

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОГО ПОРТАЛУ,
ПРИСВЯЧЕНОГО КІНО
SOFTWARE IMPLEMENTATION OF AN INFORMATION PORTAL
DEDICATED TO CINEMA

Виконав(ла): студент(ка) 4 курсу, групи КНТ-141

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

БУДЖЕЛІДА С.Н.

(ПРИЗВИЩЕ та ініціали)

Керівник ЛЕОЩЕНКО Д.С.

(ПРИЗВИЩЕ та ініціали)

Рецензент БАБЕНКО Н.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

БУДЖЕЛІДА Саміри Нуреддінівни
 (ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація інформаційного порталу,
присвяченого кіно.
Software Implementation of an Information Portal Dedicated to Cinema

керівник проєкту (роботи) д-р філософ., доцент, ЛЕОЩЕНКО Сергій
Дмитрович
 (науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 02 червня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження.
2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ЛЕОЩЕНКО С.Д., доцент		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання « 14 » квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Саміра БУДЖЕЛІДА
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Сергій ЛЕОЩЕНКО
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 92 с., 11 табл., 33 рис., 3 дод., 22 джерела.

ВЕБСАЙТ, РЕЄСТРАЦІЯ, АВТОРИЗАЦІЯ, КІНОФІЛЬМ, ВІДГУК, КОМЕНТАР, ТРЕЙЛЕР, ОБРАНЕ.

Об'єкт дослідження – процес розробки інформаційних вебсистем для поширення кіно- та мультимедійного контенту.

Предмет дослідження – засоби розробки інформаційної вебсистеми для поширення кіно- та мультимедійного контенту

Мета роботи – розробити інформаційний портал, присвячений кіно, з функціоналом авторизації, реєстрації, перегляду фільмів (трейлерів), оцінювання, коментування та адміністративного керування контентом.

Матеріали, методи та технічні засоби: У роботі використано такі технології: React (для клієнтської частини), Node.js + Express (серверна частина), Sequelize (ORM), PostgreSQL (база даних). Застосовано метод об'єктно-орієнтованого програмування, компонентний підхід до створення інтерфейсу, REST-архітектуру для API. Джерелами даних є публічні ресурси з описами та трейлерами фільмів, а також дані, які вводить користувач.

Результати. Було розроблено працюючий вебдодаток, який забезпечує реєстрацію та авторизацію користувачів, додавання фільмів адміністратором, перегляд, коментування та оцінювання фільмів. Додаток також надає можливість групувати фільми за жанрами, отримувати рекомендації та має зручний інтерфейс для користувача.

Висновки. Інформаційний портал дозволяє швидко знаходити цікаві фільми, ділитися думками з іншими користувачами, а адміністраторам – керувати вмістом. Такий ресурс сприяє покращенню доступу до культурного контенту та створенню кіноспільноти.

Галузь використання – платформа може бути використана як онлайн-ресурс для кіноманів, кінотеатрів, навчальних або культурних установ, або як основа для комерційного продукту (наприклад, онлайн-кінотеатр).

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 92 pages, 11 tables, 33 figures, 3 appendixes, 22 sources.

WEBSITE, REGISTRATION, AUTHORISATION, FILM, REVIEW, COMMENT, TRAILER, FAVOURITES.

Research object – the process of developing information web systems for distributing cinema and multimedia content.

Research subject – the means of developing an information web system for distributing cinema and multimedia content.

Research goal– To develop an information portal dedicated to cinema with the functionality of authorisation, registration, watching films (trailers), rating, commenting and administrative management of content.

Materials, methods and technical means: The following technologies were used in the work: React (client side), Node.js + Express (server side), Sequelize (ORM), PostgreSQL (database). The method of object-oriented programming, component approach to interface development, REST architecture for API were used. The data sources are public resources with film descriptions and trailers, as well as data entered by the user.

Results. A working web application has been developed that provides registration and authorisation of users, adding films by the administrator, viewing, commenting and rating films. The application also provides the ability to group films by genre, receive recommendations, and has a user-friendly interface.

Conclusions. The information portal allows users to quickly find interesting films, share their opinions with other users, and allow administrators to manage content. This resource helps to improve access to cultural content and create a film community.

Field of application – the platform can be used as an online resource for film lovers, cinemas, educational or cultural institutions, or as a basis for a commercial product (e.g. online cinema).

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	9
Вступ.....	10
1 Аналіз предметної області.....	12
1.1 Аналіз предметної області.....	12
1.2 Огляд існуючих систем, сервісів, сайтів, що надають змогу переглядати фільми і створювати власні колекції.....	13
1.2.1 Передмова	13
1.2.2 Сервіс Sweet.tv.....	15
1.2.3 Netflix	16
1.2.4 IMDb (Internet Movie Database).....	17
1.2.5 Amazon Prime Video.....	18
1.2.6 Порівняння існуючих систем, сервісів, сайтів, що надають змогу переглядати фільми і створювати власні колекції.....	19
1.2.7 Висновки за результатами порівняння.....	21
1.3 Визначення мети та завдань системи.....	22
1.4 Архітектурні рішення	22
1.5 Ключові вимоги до системи.....	23
1.6 Висновки з розділу	23
2 Аналіз програмних засобів.....	25
2.1 Огляд особливостей мов програмування.....	25
2.2 Огляд особливостей обраних платформ і бібліотек	28
2.3 Інструменти для роботи з базами даних та API.....	29
2.3.1 PostgreSQL – сучасна система керування базами даних.....	30
2.3.2 pgAdmin – інтерфейс для візуального керування базою.....	31
2.3.3 Postman – платформа для тестування API-запитів	32
2.3.4 Проектування структури бази даних у draw.io	34
2.3.5 Сумісна робота інструментів	35
3 Основні рішення з реалізації системи.....	37

