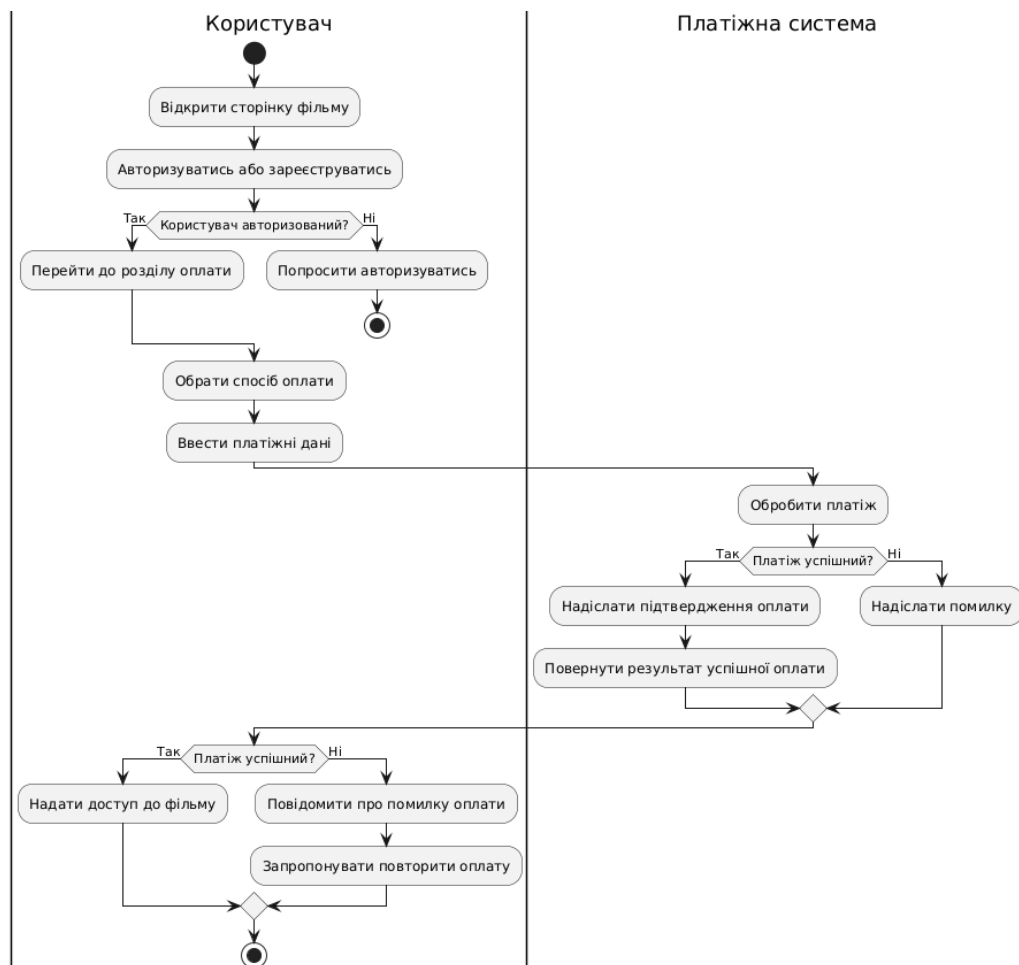


Рис. 2.4 – Діаграма активності: процес купівлі фільму

2.4. Реалізація функціоналу

Реалізація функціоналу онлайн-кінотеатру базується на клієнт-серверній архітектурі, де фронтенд створено за допомогою React, а бекенд – на Express з використанням MongoDB. Такий підхід забезпечує ефективне розділення відповідальностей, що сприяє масштабованості та підтримованості проєкту. Для обробки даних використано REST API, а взаємодія з базою даних реалізована через ORM-бібліотеку Mongoose, яка дозволяє описувати моделі, зв'язки й валідацію. Додатково впроваджено механізми авторизації, завантаження мультимедійних файлів, оплат і перегляду історії. Подальші розділи розкривають реалізацію основних модулів системи з поясненням логіки та прикладами коду.

Наведений фрагмент коду реалізує підключення та відключення від бази даних MongoDB у проєкті онлайн-кінотеатру за допомогою бібліотеки mongoose. У функції connectDb спочатку перевіряється наявність та коректність змінної оточення MONGOURL – у разі її відсутності або якщо вона вказує на локальний сервер, генерується помилка. Якщо адреса валідна, виконується асинхронне підключення до MongoDB із використанням сучасних параметрів для забезпечення стабільності та сумісності. У разі успіху виводиться повідомлення про успішне з'єднання, а при помилці – фіксується помилка в консолі й викидається виняток. Додатково реалізовано функцію mongooseDisconnect, яка забезпечує коректне закриття з'єднання з базою даних із відповідним логуванням, що є важливим під час завершення роботи сервера або в процесах тестування.

```
const connectDb = async () => {
  try {
    if (
      !process.env.MONGOURL ||
      process.env.MONGOURL === "mongodb://localhost:27017/bestflix"
    ) {
      throw new Error("MongoDB URL не налаштований або використовує localhost");
    }
    await mongoose.connect(process.env.MONGOURL, {
      useNewUrlParser: true,
      useUnifiedTopology: true,
    });
    console.log("MongoDB підключено успішно");
  } catch (error) {
    console.error("MongoDB connection error:", error.message);
    throw error;
  }
};

const mongooseDisconnect = async () => {
  try {
    await mongoose.disconnect();
    console.log("MongoDB відключено");
  } catch (error) {
    console.error("Помилка відключення MongoDB:", error);
  }
};
```

Повний код запуску сервера наведено в Додатку А, а нижче розглянуто лише логіку функції startServer(), що відповідає за ініціалізацію основного

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				37
Змн.	Арк.	№ докум.	Підпис	Дата		

процесу. Функція `startServer()` є асинхронною та виконує два основні завдання: підключення до бази даних і запуск веб-сервера. У блоці `try` спочатку викликається функція `connectDb()`, яка встановлює з'єднання з MongoDB. Якщо з'єднання успішне, сервер запускається на вказаному порту, і в консоль виводиться інформація про успішний запуск, а також доступ до API та health-check ендпоінту. У разі помилки підключення до бази спрацьовує блок `catch`, і сервер все одно запускається, але вже без доступу до бази даних. Такий підхід дозволяє не блокувати роботу фронтенду або інших сервісів, навіть якщо тимчасово недоступна БД. Це підвищує гнучкість і стійкість архітектури до відмов.

```
async function startServer() {
  try {
    await connectDb();
    app.listen(port, () => {
      console.log(Server is running on port ${port});
      console.log(API доступне на: http://localhost:${port});
      console.log(Health check: http://localhost:${port}/health);
    });
  } catch (error) {
    console.error("Помилка запуску сервера:", error);
    console.log("Запуск сервера без підключення до БД...");
    app.listen(port, () => {
      console.log(Server is running on port ${port} (без БД));
      console.log(API доступне на: http://localhost:${port});
    });
  }
}
```

Фрагмент коду, наведений нижче, реалізує `middleware`-функцію `verifyToken`, яка відповідає за перевірку автентичності користувача на основі JWT-токена.

Функція спочатку зчитує заголовок `token` з об'єкта `req.headers`. Якщо заголовок присутній, з нього вилучається сам токен шляхом поділу рядка за пробілом. Далі виконується перевірка токена за допомогою методу `jwt.verify()`, де вказується секретний ключ, збережений у змінній середовища `SECRET_KEY_FOR_CRYPTOJS`. У разі помилки перевірки повертається відповідь з кодом 403 і повідомленням про недійсний токен. Якщо перевірка успішна, інформація про користувача записується в `req.user`, після чого

		<i>В.В. Барабаш</i>			<i>ІПЗ.КР.Б – 121 – 25 – ПЗ</i>	<i>Арк.</i>
		<i>О.Г. Чижмоторя</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		38

викликається next() для передачі керування до наступного обробника. Якщо ж токен не надано, повертається помилка 401 Unauthorized. Такий механізм дозволяє ефективно захищати приватні маршрути в системі.

```
const jwt = require("jsonwebtoken");
const verifyToken = async (req, res, next) => {
  const authHeader = req.headers.token;
  if (authHeader) {
    const token = authHeader.split(" ")[1];
    await jwt.verify(
      token,
      process.env.SECRET_KEY_FOR_CRYPTOJS,
      (err, user) => {
        if (err) {
          return res.status(403).json("Token is not valid!");
        }
        req.user = user;
        next();
      }
    );
  } else {
    return res.status(401).json("You are not authenticated!");
  }
};
module.exports = verifyToken;
```

У проєкті реалізовано повноцінну інтеграцію з хмарним сервісом Cloudinary для зберігання зображень і відео, що додаються до фільмів. За допомогою бібліотек cloudinary, multer та multer-storage-cloudinary налаштовано два окремих сховища – для зображень та відеофайлів. Зображення автоматично трансформуються (обрізка, оптимізація якості, формат), що дозволяє забезпечити швидке завантаження сторінок та адаптацію до різних пристроїв. Для відео визначено окреме сховище з підтримкою популярних форматів, таких як mp4, webm та інші. Універсальний middleware uploadFields дозволяє одночасно завантажувати кілька типів медіа (постери, трейлери, відео) до відповідних директорій. Крім завантаження, реалізовано функцію видалення файлів із Cloudinary за їхнім publicId. Також передбачена утиліта для отримання publicId із URL, що полегшує управління контентом. Повна реалізація наведена в додатку А.

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				39
Змн.	Арк.	№ докум.	Підпис	Дата		

Контролер фільмів відіграє ключову роль у функціонуванні API, забезпечуючи повний набір операцій для управління об'єктами типу фільм. Він реалізований у вигляді набору асинхронних функцій, кожна з яких виконує окреме завдання – створення через функцію `createMovie`, оновлення через `updateMovie`, видалення через `deleteMovie`, отримання одного фільму за допомогою `getMovie` або списку фільмів через `getAllMovies`, а також пошук за ключовими словами у `searchMovies` і фільтрація за жанром або категорією у `getMoviesByGenre` та `getMoviesByCategory`. Особливе значення має функція `getMoviesStats`, яка формує зведену статистику: кількість фільмів, розподіл за жанрами, категоріями та роками. Усі функції мають єдину структуру – `const methodName = async req, res => ...`, що сприяє уніфікації коду та полегшує супровід. Логіка методів супроводжується валідацією вхідних даних, логуванням і обробкою помилок. Повна реалізація контролера наведена в додатку А. Також слід зазначити, що всі інші контролери в проєкті – зокрема для жанрів, категорій, користувачів та інших сутностей – написані у подібному стилі, з дотриманням принципів REST, розподілом відповідальності та чистим підтримуваним кодом.

```
const createMovie = async req, res => {};
const updateMovie = async req, res => {};
const deleteMovie = async req, res => {};
const getMovie = async req, res => {};
const getAllMovies = async req, res => {};
const searchMovies = async req, res => {};
const getMoviesByGenre = async req, res => {};
const getMoviesByCategory = async req, res => {};
const getMoviesStats = async req, res => {};
```

У проєкті реалізовано окремий роутер для керування фільмами на базі Express. Він включає як публічні, так і захищені маршрути з перевіркою ролей і підтримкою завантаження файлів. Роутери для інших сутностей реалізовані за аналогічним підходом.

```
const express = require('express');
const router = express.Router();
const verifyToken = require('../verifyToken');
const adminAuth = require('../middleware/adminAuth');
const { uploadFields } = require('../config/cloudinary');
```