3.1 Вибір засобів реалізації	37
3.2 Функціональні вимоги до системи	38
3.3 Розробка бази даних.....	39
3.4 Структура програмного забезпечення	42
3.5 Висновок до розділу	45
4 Експлуатація, тестування та експериментальне дослідження програми.....	46
4.1 Характеристики програми	46
4.2 Застосування програми.....	46
4.2.1 Сценарії використання (Use Case діаграми).....	46
4.2.2 Діаграма бізнес-процесу	49
4.3 Тестування програми	50
4.4 Структура програми.....	52
4.5 Експлуатація програми.....	53
Висновки	60
Перелік джерел посилання	62
Додаток А Технічне завдання	64
Додаток Б Текст програми	67
Додаток В Слайди презентації.....	84

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;

CRUD – Create, Read, Update, Delete;

draw.io – онлайн-інструмент для створення діаграм та схем;

JSON – JavaScript Object Notation;

ORM – Object-Relational Mapping;

pgAdmin – інструмент для візуального керування базою даних PostgreSQL;

Postman – інструмент для тестування API-запитів;

Sequelize – ORM-бібліотека для Node.js;

UI – User Interface;

БД – база даних.

ВСТУП

У сучасному інформаційному суспільстві вебтехнології стрімко розвиваються, охоплюючи всі сфери людської діяльності – від бізнесу до освіти, від науки до індустрії розваг. Особливо активний розвиток відбувається у галузі медіа-контенту, зокрема в кіноіндустрії, де цифрові платформи стали головним способом розповсюдження та споживання фільмів, серіалів, трейлерів і рецензій.

Поява таких платформ, як IMDb, Netflix, Letterboxd демонструє інтерес суспільства до персоналізованих і інтерактивних джерел медіа. Однак більшість із них або зосереджені на комерційній дистрибуції, або не дають можливості повноцінно взаємодіяти з контентом для пересічного користувача без платної підписки чи обмежень. Саме тому створення незалежного, функціонального, інформативного й доступного порталу, присвяченого кіно, є важливим і актуальним завданням.

Критичний аналіз: більшість існуючих платформ мають обмежений доступ до взаємодії з контентом (наприклад, залишати коментарі чи оцінки можна лише зареєстрованим або платним користувачам). Крім того, багато з них не забезпечують відкриту модерацію коментарів, не мають функції персональних рекомендацій або не дозволяють гнучко сортувати фільми за жанрами, рейтингами тощо.

Сучасні користувачі очікують від платформи не лише перегляду, а й можливості бути частиною спільноти – залишати думки, впливати на рейтинг, ділитися враженнями. Інтерфейс також часто перевантажений рекламою або неадаптований для мобільних пристроїв. Таким чином, є запит на зручні, естетичні та відкриті системи для кінолюбителів.

Ретроспективні огляди: упродовж останніх років спостерігається стійкий тренд до створення нішевих інформаційних порталів. Наприклад, платформи, присвячені лише аніме, документальному кіно або фільмам конкретного жанру, набирають популярності серед аудиторій, що прагнуть

спеціалізованого контенту. Проте часто ці ресурси створені або на застарілих технологіях, або не мають мобільної адаптації, або не підтримують сучасні інтерактивні функції, такі як рекомендаційні системи.

Вивчення прикладів таких сайтів дозволило виділити ключові функціональні потреби: зручний пошук, система оцінок, коментарі, індивідуальні рекомендації, можливість додавати новий контент. Все це враховано в цій дипломній роботі.

Актуальність: з огляду на популярність потокових сервісів і бажання користувачів брати активну участь у формуванні кіноконтенту (залишати рецензії, створювати добірки тощо), подібні проєкти мають високий попит.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

У цьому розділі було розглянуто існуючі кіносервіси, їхній функціонал та недоліки, визначено мету й завдання майбутнього вебдодатку, обґрунтовано вибір мікросервісної архітектур, сформульовано ключові функціональні й нефункціональні вимоги до системи, а також зроблено висновки щодо шляхів покращення існуючих рішень.

1.1 Аналіз предметної області

Кінематограф як складова сучасної культури є важливою галуззю інформаційного простору, що постійно розвивається. Зі зростанням кількості кінопродукції, а також із переходом користувачів до цифрового споживання контенту, актуальною стала необхідність у створенні інформаційних порталів, які б надавали доступ до структурованих, актуальних і достовірних даних про фільми, серіали, акторів, режисерів та інші елементи, пов'язані з кіноіндустрією.

Інформаційний портал, присвячений кіно, є вебресурсом, що забезпечує користувачів детальною інформацією про кінематографічні твори. Основними завданнями такого portalу є збирання, зберігання, обробка та відображення інформації про фільми у зручній формі. Також важливими складовими є інтерактивність, можливість користувацької взаємодії, а також функції персоналізації – рекомендації фільмів на основі інтересів, переглядів або оцінок користувача.

Потреба в інформаційних системах такого типу зумовлена значним обсягом інформації в галузі кінематографу. Кожного року випускається тисячі фільмів у різних країнах світу, що створює необхідність для ефективної систематизації та подання інформації. Користувачі очікують зручного та швидкого доступу до бази фільмів, можливості ознайомлення з трейлерами,

перегляду описів, оцінювання переглянутого контенту та обміну думками з іншими користувачами.

Сучасний інформаційний портал про кіно має відповідати низці вимог. По-перше, він має бути технічно надійним і доступним з будь-яких пристроїв. По-друге, необхідна зручна система навігації та фільтрації контенту за різними критеріями – жанром, роком випуску, країною, рейтингом тощо. По-третє, важливою є наявність користувацького функціоналу – авторизації, додавання оцінок і коментарів, можливості зберігати обрані фільми, отримувати рекомендації. Також важливо забезпечити оновлюваність інформації, що передбачає створення зручного інтерфейсу для адміністраторів платформи.

До основних об'єктів предметної області належать фільми, жанри, користувачі, оцінки, коментарі, актори, трейлери. Між цими об'єктами існують певні зв'язки, які необхідно правильно враховувати під час розробки структури даних та реалізації функціоналу portalу. Наприклад, один фільм може належати до кількох жанрів, мати багато акторів, отримувати численні оцінки та коментарі від різних користувачів.

Таким чином, аналіз предметної області дозволяє зробити висновок, що створення інформаційного portalу про кіно є доцільним та актуальним завданням. Така система повинна забезпечити користувачів зручним інструментом для пошуку, перегляду та оцінки фільмів, а також надати адміністраторам можливість ефективно керувати контентом.

1.2 Огляд існуючих систем, сервісів, сайтів, що надають змогу переглядати фільми і створювати власні колекції

1.2.1 Передмова

Перш ніж переходити до розробки власного інформаційного portalу, доцільно провести огляд і аналіз уже наявних рішень у сфері онлайн-дистрибуції фільмів. Такий аналіз дозволяє не лише виявити сильні сторони існуючих сервісів, але й визначити недоліки, які можна уникнути або

вдосконалити у власній системі. Основні розглянуті міжнародні та локальні кіносервіси:

- Sweet.tv,
- Netflix,
- IMDb,
- Amazon Prime Video.

Під час написання роботи було проведено аналіз основних сервісів та сайтів, які надають можливість перегляду фільмів та створення власних колекцій. Аналіз здійснювався за такими основними критеріями:

- можливість перегляду, сортування, коментування, оцінювання контенту;
- якість інтерфейсу;
- наявність персоналізації (рекомендації, історія перегляду, персональні добірки);
- мобільна адаптація (адаптивність під смартфони, планшети, Smart TV);
- швидкість завантаження, наявність кешування, стабільність роботи;
- авторизація через соцмережі, можливість створення власного списку перегляду, мовна локалізація.

На сучасному етапі розвитку цифрових технологій та глобалізації ринку розваг, онлайн-кіносервіси стали одним із головних каналів розповсюдження мультимедійного контенту. Вони відіграють ключову роль у формуванні культурного простору та задоволенні попиту користувачів на фільми, серіали, документальні стрічки та інший відеоконтент. Ринок стримінгових платформ є динамічним і насиченим, а сервіси конкурують між собою за рахунок унікального контенту, функціональності, якості інтерфейсу та персоналізованого досвіду користувача.

Серед найпопулярніших кіносервісів, які користуються попитом серед української аудиторії, варто виділити Sweet.tv, а також міжнародні платформи

Netflix, IMDb, Amazon Prime Video. Далі розглянемо ключові функції, переваги та недоліки цих платформ.

1.2.2 Сервіс Sweet.tv

Sweet.tv – це сучасний український OTT-сервіс (Over-The-Top), який надає доступ до телеканалів, фільмів, серіалів, спортивних трансляцій та дитячого контенту через інтернет [1]. Заснований у 2018 році, він швидко став одним із лідерів ринку легального відеоконтенту в Україні, орієнтуючись на якість, зручність та підтримку української мови.

Один із небагатьох офіційних українських стримінгових сервісів, який працює на підставі ліцензій.

Сервіс належить Sweet.TV Limited (афілійована структура медіахолдингу 1+1 media), що забезпечує йому доступ до ексклюзивного контенту, зокрема телеканалів групи (1+1, ТЕТ, ПлюсПлюс тощо) та власних продукційних проектів [1].

Інтерфейс вебзастосунку Sweet.TV показано на рис. 1.1 [2].