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				40
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const {
  createMovie,
  updateMovie,
  deleteMovie,
  getMovie,
  getAllMovies,
  searchMovies,
  getMoviesByGenre,
  getMoviesByCategory,
  getMoviesStats
} = require('./movie.controller');
router.get('/', getAllMovies);
router.get('/search', searchMovies);
router.get('/stats', getMoviesStats);
router.get('/genre/:genre', getMoviesByGenre);
router.get('/category/:category', getMoviesByCategory);
router.get('/:id', getMovie);
router.post('/', adminAuth, uploadFields, createMovie);
router.put('/:id', adminAuth, uploadFields, updateMovie);
router.delete('/:id', adminAuth, deleteMovie);
module.exports = router;

```

Висновки до другого розділу

У другому розділі було розроблено повну архітектуру платформи онлайн-кінотеатру, починаючи від побудови структури бази даних до реалізації ключових серверних та клієнтських компонентів. Було визначено основні сутності системи, такі як фільми, жанри, категорії, коментарі, платежі й покупки, а також описано їхні зв'язки й особливості реалізації. Розглянуто загальну логіку роботи платформи, включаючи маршрутизацію, завантаження медіа на Cloudinary, перевірку токенів і обробку транзакцій. Значну увагу приділено створенню зручного та масштабованого API з використанням Express, Mongoose і Stripe. Для наочності надано діаграми архітектури, алгоритмів і послідовності дій під час купівлі фільму. Контролери реалізовано у вигляді окремих модулів із чітким поділом відповідальностей, що спрощує підтримку та розвиток проєкту. Таким чином, було створено гнучкий і ефективний серверний застосунок, здатний забезпечити надійну взаємодію з клієнтом і стабільну роботу платформи.

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				41
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУ

3.1. Порядок встановлення та налаштування системи

Діаграма розгортання, зображена на рисунку 3.1, наочно демонструє фізичну структуру компонентів онлайн-кінотеатру під час його розгортання та взаємозв'язки між ними. Вона відображає основні вузли, через які проходить обробка даних і взаємодія з користувачем: клієнтський інтерфейс, сервер додатку, база даних, зовнішня платіжна система, а також хмарне сховище для зображень і відео. Такий підхід дозволяє оцінити, як саме окремі частини системи взаємодіють між собою у фізичному середовищі, зокрема через HTTP, TCP/IP і API-виклики.

Клієнтська частина реалізована у вигляді браузера, який надсилає запити до серверного застосунку, зібраного на Express.js. Сервер обробляє запити до API, взаємодіє з базою даних MongoDB через Mongoose, а також опрацьовує медіафайли за допомогою Cloudinary. Інтеграція з платіжною системою здійснюється через окремий модуль, що надсилає запити до Stripe або подібного сервісу для ініціалізації та перевірки транзакцій. Після успішного платежу сервер надає користувачеві доступ до контенту – наприклад, можливість перегляду купленого фільму.

У процесі налаштування система потребує конфігурації середовищних змінних, розгортання backend-застосунку, підключення до бази даних і розміщення фронтенду. Завдяки чіткому поділу на модулі та стандартній структурі Node.js-застосунку, розгортання відбувається швидко та передбачувано. Вся архітектура є масштабованою, дозволяє легко додавати нові сервіси та підтримує горизонтальне розширення при збільшенні навантаження.

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				42
Змн.	Арк.	№ докум.	Підпис	Дата		

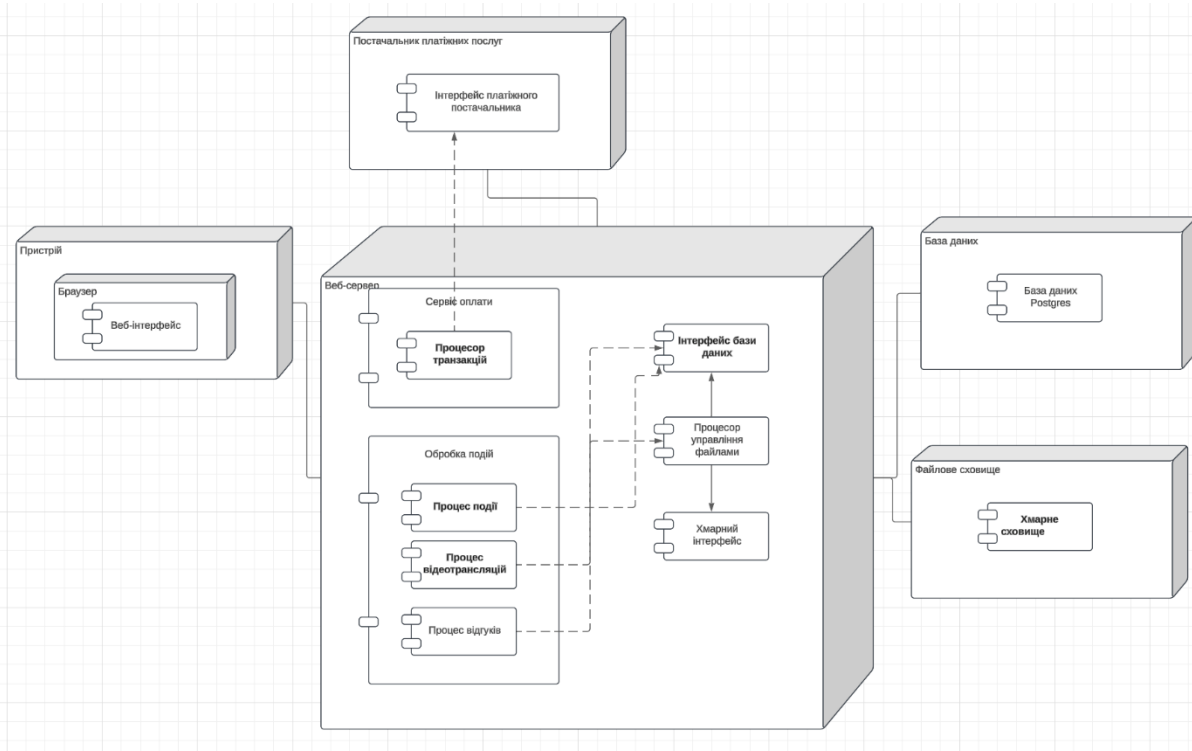


Рис. 3.1 – Діаграма розгортання онлайн-кінотеатру.

3.2 Структура інтерфейсу та порядок взаємодії з інтернет-магазином ювелірних виробів

Після відкриття вебзастосунку користувач потрапляє на головну сторінку (рис. 3.2), яка виконує функцію вітального екрана та фільмотеки. У верхній частині розміщено навігаційне меню з переходами до основних розділів: Головна, Фільми, Жанри та Категорії. Центральна частина інтерфейсу містить вітальний блок із назвою платформи, коротким описом, а також відображенням статистики – кількості фільмів, жанрів та категорій, що вже доступні на сайті.

Нижче розташована панель фільтрів, яка дозволяє здійснювати пошук за ключовими словами, а також обирати жанри та категорії з інтерактивних кнопок. Усі фільми виводяться у вигляді карток, які містять обкладинку, позначку типу (фільм або серіал), вік релізу, назву та короткий опис. Клік по картці веде на сторінку з детальною інформацією про вибраний фільм.

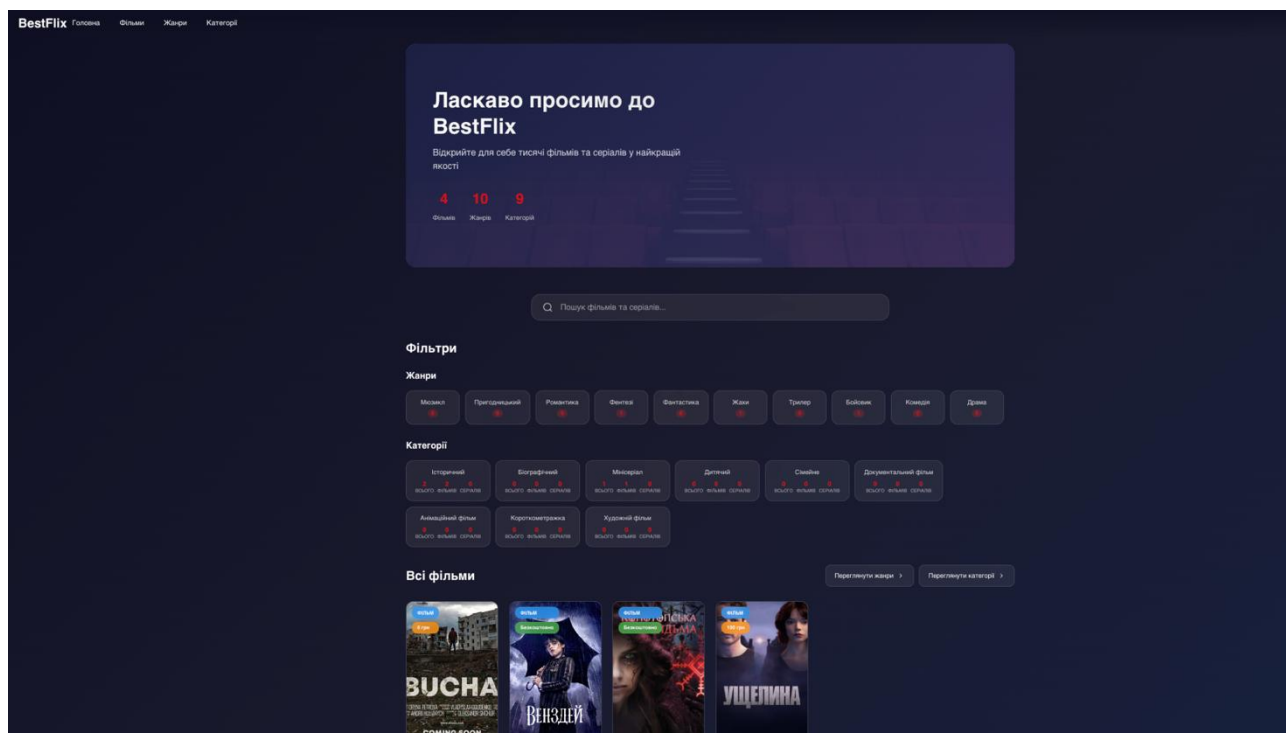


Рис. 3.2 – Головна сторінка

Для отримання повного доступу до можливостей платформи користувачу необхідно зареєструватися або увійти до свого акаунту (рис 3.3-3.4). При переході на сторінку реєстрації відкривається форма з поетапним введенням даних: спочатку вводиться ім'я, прізвище та унікальне ім'я користувача, після чого – адреса електронної пошти та пароль. Інтерфейс простий, інтуїтивно зрозумілий, із чіткою візуалізацією кроків процесу. Сторінка входу реалізована окремо – користувач вводить email і пароль, може обрати опцію "Запам'ятати мене" або скористатися функцією відновлення паролю. Після входу відкривається доступ до перегляду фільмів, додавання у вподобане, створення коментарів, а також доступу до особистого профілю.

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижморя				44
Змн.	Арк.	№ докум.	Підпис	Дата		

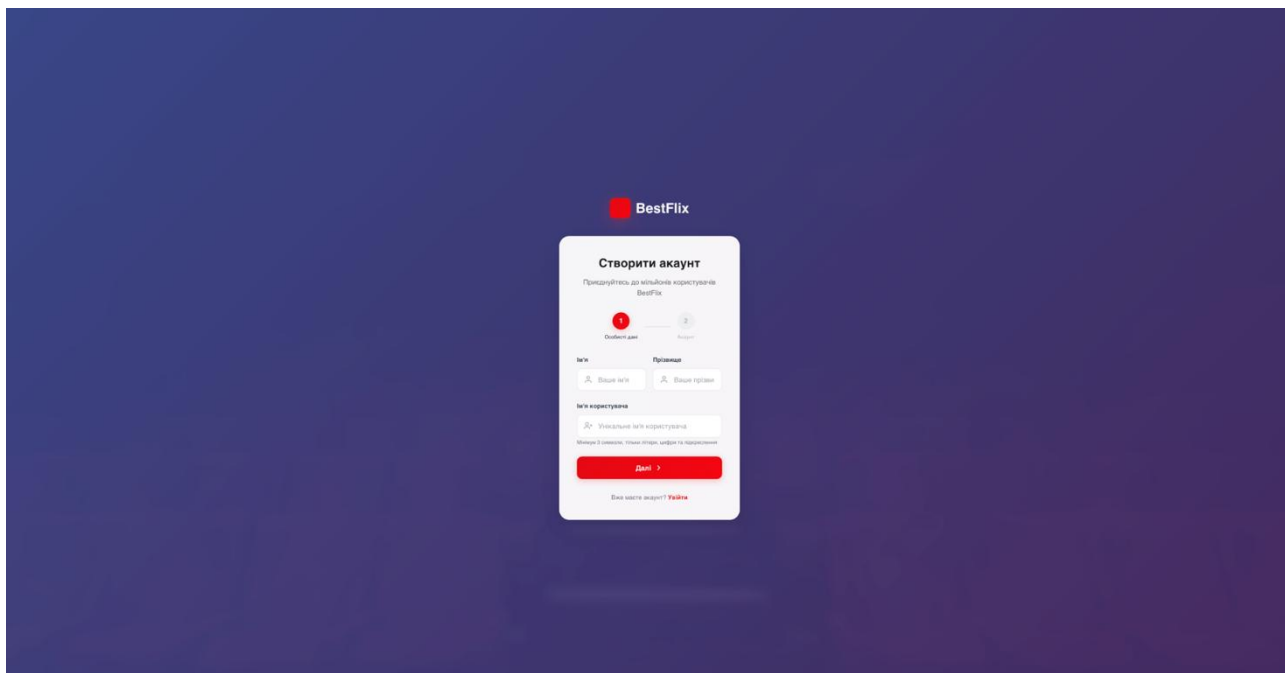


Рис. 3.3 – Сторінка реєстрації користувача

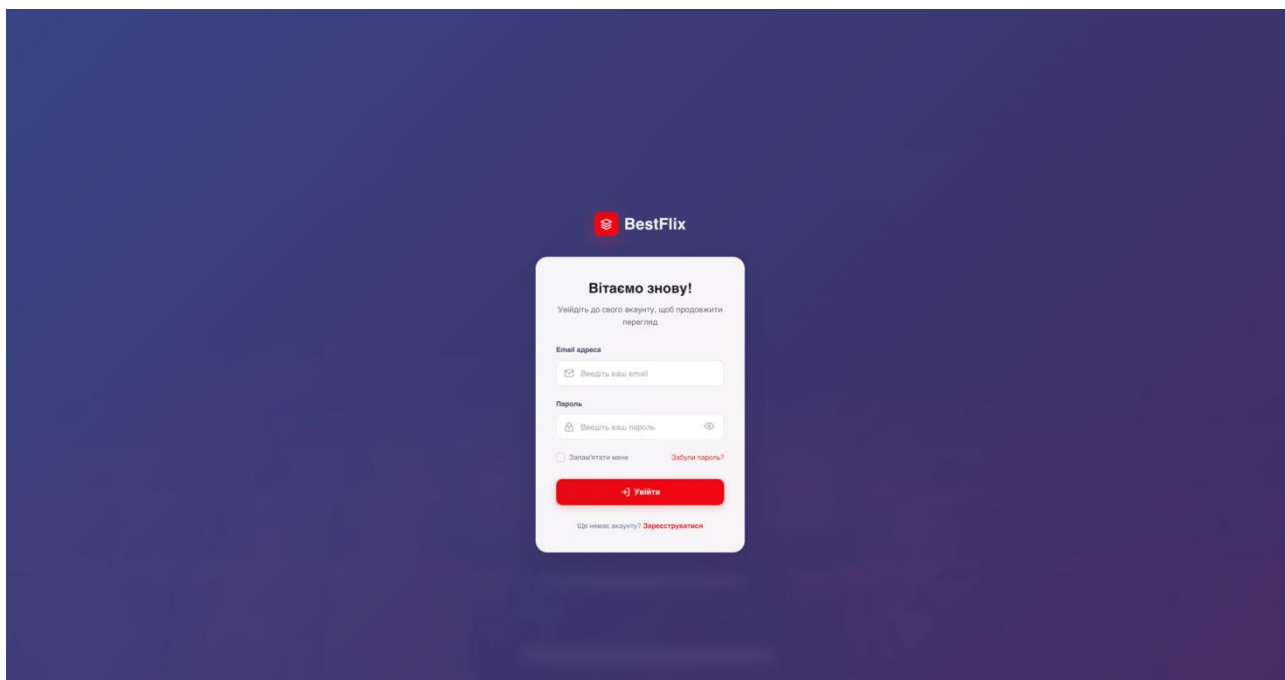


Рис. 3.4 – Сторінка входу до акаунту

Після входу до онлайн-кінотеатру користувач отримує доступ до свого особистого кабінету, де може переглянути основну інформацію про акаунт, статистику витрат, кількість придбаних фільмів і збережених вподобань. Інтерфейс профілю дозволяє редагувати персональні дані, а також відслідковувати останню активність (рис 3.5). Такий підхід забезпечує зручність

		V.B. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				45
Змн.	Арк.	№ докум.	Підпис	Дата		

керування обліковим записом і підвищує рівень персоналізації взаємодії з платформою.

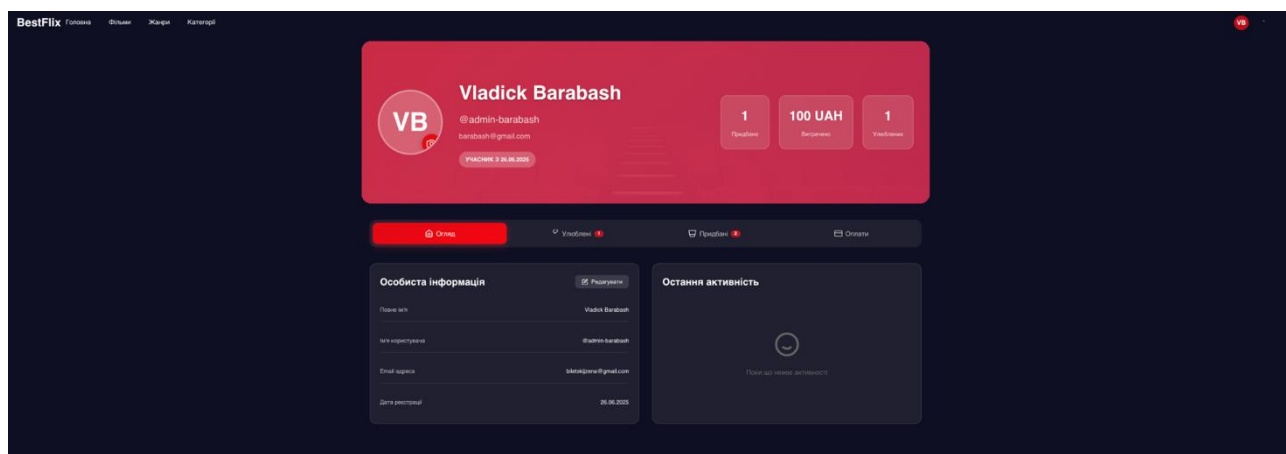


Рис. 3.5 – Особистий кабінет користувача