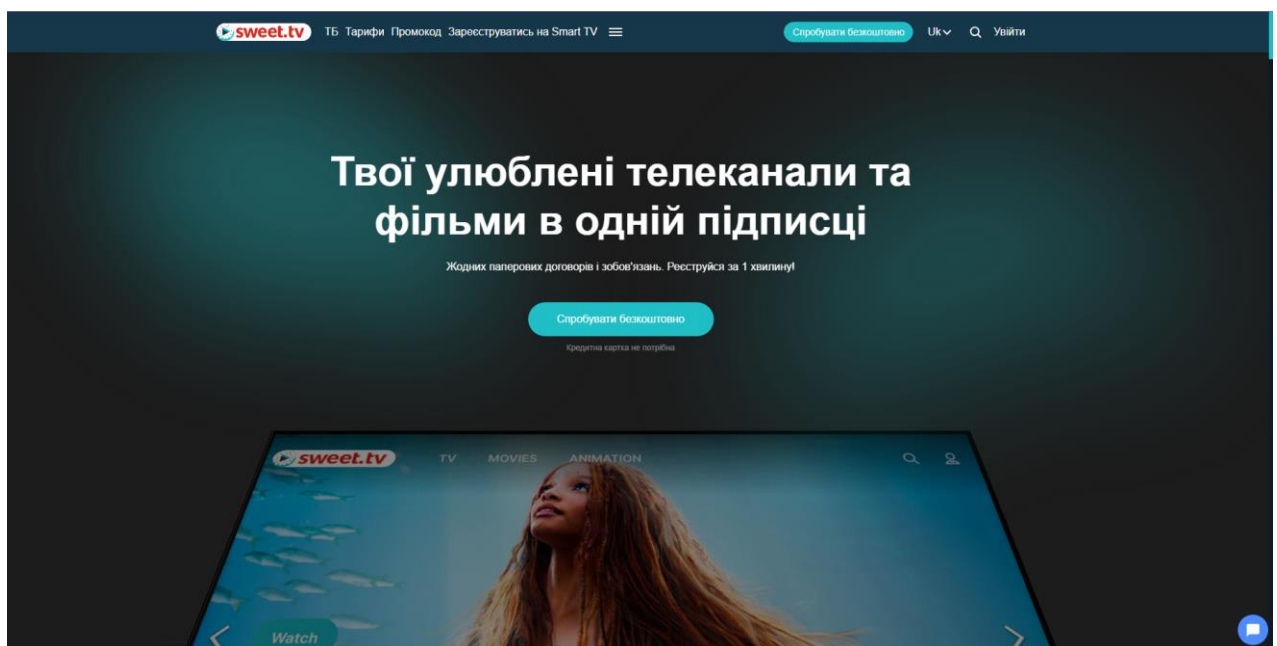


Рисунок 1.1 – Інтерфейс вебсервісу Sweet.tv [2]

Основні можливості:

- платна підписка з доступом до преміального контенту;
- велика база українських фільмів і серіалів;
- підтримка Smart TV;
- функція прокату фільмів;
- дубляж українською мовою.

Серед недоліків:

- обмеження для безкоштовних користувачів;
- не завжди стабільна якість стріму;
- менш гнучка система персоналізації порівняно з глобальними конкурентами.

1.2.3 Netflix

Netflix – це світовий лідер серед OTT-сервісів (Over-The-Top), який перетворив індустрію розваг, пропонуючи мільйонам користувачів стрімінг фільмів, серіалів, документалок та власного ексклюзивного контенту [3]. Заснований у 1997 році як сервіс з прокату DVD, Netflix у 2007 році перейшов на онлайн-трансляції та став піонером у сфері потокового відео. Сьогодні він доступний у більш ніж 190 країнах, має понад 260 мільйонів платних підписників (2024 рік) [3].

Найвідоміший глобальний стримінговий сервіс. Його переваги:

- офіційний ліцензійний контент;
- гнучка система рекомендацій на основі поведінки користувача;
- високоякісний власний контент (Netflix Originals);
- можливість завантаження відео для офлайн-перегляду;
- багатомовна підтримка;
- адаптивний інтерфейс для всіх пристроїв.

Серед мінусів:

- обмежена бібліотека в Україні через регіональні обмеження;

- відсутність повноцінного дубляжу українською для багатьох фільмів;
- висока ціна передплати для деяких категорій користувачів.

Інтерфейс вебзастосунку Netflix показано на рис. 1.2 [4].

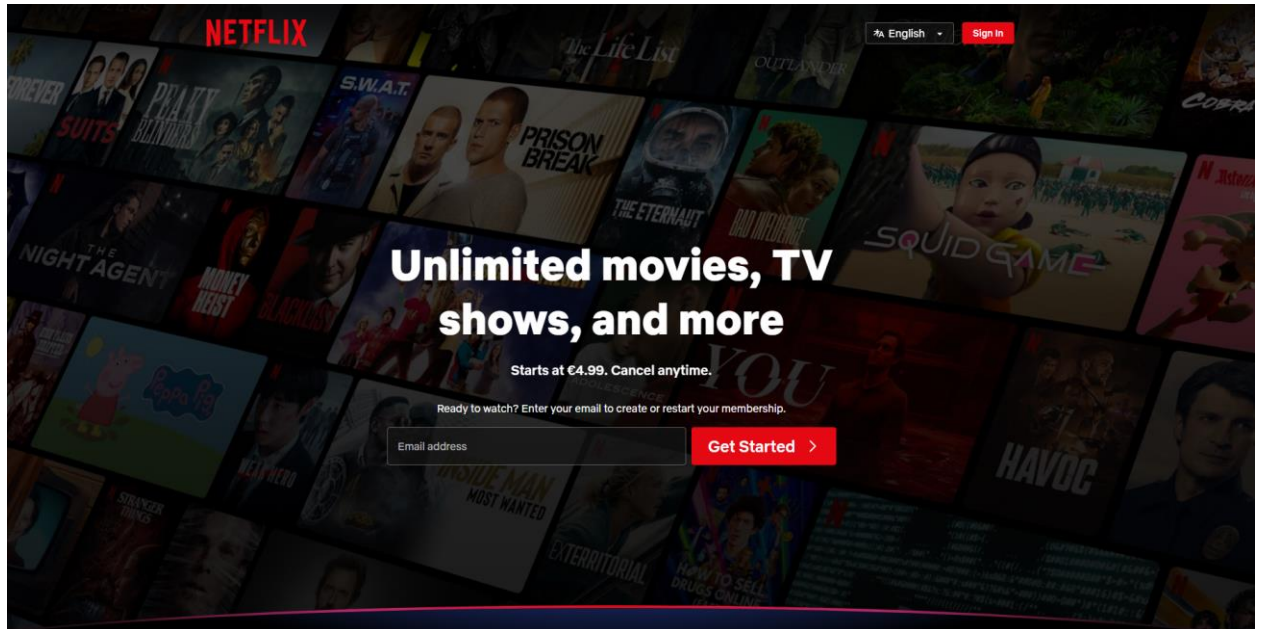


Рисунок 1.2 – Інтерфейс вебсервісу Netflix [4]

1.2.4 IMDb (Internet Movie Database)

IMDb (Internet Movie Database) – це найбільша у світі онлайн-база даних про кіно, телебачення та відеоігри, яка налічує понад 10 мільйонів назв (фільмів, серіалів, епізодів) та 12 мільйонів персоналій (акторів, режисерів, сценаристів) [5].

Запущений у 1990 році як фан-проект, IMDb швидко перетворився на авторитетний джерело інформації для кіношників і глядачів [5]. У 1998 році сервіс був придбаний Amazon, що дозволило значно розширити його функціонал [5].

IMDb – це не стримінговий сервіс у класичному розумінні, а глобальна база даних фільмів та серіалів. Вона використовується для:

- перегляду рейтингів, рецензій, акторського складу;

- збирання відгуків користувачів;
- формування списків до перегляду.

Інтерфейс вебзастосунку IMDb показано на рис. 1.3 [6].

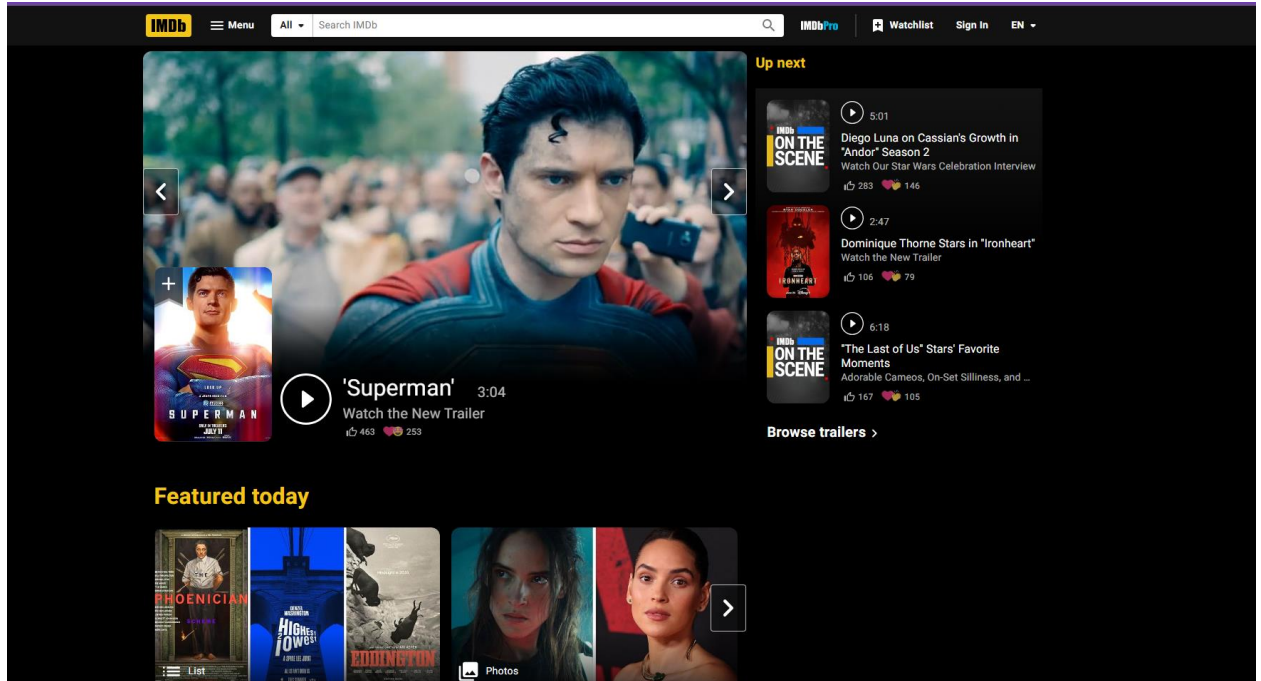


Рисунок 1.3 – Інтерфейс вебсервісу IMDb [6]

Однак IMDb не забезпечує перегляд контенту напряму (окрім IMDb TV, доступного лише в США), тому виконує допоміжну інформаційну роль [6].

1.2.5 Amazon Prime Video

Amazon Prime Video – це популярний стрімінговий сервіс від світового інтернет-гіганта Amazon, який надає доступ до величезної бібліотеки фільмів, серіалів, документальних проєктів, спортивних трансляцій та ексклюзивного оригінального контенту [7]. Цей сервіс є частиною комплексної підписки Amazon Prime, але також доступний окремо, що робить його зручним вибором для різних категорій користувачів [7].

Інтерфейс вебзастосунку Amazon Prime Video показано на рис. 1.4 [8].