Щоб переглянути фільм, користувач має перейти на його детальну сторінку. Тут відображається постер, назва, опис, рік випуску, тривалість, жанр, вікове обмеження та додаткова інформація про режисера, акторський склад, країну виробництва й категорію. Також доступна кнопка для запуску відео, можливість додати фільм до улюблених і переглянути статистику переглядів, лайків і рейтингу. Крім того, користувач може ознайомитись з коментарями до фільму. Інтерфейс детальної сторінки фільму зображено на рис. 3.6.

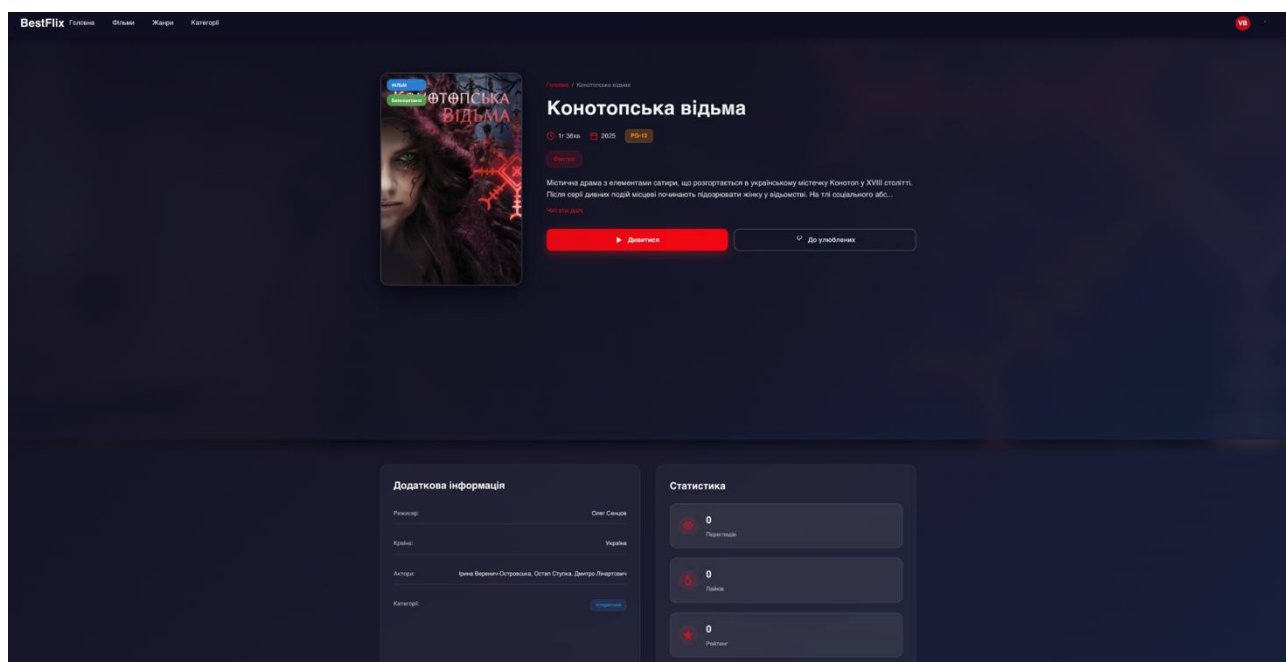


Рис. 3.6 – Детальна сторінка фільму

		<i>В.В. Барабаш</i>			<i>ІПЗ.КР.Б – 121 – 25 – ПЗ</i>	<i>Арк.</i>
		<i>О.Г. Чижмотря</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		<i>46</i>

Після натискання на кнопку «Дивитися» користувачу відкривається вбудований відеоплеєр, у якому починається відтворення фільму. Інтерфейс плеєра дозволяє користувачу керувати процесом перегляду: зупиняти або продовжувати відтворення, перемикатися на повноекранний режим, регулювати гучність тощо (рис. 3.7).

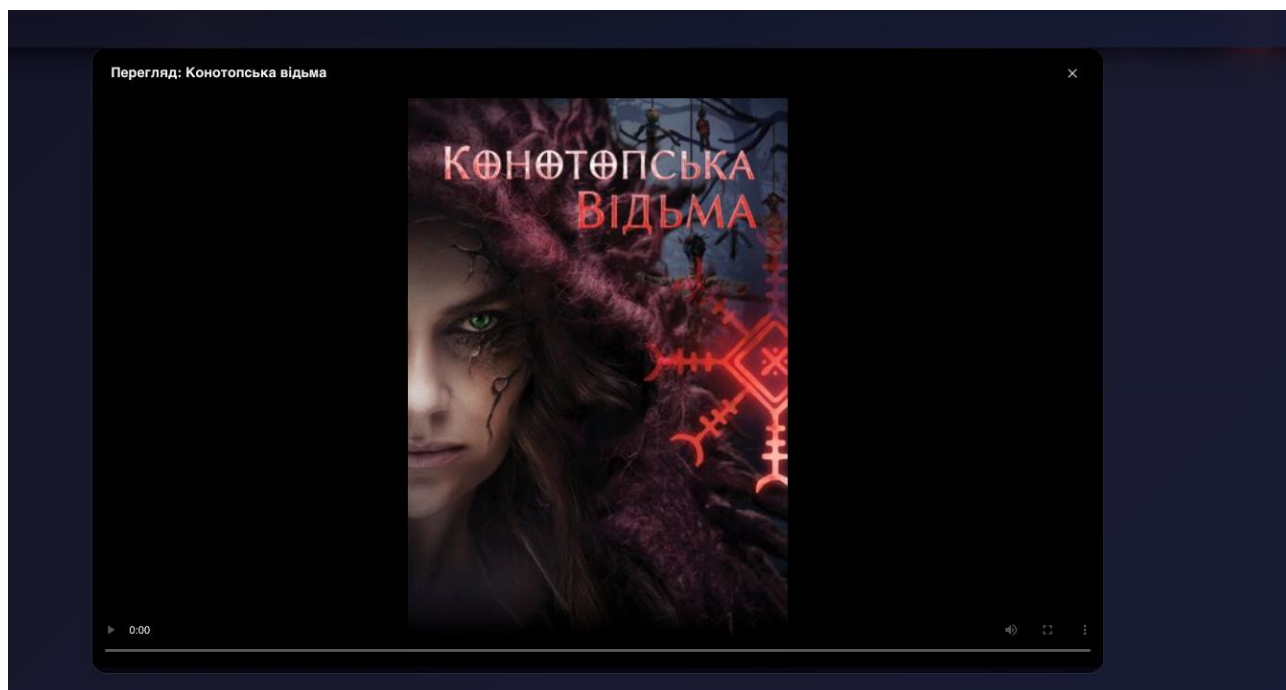


Рис. 3.7 – Відтворення фільму у вбудованому відеоплеєрі

У разі, якщо фільм є платним, користувач повинен його придбати перед переглядом. Для цього необхідно натиснути на кнопку з зазначеною вартістю, яка відкриє відповідний процес оплати (рис. 3.8).

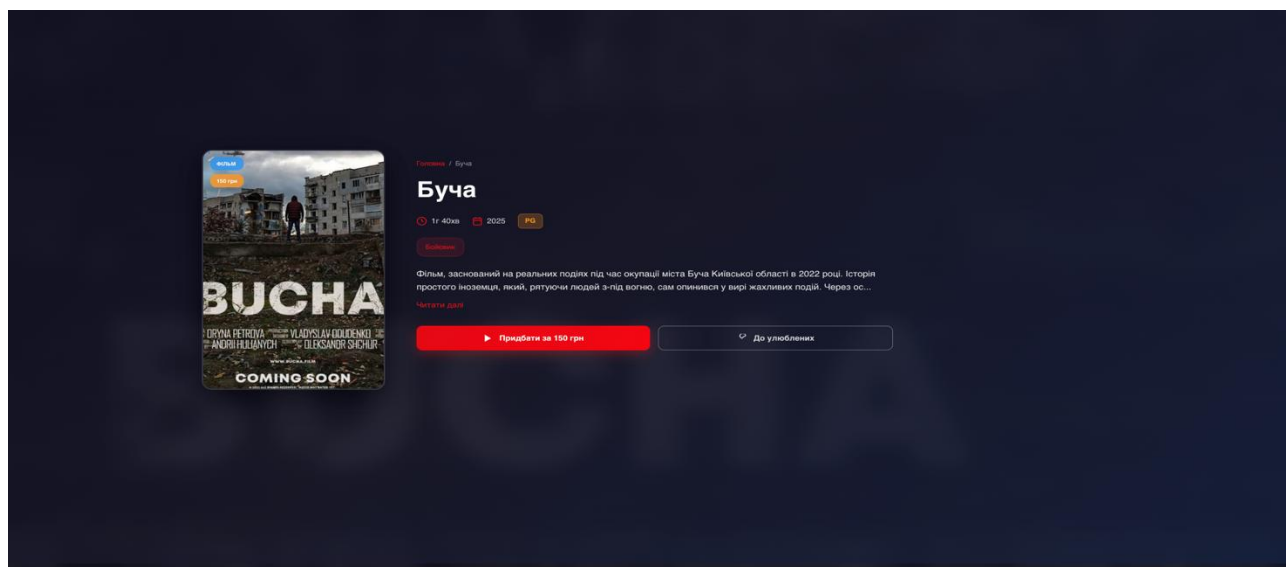


Рис. 3.8 – Інтерфейс придбання платного фільму

		V.B. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				47
Змн.	Арк.	№ докум.	Підпис	Дата		

Після того як користувач натискає кнопку «Придбати», він переходить до форми оплати (рис. 3.9), де має ввести реквізити банківської картки: номер картки, термін дії, CVV-код і ім'я власника. У цьому ж інтерфейсі відображається інформація про фільм, його тип та загальна сума до оплати. Користувач має можливість підтвердити оплату або скасувати її.

Після успішного введення платіжних даних і натискання кнопки «Оплатити 150 грн», система обробляє транзакцію, і в разі успішної оплати на екрані з'являється повідомлення «Оплата успішна!» з деталями платежу, зокрема: назва фільму, сума, дата та час, статус і ідентифікатор транзакції (рис. 3.10). Користувач також отримує підтвердження на електронну пошту та має можливість одразу перейти до перегляду фільму, натиснувши кнопку «Дивитися зараз», або повернутися до опису фільму.

Рис. 3.9 – Форма оплати фільму

Рис. 3.10 – Повідомлення про успішну оплату

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				48
Змн.	Арк.	№ докум.	Підпис	Дата		

На рисунку 3.11 зображено інтерфейс адміністративної панелі платформи BestFlix, яка призначена для керування основними функціональними складовими системи. Цей інтерфейс доступний лише користувачам із правами адміністратора і забезпечує централізований доступ до всіх ключових модулів. У верхній частині панелі виводиться особиста інформація адміністратора, зокрема ім'я користувача, електронна адреса, повне ім'я, дата реєстрації в системі та час останнього входу. Також відображається роль користувача з відповідним статусом – “Адміністратор”.

У центральній частині інтерфейсу представлено блоки з кнопками, що дозволяють швидко переходити до керування окремими розділами. Серед них – управління користувачами, управління жанрами, категоріями, фільмами, а також перегляд і контроль платежів. Така структура дозволяє адміністратору ефективно виконувати завдання з підтримки контенту, структури платформи й контролю доступу користувачів.

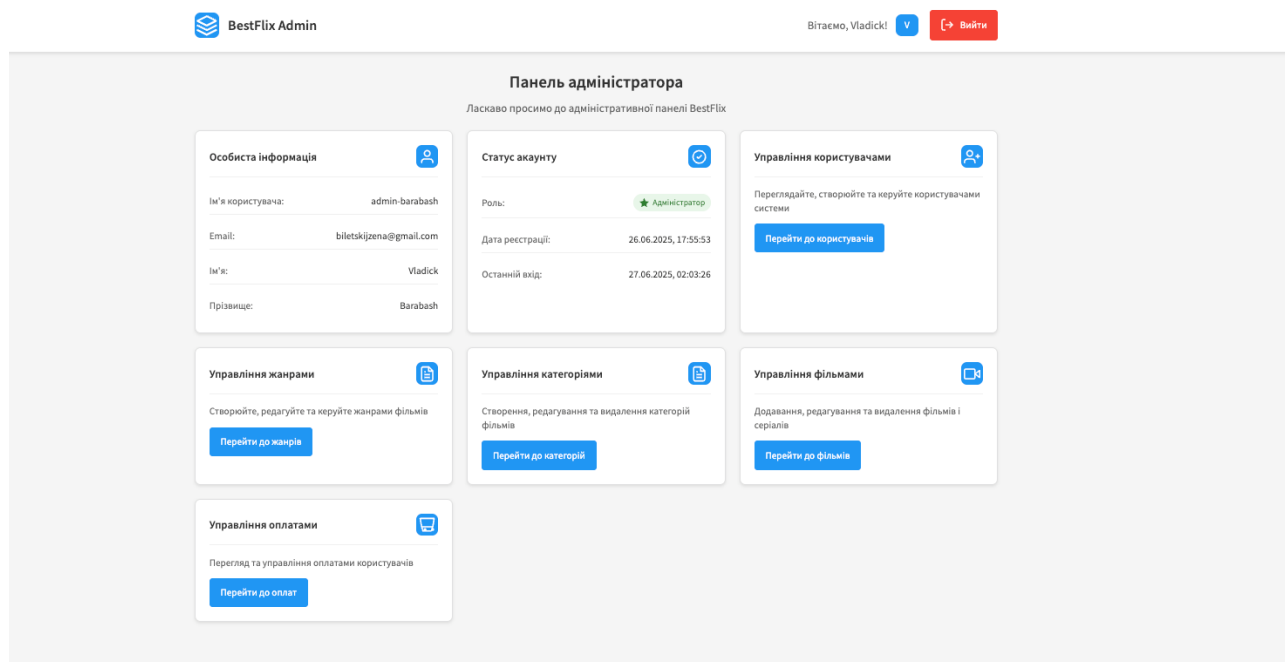


Рис. 3.11 – Інтерфейс адміністративної панелі

На рисунку 3.12 представлено сторінку керування жанрами у системі BestFlix, яка дозволяє адміністратору додавати, редагувати, деактивувати або видаляти жанри фільмів. Інтерфейс містить таблицю з назвами жанрів, коротким описом, статусом, датою створення та набором дій для кожного елемента.

		V.B. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижевська				49
Змн.	Арк.	№ докум.	Підпис	Дата		

Зокрема, для кожного жанру передбачено кнопки деактивації, редагування та видалення. У правому верхньому куті розташована кнопка “Створити жанр”, яка відкриває форму для додавання нового запису. Така функціональність дозволяє швидко адаптувати перелік жанрів до потреб платформи, забезпечуючи актуальність контенту.

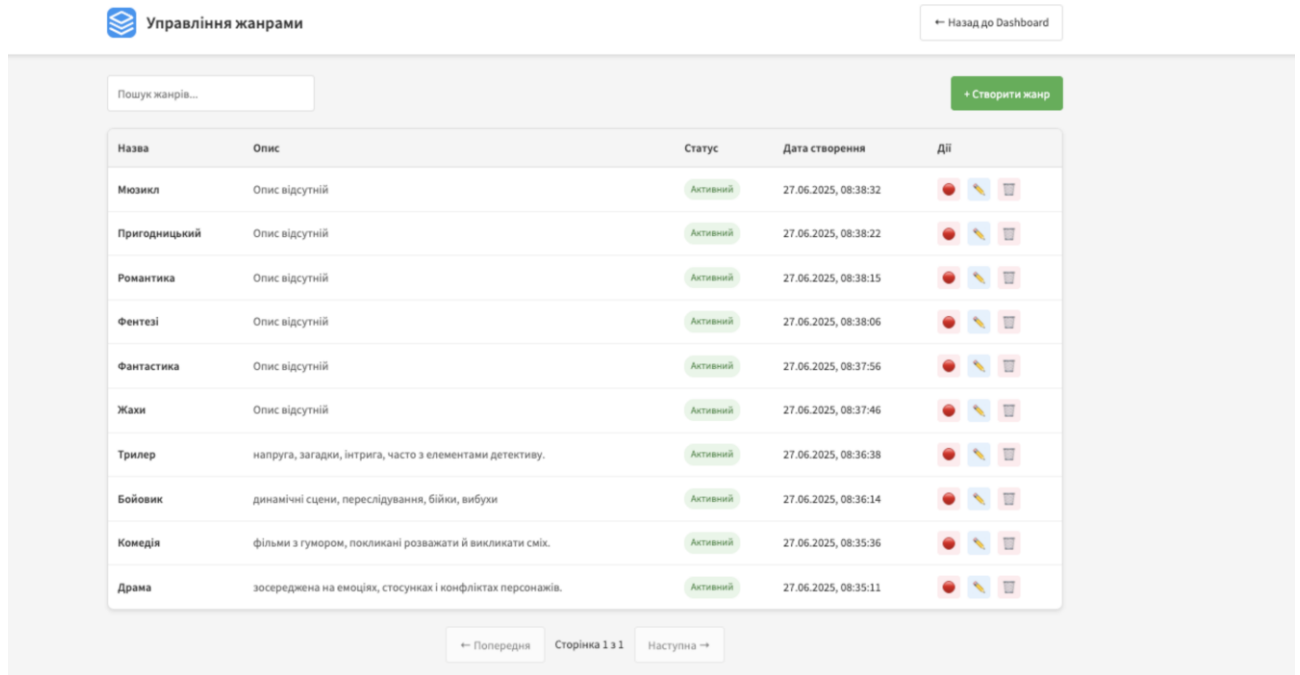


Рис. 3.12 – Інтерфейс керування жанрами

На рисунку 3.13 зображено інтерфейс керування категоріями у системі BestFlix. Ця сторінка дозволяє адміністратору створювати, редагувати та видаляти категорії для фільмів і серіалів. У таблиці відображається назва кожної категорії, її опис, тип, дата створення, а також кнопки для редагування та видалення. Зверху розміщено поле пошуку для швидкого знаходження потрібної категорії та кнопка «Створити категорію» для додавання нової. Такий функціонал забезпечує зручне адміністрування структури контенту на платформі.

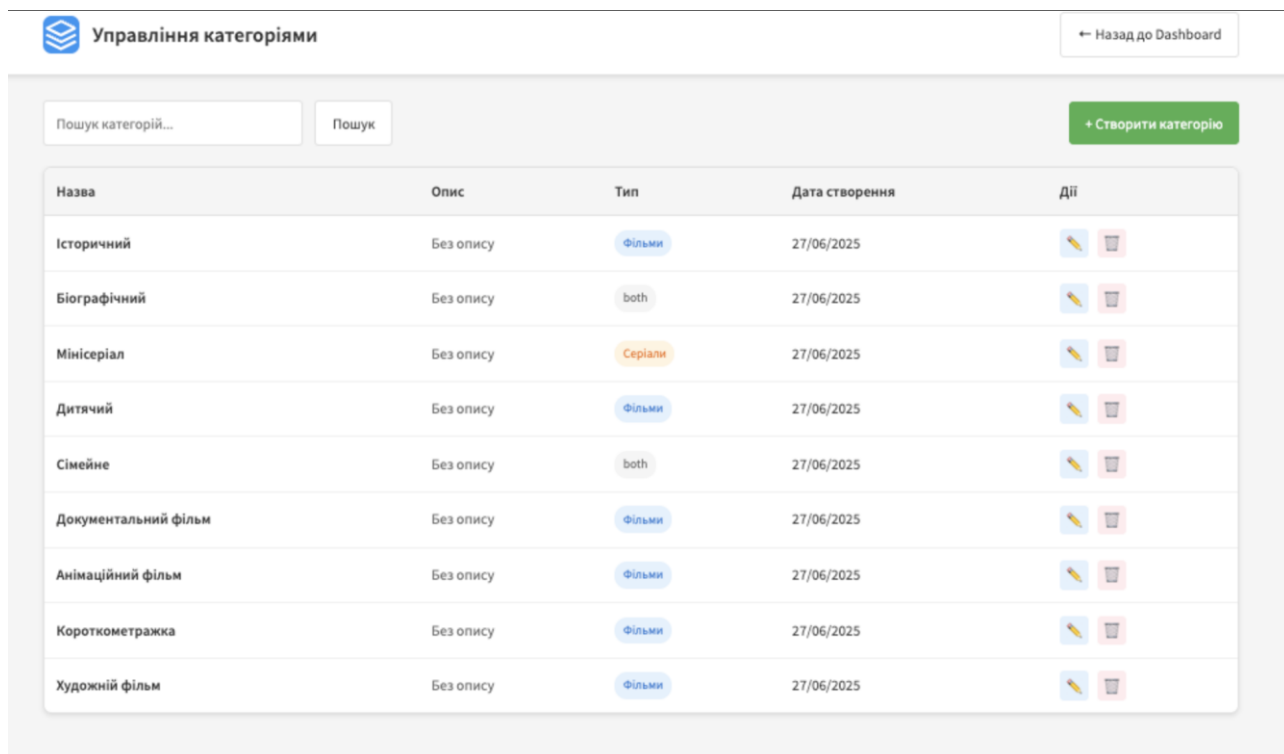


Рис. 3.13 – Сторінка керування категоріями

На рисунку 3.14 представлено інтерфейс сторінки керування фільмами в адміністративній панелі системи BestFlix. Цей розділ дозволяє адміністратору переглядати перелік усіх доданих фільмів, фільтрувати їх за назвою, а також здійснювати основні дії - редагування та видалення. У таблиці зручно відображено мініатюру постеру, назву фільму, тип (наприклад, фільм), рік випуску, жанр, категорію та ціну. Для прикладу, фільм «Буча» належить до жанру бойовик, має історичну категорію і коштує 150 грн, тоді як «Вендсдей» є жахами в категорії мінісеріал і доступний безкоштовно. Над таблицею розміщено поле пошуку для зручної навігації, а також кнопка «Додати фільм», яка відкриває форму для створення нового запису. Завдяки такому інтерфейсу адміністратор має змогу ефективно керувати кіноконтентом платформи.

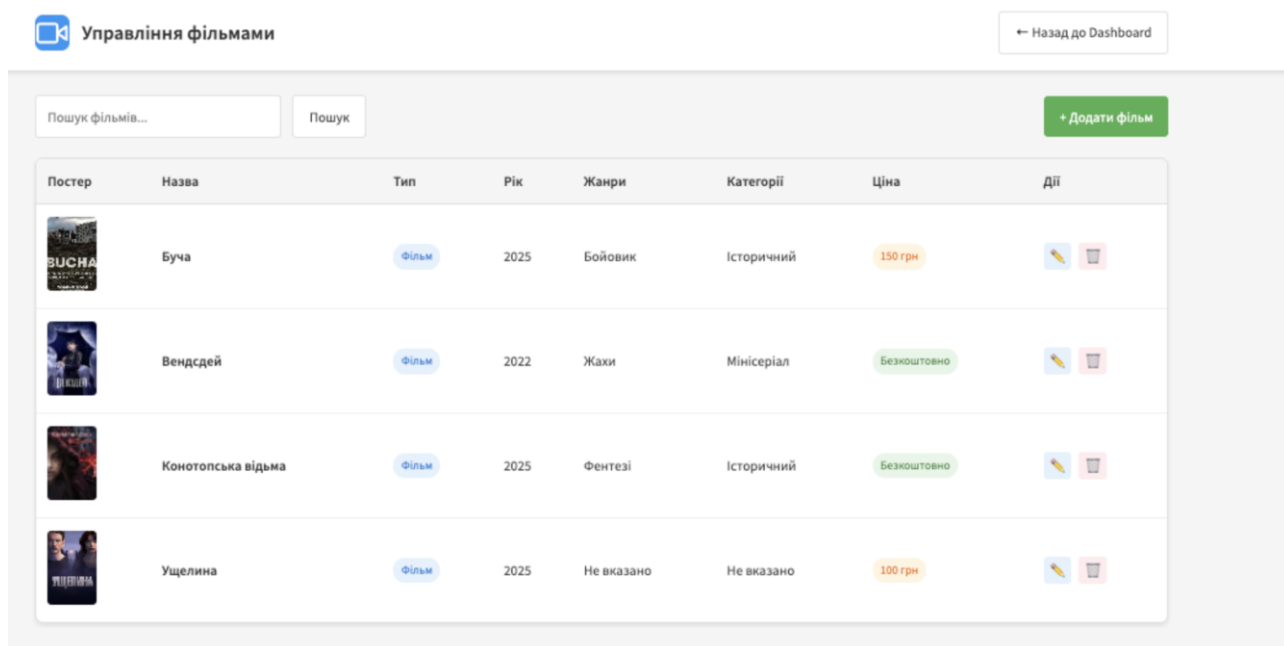


Рис. 3.14 – Сторінка керування фільмами

На рисунку 3.15 зображено сторінку керування платежами, яка доступна адміністратору платформи. Вона містить загальні статистичні показники: кількість усіх оплат, завершених, тих, що очікують, а також сумарний дохід. У нижній частині відображається детальна таблиця з інформацією про всі транзакції - ім'я користувача, назву фільму, суму оплати, статус платежу, спосіб оплати, дату здійснення та можливі дії. У прикладі показано, що користувач admin-barabash здійснив дві оплати карткою: за фільм «Буча» на 150 грн та «Ущелина» на 100 грн. Обидві транзакції завершені, а загальний дохід склав 250 грн. Такий інтерфейс дозволяє адміністратору легко контролювати надходження коштів та керувати фінансовими даними.