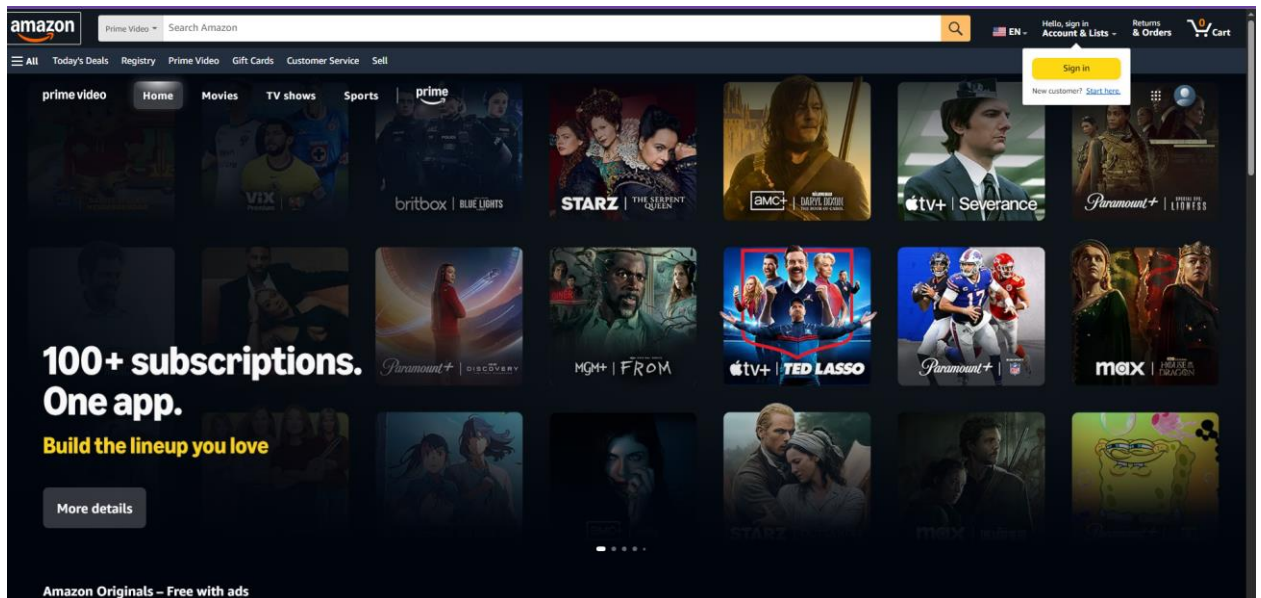


Рисунок 1.4 – Інтерфейс вебсервісу Amazon Prime Video [8]

Один із лідерів серед платних відеосервісів [8]. Пропонує:

- велику кількість преміального контенту;
- стабільний стрімінг у HD та 4K;
- підтримку кількох профілів;
- додаткові канали (HBO, Paramount+ тощо).

Недоліки:

- не завжди зручна навігація;
- недостатній переклад українською мовою;
- частина контенту вимагає окремої оплати, навіть з активною підпискою.

1.2.6 Порівняння існуючих систем, сервісів, сайтів, що надають змогу переглядати фільми і створювати власні колекції

Оцінка ключових характеристик, що впливають на вибір тієї чи іншої платформи користувачами (табл. 1.1).

Таблиця 1.1 – Порівняльна характеристика платформ для перегляду фільмів

Характеристика	Sweet.tv	Netflix	IMDb	Amazon Prime Video
Тип платформи	Онлайн-кінотеатр	Стримінговий сервіс	Онлайн-база даних про фільми	Стримінговий сервіс
Основний функціонал	Перегляд фільмів та серіалів, телебачення	Перегляд оригінальних фільмів, серіалів	Інформація про фільми, рейтинги, рецензії	Перегляд фільмів, серіалів, аренда та купівля
Наявність реєстрації	Так	Так	Необов'язкова	Так
Наявність підписок / платна основа	Платна підписка	Платна підписка	Безкоштовно, із рекламою	Платна підписка
Підтримка української мови	Так (локалізовано, дубляж)	Частково	Ні	Частково
Можливість залишати оцінки/відгуки	Так (оцінки)	Так	Так (одна з ключових функцій)	Так
Мобільний застосунок	Так	Так	Так	Так
Підтримка Smart TV	Так	Так	Ні	Так
Оригінальний контент	Ні	Так (Netflix Originals)	Ні	Так (Amazon Originals)
Рекламні вставки	Ні (у платній версії)	Ні	Так (у безкоштовній версії)	Ні
Глобальне покриття	Переважно Україна	Глобальне	Глобальне	Глобальне
Основна цільова аудиторія	Українські глядачі	Глобальна аудиторія	Кіномани, критики, дослідники	Глобальна аудиторія

У таблиці 1.2 узагальнено основні переваги та недоліки кожної з проаналізованих платформ. Це дозволяє краще зрозуміти сильні та слабкі

сторони сервісів з точки зору користувацького досвіду та доступності контенту.

Таблиця 1.2 – Переваги та недоліки платформ

Платформа	Переваги	Недоліки
Sweet.tv	Локалізований контент (український дубляж). Підтримка Smart TV. Доступна ціна підписки для України.	Обмежений вибір контенту. Немає оригінального контенту.
Netflix	Широкий каталог фільмів і серіалів. Власний оригінальний контент. Висока якість стрімінгу.	Висока вартість підписки. Часткова локалізація українською.
IMDb	Детальна інформація про фільми. Рейтинги, рецензії, акторські склади. Безкоштовний доступ.	Немає можливості перегляду відео. Містить рекламу.
Amazon Prime Video	Велика бібліотека контенту. Оригінальні серіали та фільми. Інтеграція з іншими сервісами Amazon.	Часткова локалізація. Платна підписка. Не вся бібліотека доступна в регіоні.

1.2.7 Висновки за результатами порівняння

Таким чином, аналіз існуючих рішень дозволяє зробити висновок, що жодна з платформ не є універсальною та ідеально адаптованою під вимоги сучасного користувача. Основні недоліки, які спостерігаються на більшості платформ:

- відсутність персоналізації;
- рекламне навантаження;
- низька швидкодія на слабких пристроях;
- незручність інтерфейсу або навігації.

Ці недоліки й формують передумови для створення нового інформаційного вебпорталу, орієнтованого на зручність, адаптивність і персоналізований досвід користувача.

1.3 Визначення мети та завдань системи

Мета проєкту – створити сучасний, адаптивний вебдодаток для перегляду фільмів і трейлерів з розширеним функціоналом: рейтингуванням, коментуванням, створенням персоналізованих добірок і рекомендацій на основі поведінки користувача.

Основні завдання:

- розробити структуру зручного користувацького інтерфейсу;
- реалізувати функції оцінки та коментарів;
- додати фільтрацію за жанрами, рейтингом, роком тощо;
- врахувати можливість створення нових жанрів з інтерфейсу;
- передбачити гнучку систему прав доступу (гості, користувачі, адміністратори);
- зробити платформу легкою у підтримці й масштабуванні.

Також важливо уникнути типових проблем аналогів – перевантаженого інтерфейсу, повільного завантаження, рекламного спаму та складної навігації.

1.4 Архітектурні рішення

Система розробляється із використанням мікросервісної архітектури, де окремі сервіси відповідають за:

- клієнтську частину (React-додаток);
- API (Node.js + Express);
- управління файлами (зображення, постери);
- сервіс аутентифікації та авторизації;
- майбутній email-сервіс (наприклад, повідомлення або відновлення паролю).

1.5 Ключові вимоги до системи

Функціональні вимоги:

- авторизація та реєстрація (JWT, з розподілом на ролі: гість, користувач, адміністратор);
- додавання, редагування й видалення фільмів (через адмін-панель);
- система рейтингів та коментарів (один користувач – одна оцінка, модеровані коментарі);
- фільтрація та сортування фільмів за жанрами, датою, популярністю;
- персоналізовані рекомендації (на основі оцінок або переглядів).

Нефункціональні вимоги:

- адаптивність (доступність на ПК, планшетах, мобільних пристроях, Smart TV);
- висока швидкодія (асинхронна обробка, кешування, lazy loading);
- масштабованість (додавання нових функцій без повного переписування системи);
- безпека (захист від SQL-ін'єкцій, перевірка даних на бекенді).

1.6 Висновки з розділу

У результаті аналізу предметної області були розглянуті сучасні кіносервіси – такі як Sweet.tv, Netflix, IMDb, Amazon Prime Video – за критеріями функціональності, адаптивності, зручності інтерфейсу, персоналізації та технічної реалізації. Виявлено низку типових недоліків: перевантаженість інтерфейсу, повільне завантаження, відсутність якісної локалізації, обмежений доступ до функцій у безкоштовних версіях та надмірна кількість реклами.

На основі цього сформульовано мету створення власного вебдодатку, що усуває зазначені недоліки. Визначені ключові вимоги: адаптивний

інтерфейс, підтримка авторизації, швидкодія, масштабованість і зручна навігація. Отримані результати стануть основою для подальшого проєктування системи.

2 АНАЛІЗ ПРОГРАМНИХ ЗАСОБІВ

У даному розділі розглядаються інструменти, які будуть використані при реалізації вебзастосунку: мови програмування, архітектурні рішення, програмні фреймворки та бібліотеки. Обґрунтовується доцільність їх вибору відповідно до поставлених вимог і функціоналу майбутнього продукту.

2.1 Огляд особливостей мов програмування

Для реалізації інформаційного порталу, присвяченого кіно, можна було розглядати кілька мов програмування, які активно використовуються у веброзробці. Серед них Python з фреймворками Django або Flask є популярним вибором для швидкої розробки серверної частини, особливо завдяки простоті синтаксису та великій кількості готових рішень [9]. Django надає повноцінну інфраструктуру, включаючи ORM, систему авторизації та адміністративну панель [9]. Проте недоліком Python є складність реалізації сучасного інтерактивного інтерфейсу на клієнтській стороні, що вимагає окремого frontend-стека, такого як React або Vue.

Аналогічно, PHP з Laravel міг бути використаний як надійний бекенд-фреймворк. Laravel має зручну систему маршрутизації, шаблонізацію та підтримку баз даних, однак поступово втрачає популярність серед сучасних розробників через свою менш адаптивну архітектуру [10]. Ruby з фреймворком Rails теж підходить для швидкої реалізації MVP, проте його спільнота скорочується, і він менш поширений у середовищі українських ІТ-компаній [11].