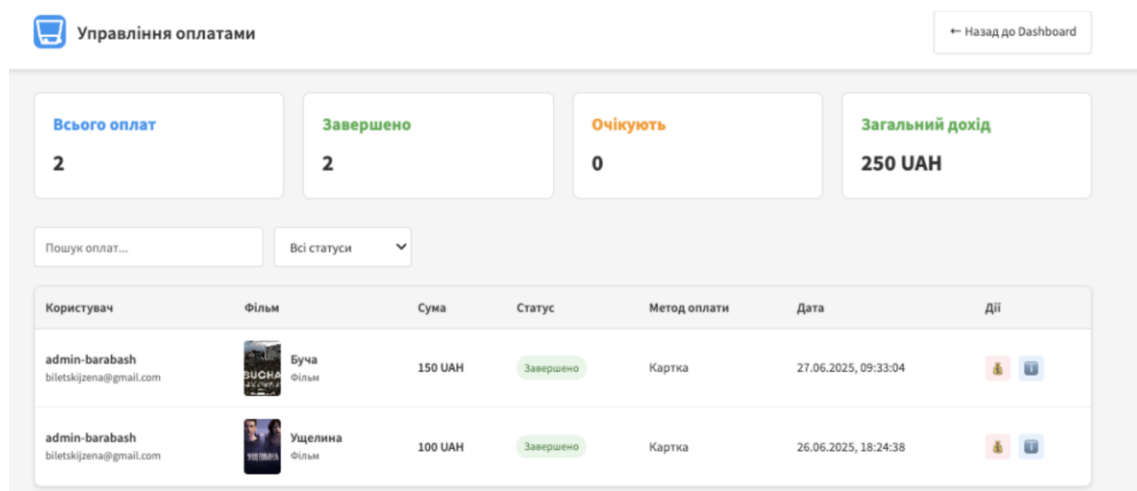


Рис. 3.15 – Керування категоріями ювелірних виробів в адмін-панелі

Також адміністратор має можливість керувати користувачами системи. На рисунку 3.16 представлено інтерфейс сторінки управління користувачами, де адміністратор може переглядати список усіх зареєстрованих користувачів, їхні електронні адреси, статуси активності, ролі, дату реєстрації та останній вхід у систему. Крім цього, в інтерфейсі доступні дії для кожного користувача, зокрема блокування, перегляд профілю або повне видалення. Наприклад, користувач із логіном admin-barabash має статус «Активний», роль «Адміністратор». Таким чином, система забезпечує зручний контроль за обліковими записами та дозволяє оперативно реагувати на зміни в активності або порушення правил платформи.

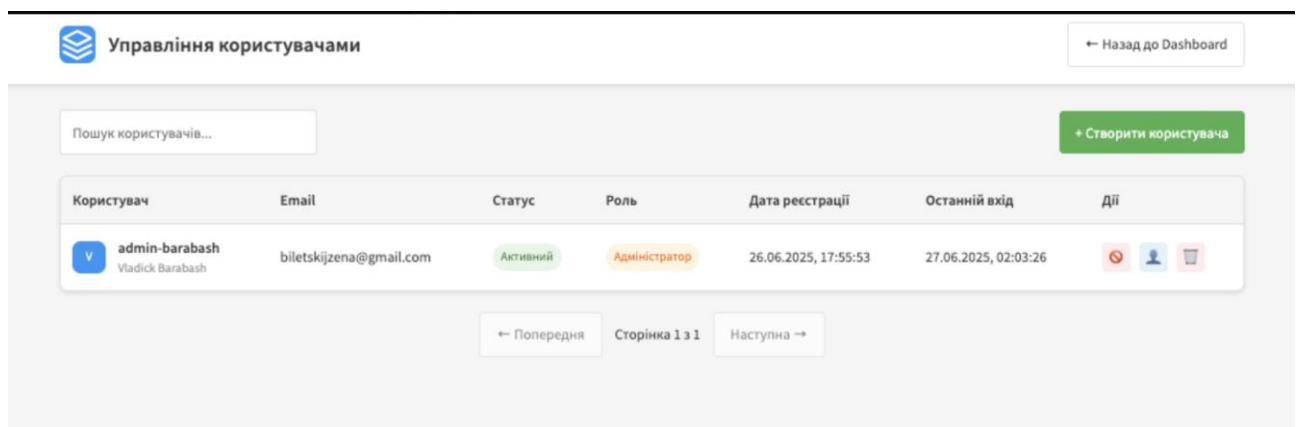


Рис. 3.16 – Сторінка керування користувачами

3.3. Тестування роботи програмного продукту

Після завершення основної розробки онлайн-кінотеатру було проведено ручне тестування, що охоплює як функціональні можливості, так і зручність взаємодії користувача з системою. Метою тестування було переконатися в коректній роботі всіх ключових модулів застосунку, відповідності реалізованих функцій до вимог технічного завдання, а також забезпеченні стабільності та безпечності системи в реальних умовах. Окрема увага приділялась перевірці обробки файлів, авторизації користувачів, створення фільмів і здійснення замовлень.

Функціональні перевірки включали наступні сценарії: реєстрація та вхід користувача, авторизація адміністратора, створення нового фільму з завантаженням зображень і трейлера, фільтрація за категоріями та жанрами,

		V.B. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				53
Змн.	Арк.	№ докум.	Підпис	Дата		

пошук, перегляд фільму, додавання в кошик, оформлення покупки, перевірка платежів, а також доступ до списку придбаних фільмів. Також протестовано захист приватних маршрутів за допомогою middleware та обмеження прав доступу до адміністративного функціоналу.

Окремо було перевірено адаптивність інтерфейсу на різних пристроях, коректне відображення сторінок у популярних браузерях, та відсутність критичних помилок при різному навантаженні. Усі тести проводились за попередньо складеними чеклістами, де фіксувались: функція, кроки перевірки, очікуваний результат та фактичний статус. Це дозволило структуровано оцінити якість реалізації програмного продукту та виявити незначні недоліки, які були усунуті на завершальному етапі розробки (таб. 3.1).

Таблиця 3.1

Результати тестування основних функцій програмного продукту

№ тесту	Функціональність	Кроки перевірки	Очікуваний результат	Статус
1	Реєстрація користувача	Відкрити сторінку реєстрації, заповнити форму, натиснути “Зареєструватися”	Користувача створено, автоматичний перехід до особистого кабінету	Пройдено
2	Авторизація користувача	Ввести email і пароль, натиснути “Увійти”	Перехід до кабінету, ініціалізація сесії через JWT	Пройдено

Продовження таблиці 3.1

3	Додавання фільму (адміністратор)	Увійти як адміністратор, відкрити форму додавання фільму, заповнити поля	Новий фільм з'являється у списку, медіафайли збережені в Cloudinary	Пройдено
4	Пошук фільму	Увести ключове слово у пошуковий рядок	Виводиться список релевантних фільмів	Пройдено
5	Фільтрація за категорією	Вибрати категорію з доступного списку	Виводяться лише фільми відповідної категорії	Пройдено
6	Перегляд фільму	Натиснути на картку фільму у списку	Відкривається сторінка з описом, трейлером і кнопкою покупки	Пройдено
7	Покупка фільму	Додати фільм у кошик, перейти до оплати, вибрати спосіб і підтвердити	Виводиться повідомлення про успішну покупку, запис зберігається у БД	Пройдено
8	Захист адміністративного доступу	Увійти як звичайний користувач, перейти на /admin	Виводиться повідомлення про відсутність прав або редирект	Пройдено

9	Видалення фільму	Увійти як адміністратор, перейти до фільму, натиснути “Видалити”	Фільм видалено з бази та Cloudinary, відображення оновлюється	Пройдено
10	Адаптивність інтерфейсу	Відкрити сайт на мобільному пристрої	Усі елементи інтерфейсу коректно масштабуються, навігація працює	Пройдено

Висновки до третього розділу

У третьому розділі описано архітектуру, основні компоненти та логіку взаємодії системи онлайн-кінотеатру. Було наведено діаграму розгортання та пояснено принципи взаємодії клієнта, сервера, бази даних і хмарного сховища. Розглянуто порядок встановлення і налаштування системи, включно з API-роутами, middleware та завантаженням медіа.

Також проаналізовано інтерфейс користувача, логіку фільтрації, пошуку, відображення відео та функціонал адміністратора. Пояснено структуру контролерів і реалізацію авторизації за допомогою JWT.

Проведено ручне тестування функціональності, результати якого підтвердили стабільну роботу системи.

ВИСНОВКИ

У ході виконання бакалаврської кваліфікаційної роботи було не просто реалізовано технічний прототип, а створено повнофункціональний вебзастосунок онлайн-кінотеатру, який комплексно відповідає сучасним вимогам продуктивності, інформаційної безпеки й ергономіки. Першочергово здійснено ґрунтовний аналіз ринкових аналогів, що дозволило чітко сформулювати конкурентні переваги майбутньої системи, визначити критичні місця в існуючих платформах та розробити набір вимог для її архітектури. На основі отриманих результатів обґрунтовано вибір стеку технологій: у ролі серверного середовища використано Node.js з Express.js завдяки їх високій продуктивності, неблокувальній моделі введення-виведення та широкій екосистемі модулів; за рендеринг та інтерактивність клієнтської частини відповідає React, що забезпечує швидку реакцію інтерфейсу, ефективно повторне використання компонентів і можливість подальшого масштабування; для зберігання даних застосовано MongoDB, яка надає гнучку схему та горизонтальну масштабованість, а для керування мультимедійними файлами інтегровано Cloudinary, що знімає навантаження з основного сервера та спрощує роботу з проміжною оптимізацією зображень і відео.

Функціонально система охоплює всі ключові сценарії взаємодії користувача з мультимедійним контентом. Користувачі можуть реєструватися та проходити автентифікацію, здійснювати пошук і фільтрацію за жанрами, категоріями, країною та роком випуску, керувати персональними підбірками, залишати контент у «Обраному», переглядати історію покупок і редагувати елементи власного профілю. Паралельно з цим для адміністраторів реалізовано захищену панель керування, де передбачено створення, оновлення та видалення фільмів, серіалів, жанрів і категорій, а також перегляд агрегованої статистики за відвідуваністю, популярністю певних стрічок і загальними тенденціями споживання контенту. Усі ці можливості поєднуються з надійним багаторівневим контролем доступу, що базується на JWT-токенах і проміжних

		<i>В.В. Барабаш</i>			<i>ІПЗ.КР.Б – 121 – 25 – ПЗ</i>	<i>Арк.</i>
		<i>О.Г. Чижмотря</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		<i>57</i>

middleware-перевірках, які запобігають несанкціонованому втручанню та забезпечують правильний розподіл ролей.

Особливу увагу зосереджено на якості коду та стабільності роботи застосунку. Проведено ручне тестування основних REST-ендпоінтів, перевірено коректність валідації вхідних даних і обробку помилок, протестовано відображення інтерфейсу на різних діагоналях екранів і швидкодію під час одночасного завантаження великої кількості медіафайлів. Результати тестів засвідчили відсутність критичних дефектів і підтвердили, що система витримує заявлені навантаження без деградації користувацького досвіду. Крім того, налаштовано базову CI/CD-конвеєризацію, що автоматизує збірку, запуск тестів і деплой, зменшуючи ризики помилок під час розгортання нових версій.

Отже, підсумовуючи виконану роботу, можна стверджувати, що розроблений онлайн-кінотеатр перебуває на високому рівні готовності до промислового впровадження. Його архітектура закладає фундамент для подальшого розвитку: інтеграції мобільного клієнта або PWA-версії, розширення мовних локалізацій, упровадження персоналізованих алгоритмів рекомендацій на основі машинного навчання, а також переходу до підписних або гібридних моделей монетизації. Система вже зараз може стати повноцінним майданчиком для дистрибуції ліцензійного контенту, а її гнучка модульна організація відкриває широкі можливості для адаптації під потреби конкретного бізнесу чи стартап-ініціативи.

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				58
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Netflix [Електронний ресурс]. – Режим доступу: <https://www.netflix.com/ua/>
2. Amazon Prime Video [Електронний ресурс]. – Режим доступу: https://www.primevideo.com/offers/nonprimehomepage/ref=dv_web_force_root
3. Disney+ [Електронний ресурс]. – Режим доступу: <https://www.disneyplus.com/welcome/unavailable>
4. HBO Max [Електронний ресурс]. – Режим доступу: <https://www.max.com/geo-availability>
5. Документація React [Електронний ресурс] - Режим доступу до ресурсу: <https://react.dev/>
6. Документація Express [Електронний ресурс] - Режим доступу до ресурсу: <https://expressjs.com/>
7. SASS [Електронний ресурс] - Режим доступу до ресурсу: <https://sass-lang.com/>
8. Документація Material-UI [Електронний ресурс] - Режим доступу до ресурсу: <https://mui.com/material-ui/>
9. Документація React Router DOM [Електронний ресурс] - Режим доступу до ресурсу: <http://reactrouter.com/>
10. Node.js [Електронний ресурс] - Режим доступу до ресурсу: <https://nodejs.org/uk>
11. MongoDB [Електронний ресурс] - Режим доступу до ресурсу: <https://www.mongodb.com/>
12. JWT [Електронний ресурс] - Режим доступу до ресурсу: <https://jwt.io/>
13. Cloudinary [Електронний ресурс] - Режим доступу до ресурсу: <https://cloudinary.com/>
14. Документація Stripe [Електронний ресурс] - Режим доступу до ресурсу: <https://stripe.com/>
15. Nodemon [Електронний ресурс] - Режим доступу до ресурсу: <https://nodemon.io/>

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмоторя				59
Змн.	Арк.	№ докум.	Підпис	Дата		

16. PostgreSQL [Електронний ресурс] - Режим доступу до ресурсу:
<https://www.postgresql.org/>
17. MySQL [Електронний ресурс] - Режим доступу до ресурсу:
<https://www.mysql.com/>
18. Документація Tailwind CSS [Електронний ресурс] - Режим доступу до ресурсу: <https://tailwindcss.com/docs>
19. Joana Almeida. Node.js vs. PHP: An In-Depth Comparison Guide for Web Development – DistantJob [Електронний ресурс] / Joana Almeida – 9 червня 2022 – Режим доступу до ресурсу: <https://distantjob.com/blog/nodejs-vsphp/>.
20. What is middleware? [Електронний ресурс] – Режим доступу до ресурсу: <https://www.redhat.com/en/topics/middleware/what-is-middleware>
21. Cloudinary for Node.js [Електронний ресурс] - Режим доступу до ресурсу: https://cloudinary.com/documentation/node_integration
22. Building your own hooks [Електронний ресурс] - Режим доступу до ресурсу: <https://legacy.reactjs.org/docs/hooks-custom.html>
23. Документація JavaScript [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
24. JOURNAL I. ReactJS: A Comprehensive Analysis of its features, Performance, and Suitability for Modern Web Development. INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT. 2023. Vol. 07, no. 09. P. 1. URL: <https://doi.org/10.55041/ijssrem25667>.
25. Clean Architecture [Електронний ресурс] – Режим доступу до ресурсу: <https://medium.com/codenx/clean-architecture-4c23dfa4c698>
26. Роберт М. Чиста архітектура: Мистецтво створення програмного забезпечення. Роберт М. [Видання друге] – Харків: Фабула, 2023 – 368 с
27. Learning React: Modern Patterns for Developing React Apps, 2nd edition / Alex Banks, Eve Porcello, 2020 – 307 p.
28. React Explained: Your Step-by-Step Guide to React, 1st edition / Zac Gordon, 2019 – 366 p.

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмоторя				60
Змн.	Арк.	№ докум.	Підпис	Дата		

29. Трофименко О. Г. Веб-технології та веб-дизайн : навч. посібник / О. Г. Трофименко, О. Б. Козін, О. В. Задерейко, О. Є. Плачінда. – Одеса : Фенікс, 2019. – 284 с.

30. Тест-кейс – один із ключових інструментів тестувальників ПЗ. [Електронний ресурс] – Режим доступу до ресурсу: <https://foxminded.ua/test-keis/>.

31. Документація HTML5 [Електронний ресурс] – Режим доступу до ресурсу: <https://webplatform.github.io/docs/html/html5/>

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				61
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				62
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг server.js:

```

const app = require("./app");
const mongoose = require("mongoose");
const dotenv = require("dotenv");

dotenv.config();

const port = process.env.PORT || 5000;

async function startServer() {
  try {
    await connectDb();

    app.listen(port, () => {
      console.log(`Server is running on port ${port}`);
      console.log(`API доступне на: http://localhost:${port}`);
      console.log(`Health check: http://localhost:${port}/health`);
    });
  } catch (error) {
    console.error("Помилка запуску сервера:", error);

    console.log("Запуск сервера без підключення до БД...");
    app.listen(port, () => {
      console.log(`Server is running on port ${port} (без БД)`);
      console.log(`API доступне на: http://localhost:${port}`);
    });
  }
}

const connectDb = async () => {
  try {
    if (
      !process.env.MONGODB_URL ||
      process.env.MONGODB_URL === "mongodb://localhost:27017/bestflix"
    ) {
      throw new Error("MongoDB URL не налаштований або використовує localhost");
    }

    await mongoose.connect(process.env.MONGODB_URL, {
      useNewUrlParser: true,
      useUnifiedTopology: true,
    });
  }
}

```

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				63
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    console.log("MongoDB підключено успішно");
  } catch (error) {
    console.error("MongoDB connection error:", error.message);
    throw error;
  }
};

```

```

const mongooseDisconnect = async () => {
  try {
    await mongoose.disconnect();
    console.log("MongoDB відключено");
  } catch (error) {
    console.error("Помилка відключення MongoDB:", error);
  }
};

```

```

process.on("SIGINT", async () => {
  console.log("\nShutting down server...");
  await mongooseDisconnect();
  process.exit(0);
});

```

```

process.on("SIGTERM", async () => {
  console.log("\nShutting down server...");
  await mongooseDisconnect();
  process.exit(0);
});

```

```

startServer();

```

```

module.exports = {
  connectDb,
  mongooseDisconnect,
};

```

Лістинг clodinary.js

```

const cloudinary = require("cloudinary").v2;
const { CloudinaryStorage } = require("multer-storage-cloudinary");
const multer = require("multer");

```

```

cloudinary.config({
  cloud_name: process.env.CLOUDINARY_CLOUD_NAME,
  api_key: process.env.CLOUDINARY_API_KEY,
  api_secret: process.env.CLOUDINARY_API_SECRET,

```

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				64
Змн.	Арк.	№ докум.	Підпис	Дата		

```
});
```

```
const imageStorage = new CloudinaryStorage({  
  cloudinary: cloudinary,  
  params: {  
    folder: "bestflix/images",  
    allowed_formats: ["jpg", "jpeg", "png", "webp"],  
    transformation: [  
      { width: 1920, height: 1080, crop: "limit" },  
      { quality: "auto" },  
      { fetch_format: "auto" },  
    ],  
  },  
});
```

```
const videoStorage = new CloudinaryStorage({  
  cloudinary: cloudinary,  
  params: {  
    folder: "bestflix/videos",  
    resource_type: "video",  
    allowed_formats: ["mp4", "webm", "mov", "avi"],  
  },  
});
```

```
const uploadImages = multer({  
  storage: imageStorage,  
  limits: {  
    fileSize: 1000 * 1024 * 1024,  
  },  
  fileFilter: (req, file, cb) => {  
    if (file.mimetype.startsWith("image/")) {  
      cb(null, true);  
    } else {  
      cb(new Error("Тільки зображення дозволені!"), false);  
    }  
  },  
});
```

```
const uploadVideos = multer({  
  storage: videoStorage,  
  limits: {  
    fileSize: 500 * 1024 * 1024,  
  },  
  fileFilter: (req, file, cb) => {
```

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижморя				65
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    if (file.mimetype.startsWith("video/")) {
      cb(null, true);
    } else {
      cb(new Error("Тільки відео дозволені!"), false);
    }
  },
});

const uploadFields = multer({
  storage: new CloudinaryStorage({
    cloudinary: cloudinary,
    params: (req, file) => {
      let folder = "bestflix/";
      let resourceType = "auto";

      if (file.mimetype.startsWith("image/")) {
        folder += "images";
        resourceType = "image";
      } else if (file.mimetype.startsWith("video/")) {
        folder += "videos";
        resourceType = "video";
      }

      return {
        folder: folder,
        resource_type: resourceType,
        allowed_formats: file.mimetype.startsWith("image/")
          ? ["jpg", "jpeg", "png", "webp"]
          : ["mp4", "webm", "mov", "avi"],
      };
    },
  }),
  limits: {
    fileSize: 1000 * 1024 * 1024,
  },
}).fields([
  { name: "posterImage", maxCount: 1 },
  { name: "backdropImage", maxCount: 1 },
  { name: "thumbnailImage", maxCount: 1 },
  { name: "trailerUrl", maxCount: 1 },
  { name: "videoUrl", maxCount: 1 },
]);

const deleteFromCloudinary = async (publicId) => {

```

		<i>В.В. Барабаш</i>			<i>ІПЗ.КР.Б – 121 – 25 – ПЗ</i>	Арк.
		<i>О.Г. Чижмотря</i>				66
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		

```

try {
  const result = await cloudinary.uploader.destroy(publicId);
  return result;
} catch (error) {
  console.error("Помилка видалення з Cloudinary:", error);
  throw error;
}
};

```

```

const getPublicIdFromUrl = (url) => {
  if (!url) return null;
  const parts = url.split("/");
  const filename = parts[parts.length - 1];
  return filename.split(".")[0];
};

```

```

module.exports = {
  cloudinary,
  uploadImages,
  uploadVideos,
  uploadFields,
  deleteFromCloudinary,
  getPublicIdFromUrl,
};

```

Лістинг movie.controller.js

```

const Movie = require("../models/Movie");
const Genre = require("../models/Genre");
const Category = require("../models/Category");
const {
  deleteFromCloudinary,
  getPublicIdFromUrl,
} = require("../config/cloudinary");

const createMovie = async (req, res) => {
  try {
    console.log("==== ПОЧАТКОВІ ДАНІ З ФОРМИ ====");
    console.log("req.body:", JSON.stringify(req.body, null, 2));
    console.log("req.files:", req.files);
    console.log("==== КІНЕЦЬ ПОЧАТКОВИХ ДАНИХ ====");

    const movieData = { ...req.body };

    if (req.files) {

```

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

console.log("Файли отримані:", req.files);
if (req.files.posterImage && req.files.posterImage[0]) {
    console.log("Poster path:", req.files.posterImage[0].path);
    movieData.posterImage = req.files.posterImage[0].path;
}
if (req.files.trailerUrl && req.files.trailerUrl[0]) {
    console.log("Trailer path:", req.files.trailerUrl[0].path);
    movieData.trailerUrl = req.files.trailerUrl[0].path;
}
if (req.files.videoUrl && req.files.videoUrl[0]) {
    console.log("Video path:", req.files.videoUrl[0].path);
    movieData.videoUrl = req.files.videoUrl[0].path;
}
} else {
    console.log("Файли не отримані або req.files undefined");
}

if (
    !movieData.posterImage ||
    typeof movieData.posterImage === "object" ||
    movieData.posterImage === "{}" ||
    movieData.posterImage === ""
) {
    delete movieData.posterImage;
}
if (
    !movieData.trailerUrl ||
    typeof movieData.trailerUrl === "object" ||
    movieData.trailerUrl === "{}" ||
    movieData.trailerUrl === ""
) {
    delete movieData.trailerUrl;
}
if (
    !movieData.videoUrl ||
    typeof movieData.videoUrl === "object" ||
    movieData.videoUrl === "{}" ||
    movieData.videoUrl === ""
) {
    delete movieData.videoUrl;
}

Object.keys(movieData).forEach((key) => {
    if (
        typeof movieData[key] === "object" &&
        movieData[key] !== null &&
        !Array.isArray(movieData[key]) &&
        Object.keys(movieData[key]).length === 0
    ) {
        delete movieData[key];
    }
});

```

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if (movieData.film_language === "") {
  delete movieData.film_language;
}
if (movieData.country === "") {
  delete movieData.country;
}
if (movieData.director === "") {
  delete movieData.director;
}
if (
  !movieData.title ||
  !movieData.description ||
  !movieData.posterImage ||
  !movieData.releaseYear
) {
  return res.status(400).json({
    success: false,
    message:
      "Обов'язкові поля: title, description, posterImage, releaseYear",
  });
}

if (movieData.duration) {
  movieData.duration = parseInt(movieData.duration);
}
if (movieData.releaseYear) {
  movieData.releaseYear = parseInt(movieData.releaseYear);
}
if (movieData.views) {
  movieData.views = parseInt(movieData.views) || 0;
}
if (movieData.likes) {
  movieData.likes = parseInt(movieData.likes) || 0;
}
if (movieData.rating) {
  movieData.rating = parseFloat(movieData.rating) || 0;
}

if (movieData.genres) {
  let genreIds = [];
  if (typeof movieData.genres === "string") {
    try {
      genreIds = JSON.parse(movieData.genres);
    } catch {
      genreIds = movieData.genres.split(",").map((g) => g.trim());
    }
  } else if (Array.isArray(movieData.genres)) {
    genreIds = movieData.genres;
  }
}

const genreObjects = await Genre.find({ _id: { $in: genreIds } });

```

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижморя				69
Змн.	Арк.	№ докум.	Підпис	Дата		

```

movieData.genres = genreObjects.map((genre) => genre._id);

console.log(
  "Знайдені жанри:",
  genreObjects.map((g) => ({ name: g.name, id: g._id })),
);
}

if (movieData.categories) {
  let categoryIds = [];
  if (typeof movieData.categories === "string") {
    try {
      categoryIds = JSON.parse(movieData.categories);
    } catch {
      categoryIds = movieData.categories.split(",").map((c) => c.trim());
    }
  } else if (Array.isArray(movieData.categories)) {
    categoryIds = movieData.categories;
  }

  const categoryObjects = await Category.find({
    _id: { $in: categoryIds },
  });
  movieData.categories = categoryObjects.map((category) => category._id);

  console.log(
    "Знайдені категорії:",
    categoryObjects.map((c) => ({ name: c.name, id: c._id })),
  );
}

if (movieData.cast) {
  if (typeof movieData.cast === "string") {
    try {
      movieData.cast = JSON.parse(movieData.cast);
    } catch {
      movieData.cast = movieData.cast.split(",").map((c) => c.trim());
    }
  }

  if (Array.isArray(movieData.cast)) {
    movieData.cast = movieData.cast
      .map((actor) => {
        if (typeof actor === "string" && actor.startsWith("[")) {
          try {
            return JSON.parse(actor);
          } catch {
            return actor;
          }
        }
        return actor;
      })
  }
}

```

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмоторя				70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        .flat();
    }
}

if (movieData.seasons && typeof movieData.seasons === "string") {
    try {
        movieData.seasons = JSON.parse(movieData.seasons);
    } catch {
        movieData.seasons = [];
    }
}

if (movieData["pricing.buyPrice"] !== undefined) {
    movieData.pricing = {
        buyPrice: parseFloat(movieData["pricing.buyPrice"]) || 0,
        isFree: movieData["pricing.isFree"] === "true",
    };
    delete movieData["pricing.buyPrice"];
    delete movieData["pricing.isFree"];
}

Object.keys(movieData).forEach((key) => {
    if (
        movieData[key] === "" ||
        movieData[key] === null ||
        movieData[key] === undefined
    ) {
        delete movieData[key];
    }
});

console.log("=== ДАНІ ДЛЯ ЗБЕРЕЖЕННЯ У БАЗУ ===");
console.log("movieData:", JSON.stringify(movieData, null, 2));
console.log("=== КІНЕЦЬ ДАНИХ ===");

const newMovie = new Movie(movieData);
const savedMovie = await newMovie.save();

await savedMovie.populate("genres categories");

res.status(201).json({
    success: true,
    message: "Фільм успішно створено",
    movie: savedMovie,
});
} catch (error) {
    console.error("Помилка створення фільму:", error);
    res.status(500).json({
        success: false,
        message: "Помилка сервера при створенні фільму",
        error: error.message,
    });
});

```

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмоторя				71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
  };

const updateMovie = async (req, res) => {
  try {
    const { id } = req.params;
    const existingMovie = await Movie.findById(id);

    if (!existingMovie) {
      return res.status(404).json({
        success: false,
        message: "Фільм не знайдено",
      });
    }

    const updateData = { ...req.body };
    const filesToDelete = [];

    if (req.files) {
      const fileFields = [
        "posterImage",
        "backdropImage",
        "thumbnailImage",
        "trailerUrl",
        "videoUrl",
      ];

      for (const field of fileFields) {
        if (req.files[field]) {
          if (existingMovie[field]) {
            const publicId = getPublicIdFromUrl(existingMovie[field]);
            if (publicId) filesToDelete.push(publicId);
          }

          updateData[field] = req.files[field][0].path;
        }
      }
    }

    if (updateData.genres) {
      let genreNames = [];
      if (typeof updateData.genres === "string") {
        try {
          genreNames = JSON.parse(updateData.genres);
        } catch {
          genreNames = updateData.genres.split(",").map((g) => g.trim());
        }
      } else if (Array.isArray(updateData.genres)) {
        genreNames = updateData.genres;
      }

      const genreObjects = await Genre.find({ name: { $in: genreNames } });

```

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмоторя				72
Змн.	Арк.	№ докум.	Підпис	Дата		

```

updateData.genres = genreObjects.map((genre) => genre._id);
}

if (updateData.categories) {
  let categoryNames = [];
  if (typeof updateData.categories === "string") {
    try {
      categoryNames = JSON.parse(updateData.categories);
    } catch {
      categoryNames = updateData.categories.split(",").map((c) => c.trim());
    }
  } else if (Array.isArray(updateData.categories)) {
    categoryNames = updateData.categories;
  }

  const categoryObjects = await Category.find({
    name: { $in: categoryNames },
  });
  updateData.categories = categoryObjects.map((category) => category._id);
}

if (updateData.cast && typeof updateData.cast === "string") {
  updateData.cast = updateData.cast.split(",").map((c) => c.trim());
}

if (updateData.duration) {
  updateData.duration = parseInt(updateData.duration);
}
if (updateData.releaseYear) {
  updateData.releaseYear = parseInt(updateData.releaseYear);
}
if (updateData.views) {
  updateData.views = parseInt(updateData.views) || 0;
}
if (updateData.likes) {
  updateData.likes = parseInt(updateData.likes) || 0;
}
if (updateData.rating) {
  updateData.rating = parseFloat(updateData.rating) || 0;
}

if (updateData.seasons && typeof updateData.seasons === "string") {
  try {
    updateData.seasons = JSON.parse(updateData.seasons);
  } catch {
    updateData.seasons = [];
  }
}

if (updateData.pricing && typeof updateData.pricing === "string") {
  try {
    updateData.pricing = JSON.parse(updateData.pricing);
  }
}

```

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмоторя				73
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    } catch {
      updateData.pricing = { buyPrice: 0, isFree: true };
    }
  }

Object.keys(updateData).forEach((key) => {
  if (
    updateData[key] === "" ||
    updateData[key] === null ||
    updateData[key] === undefined
  ) {
    delete updateData[key];
  }
});

console.log("=== ДАНІ ДЛЯ ОНОВЛЕННЯ У БАЗІ ===");
console.log("updateData:", JSON.stringify(updateData, null, 2));
console.log("=== КІНЕЦЬ ДАНИХ ===");

const updatedMovie = await Movie.findByIdAndUpdate(id, updateData, {
  new: true,
}).populate("genres categories");

for (const publicId of filesToDelete) {
  try {
    await deleteFromCloudinary(publicId);
  } catch (error) {
    console.error(`Помилка видалення файлу ${publicId}:`, error);
  }
}

res.status(200).json({
  success: true,
  message: "Фільм успішно оновлено",
  movie: updatedMovie,
});
} catch (error) {
  console.error("Помилка оновлення фільму:", error);
  res.status(500).json({
    success: false,
    message: "Помилка сервера при оновленні фільму",
    error: error.message,
  });
}
};

const deleteMovie = async (req, res) => {
  try {
    const { id } = req.params;
    const movie = await Movie.findById(id);

    if (!movie) {

```

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

return res.status(404).json({
  success: false,
  message: "Фільм не знайдено",
});
}

const filesToDelete = [];
const fileFields = [
  "posterImage",
  "backdropImage",
  "thumbnailImage",
  "trailerUrl",
  "videoUrl",
];

for (const field of fileFields) {
  if (movie[field]) {
    const publicId = getPublicIdFromUrl(movie[field]);
    if (publicId) filesToDelete.push(publicId);
  }
}

await Movie.findByIdAndDelete(id);

for (const publicId of filesToDelete) {
  try {
    await deleteFromCloudinary(publicId);
  } catch (error) {
    console.error(`Помилка видалення файлу ${publicId}:`, error);
  }
}

res.status(200).json({
  success: true,
  message: "Фільм та всі пов'язані файли успішно видалено",
});
} catch (error) {
  console.error("Помилка видалення фільму:", error);
  res.status(500).json({
    success: false,
    message: "Помилка сервера при видаленні фільму",
    error: error.message,
  });
}
};

const getMovie = async (req, res) => {
  try {
    const movie = await Movie.findById(req.params.id)
      .populate("genres", "name description")
      .populate("categories", "name description type");
  }
}

```

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if (!movie) {
  return res.status(404).json({
    success: false,
    message: "Фільм не знайдено",
  });
}

res.status(200).json({
  success: true,
  movie,
});
} catch (error) {
  res.status(500).json({
    success: false,
    message: "Помилка сервера",
    error: error.message,
  });
}
};

const getAllMovies = async (req, res) => {
  try {
    const {
      genre,
      category,
      type,
      page = 1,
      limit = 10,
      search,
      sortBy = "createdAt",
      sortOrder = "desc",
    } = req.query;

    let query = {};

    if (genre) {
      const genreObj = await Genre.findOne({ name: new RegExp(genre, "i") });
      if (genreObj) {
        query.genres = genreObj._id;
      } else {
        query.genres = genre;
      }
    }

    if (category) {
      const categoryObj = await Category.findOne({
        name: new RegExp(category, "i"),
      });
      if (categoryObj) {
        query.categories = categoryObj._id;
      } else {
        query.categories = category;
      }
    }
  }
};

```

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
  }

  if (type) {
    query.type = type;
  }

  if (search) {
    const searchRegex = new RegExp(search, "i");
    query.$or = [
      { title: searchRegex },
      { description: searchRegex },
      { director: searchRegex },
      { cast: { $in: [searchRegex] } },
    ];
  }

  const skip = (page - 1) * limit;
  const sortOptions = {};
  sortOptions[sortBy] = sortOrder === "desc" ? -1 : 1;

  const movies = await Movie.find(query)
    .populate("genres", "name description")
    .populate("categories", "name description type")
    .sort(sortOptions)
    .skip(skip)
    .limit(parseInt(limit));

  const total = await Movie.countDocuments(query);

  res.status(200).json({
    success: true,
    movies,
    pagination: {
      current: parseInt(page),
      pages: Math.ceil(total / limit),
      total,
    },
  });
} catch (error) {
  res.status(500).json({
    success: false,
    message: "Помилка сервера",
    error: error.message,
  });
}
};

const searchMovies = async (req, res) => {
  try {
    const { q, page = 1, limit = 10 } = req.query;

```

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if (!q) {
  return res.status(400).json({
    success: false,
    message: "Параметр пошуку обов'язковий",
  });
}

const searchRegex = new RegExp(q, "i");

const matchingGenres = await Genre.find({ name: searchRegex });
const matchingCategories = await Category.find({ name: searchRegex });

const query = {
  $or: [
    { title: searchRegex },
    { description: searchRegex },
    { director: searchRegex },
    { cast: { $in: [searchRegex] } },
    { genres: { $in: matchingGenres.map((g) => g._id) } },
    { categories: { $in: matchingCategories.map((c) => c._id) } },
  ],
};

const skip = (page - 1) * limit;

const movies = await Movie.find(query)
  .populate("genres", "name description")
  .populate("categories", "name description type")
  .sort({ createdAt: -1 })
  .skip(skip)
  .limit(parseInt(limit));

const total = await Movie.countDocuments(query);

res.status(200).json({
  success: true,
  movies,
  pagination: {
    current: parseInt(page),
    pages: Math.ceil(total / limit),
    total,
  },
});
} catch (error) {
  res.status(500).json({
    success: false,
    message: "Помилка пошуку",
    error: error.message,
  });
}
};

```

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижморя				78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const getMoviesByGenre = async (req, res) => {
  try {
    const { genre } = req.params;
    const { page = 1, limit = 10 } = req.query;

    let genreObj;
    if (genre.match(/^[0-9a-fA-F]{24}$/)) {
      genreObj = await Genre.findById(genre);
    } else {
      genreObj = await Genre.findOne({ name: new RegExp(genre, "i") });
    }

    if (!genreObj) {
      return res.status(404).json({
        success: false,
        message: "Жанр не знайдено",
      });
    }

    const query = { genres: genreObj._id };
    const skip = (page - 1) * limit;

    const movies = await Movie.find(query)
      .populate("genres", "name description")
      .populate("categories", "name description type")
      .sort({ createdAt: -1 })
      .skip(skip)
      .limit(parseInt(limit));

    const total = await Movie.countDocuments(query);

    res.status(200).json({
      success: true,
      genre: genreObj,
      movies,
      pagination: {
        current: parseInt(page),
        pages: Math.ceil(total / limit),
        total,
      },
    });
  } catch (error) {
    res.status(500).json({
      success: false,
      message: "Помилка отримання фільмів за жанром",
      error: error.message,
    });
  }
};

const getMoviesByCategory = async (req, res) => {
  try {

```

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const { category } = req.params;
const { page = 1, limit = 10 } = req.query;

let categoryObj;
if (category.match(/^[0-9a-fA-F]{24}$/)) {
  categoryObj = await Category.findById(category);
} else {
  categoryObj = await Category.findOne({ name: new RegExp(category, "i") });
}

if (!categoryObj) {
  return res.status(404).json({
    success: false,
    message: "Категорію не знайдено",
  });
}

const query = { categories: categoryObj._id };
const skip = (page - 1) * limit;

const movies = await Movie.find(query)
  .populate("genres", "name description")
  .populate("categories", "name description type")
  .sort({ createdAt: -1 })
  .skip(skip)
  .limit(parseInt(limit));

const total = await Movie.countDocuments(query);

res.status(200).json({
  success: true,
  category: categoryObj,
  movies,
  pagination: {
    current: parseInt(page),
    pages: Math.ceil(total / limit),
    total,
  },
});
} catch (error) {
  res.status(500).json({
    success: false,
    message: "Помилка отримання фільмів за категорією",
    error: error.message,
  });
}
};

const getMoviesStats = async (req, res) => {
  try {
    const totalMovies = await Movie.countDocuments();
    const totalSeries = await Movie.countDocuments({ type: "series" });
  }
}

```

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				80
Змн.	Арк.	№ докум.	Підпис	Дата		

```
const totalFilms = await Movie.countDocuments({ type: "movie" });
```

```
const genreStats = await Movie.aggregate([
  { $unwind: "$genres" },
  {
    $lookup: {
      from: "genres",
      localField: "genres",
      foreignField: "_id",
      as: "genreInfo",
    },
  },
  { $unwind: "$genreInfo" },
  {
    $group: {
      _id: "$genreInfo._id",
      name: { $first: "$genreInfo.name" },
      count: { $sum: 1 },
    },
  },
  { $sort: { count: -1 } },
  { $limit: 10 },
]);
```

```
const categoryStats = await Movie.aggregate([
  { $unwind: "$categories" },
  {
    $lookup: {
      from: "categories",
      localField: "categories",
      foreignField: "_id",
      as: "categoryInfo",
    },
  },
  { $unwind: "$categoryInfo" },
  {
    $group: {
      _id: "$categoryInfo._id",
      name: { $first: "$categoryInfo.name" },
      count: { $sum: 1 },
    },
  },
  { $sort: { count: -1 } },
  { $limit: 10 },
]);
```

```
const yearStats = await Movie.aggregate([
  { $group: { _id: "$releaseYear", count: { $sum: 1 } } },
  { $sort: { _id: -1 } },
  { $limit: 10 },
]);
```

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижевська				81
Змн.	Арк.	№ докум.	Підпис	Дата		

```

res.status(200).json({
  success: true,
  stats: {
    total: totalMovies,
    movies: totalFilms,
    series: totalSeries,
    topGenres: genreStats,
    topCategories: categoryStats,
    yearDistribution: yearStats,
  },
});
} catch (error) {
  res.status(500).json({
    success: false,
    message: "Помилка отримання статистики",
    error: error.message,
  });
}
};

```

```

module.exports = {
  createMovie,
  updateMovie,
  deleteMovie,
  getMovie,
  getAllMovies,
  searchMovies,
  getMoviesByGenre,
  getMoviesByCategory,
  getMoviesStats,
};

```

		В.В. Барабаш			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		О.Г. Чижмотря				82
Змн.	Арк.	№ докум.	Підпис	Дата		