Java зі Spring Boot або C# з ASP.NET Core демонструють високу продуктивність і добре підходять для великих корпоративних проєктів [12]. Проте ці мови мають високий поріг входу [12], потребують більше часу на розгортання і не забезпечують гнучкого інтерактивного інтерфейсу без використання окремих JavaScript-фреймворків. Окрім того, хоча C# має

велику й активну спільноту розробників, вона може бути не такою великою, як спільноти навколо деяких інших мов програмування, що може вплинути на доступність бібліотек сторонніх розробників і ресурсів підтримки, керованих спільнотою [13].

На цьому тлі JavaScript виглядає найбільш збалансованим вибором [14]. Він є єдиною мовою, яка дозволяє реалізувати як клієнтську, так і серверну частину додатку – завдяки Node.js [14].

Особливу увагу заслуговує бібліотека React, створена компанією Facebook (нині Meta) у 2013 році для побудови динамічних інтерфейсів [15]. Вона забезпечує компонентний підхід до розробки, дозволяє легко управляти станом застосунку та створювати повторювані елементи, такі як списки фільмів, жанри, картки з описом [15]. React також оптимізований для створення SPA (Single Page Application), що дозволяє уникнути перезавантаження сторінок, забезпечуючи швидкий та зручний досвід для користувача [15].

Важливо й те, що React має потужну спільноту, величезну кількість готових бібліотек та якісну документацію, що значно полегшує процес розробки. Завдяки сумісності з REST API, React ідеально інтегрується з мікросервісною архітектурою, яку заплановано реалізувати в даному проєкті. Усе це робить JavaScript у зв'язці з React найбільш доцільним вибором для створення сучасного, швидкого й масштабованого вебпорталу.

При створенні інформаційного порталу для перегляду фільмів було обрано стек технологій, який передбачає використання мови JavaScript як для клієнтської, так і для серверної частини. Також використовується SQL для взаємодії з базою даних. Нижче наведено порівняльну таблицю основних мов програмування, які розглядалися на етапі проектування системи:

В таблиці 2.1 наведено порівняння мов програмування.

Таблиця 2.1 – Порівняння мов програмування

Мова / Технологія	Сфера застосування	Переваги	Недоліки	Висновок
1	2	3	4	5
JavaScript + React	Frontend / Full-stack	Єдина мова для клієнта й сервера (через Node.js), SPA, велика спільнота, адаптивність, компонентний підхід	Потребує налаштувань для SSR (якщо потрібно), залежність від бібліотек	Оптимальний вибір для динамічного, інтерактивного та SPA-застосунку
Python + Django	Backend	Простота, вбудована адмін-панель, ORM, безпека, швидка розробка	Слабкий frontend, необхідність окремого фреймворку для інтерфейсу	Підходить для бекенду, але потребує окремого frontend-стеку
PHP + Laravel	Backend / Full-stack	Швидкий старт, Blade-шаблони, велика кількість ресурсів	Менша гнучкість інтерфейсу, застарілість підходів у деяких проектах	Можна використовувати, але менш гнучко для SPA
Ruby on Rails	Backend / Full-stack	Швидка розробка MVP, зрозумілий синтаксис	Менша популярність, менша підтримка в Україні	Не рекомендується через зменшення спільноти
Java + Spring Boot	Backend (Enterprise)	Потужність, надійність, масштабованість	Високий поріг входу, складність інтеграції з frontend	Добрий вибір для великих систем, але надлишковий для невеликого порталу

Продовження таблиці 2.1

1	2	3	4	5
C# + ASP.NET Core	Backend (Enterprise)	Висока продуктивність, безпека, підтримка Microsoft	Складність хостингу на Linux, великий об'єм системи	Хороший варіант, але перевантажений для SPA-порталу

2.2 Огляд особливостей обраних платформ і бібліотек

JavaScript – одна з найпоширеніших мов програмування у світі, яка була створена ще в 1995 році для надання динамічності вебсторінкам [16]. З роками вона значно розширила свої можливості, і сьогодні використовується не лише для клієнтської частини (frontend), але й для серверної (backend) завдяки Node.js. Саме універсальність JavaScript стала ключовим фактором у виборі технологій для створення вебпорталу, адже вона дозволяє реалізовувати повноцінні застосунки у єдиному мовному середовищі – як на клієнті, так і на сервері [16].

У свою чергу, React – це JavaScript-бібліотека з відкритим кодом, яка була розроблена в компанії Meta (раніше – Facebook) у 2013 році [17]. Первинною метою її створення було забезпечення гнучкого та ефективного рендерингу компонентів у великих інтерфейсах Facebook та Instagram [17]. З того часу React став стандартом де-факто у frontend-розробці. Завдяки своїй компонентній моделі, він дозволяє створювати багаторазові елементи інтерфейсу, що значно спрощує розробку великих динамічних застосунків [17]. Саме це є особливо важливим для медіа-платформи, де повторювані елементи – такі як картки фільмів, списки, коментарі – є основою структури.

Також важливо відзначити, що React підтримує концепцію SPA (Single Page Application) – односторінкових додатків, у яких зміна контенту відбувається без повного перезавантаження сторінки [17]. Це забезпечує

швидко, плавну й зручну взаємодію користувача з інтерфейсом, що є критично важливим для платформ, орієнтованих на велику кількість дій без перешкод – перегляд трейлерів, сортування, фільтрація, додавання в обране, тощо.

Додаткові переваги React, що обумовили його вибір:

- велика спільнота розробників і постійна підтримка з боку Meta. Це гарантує стабільність технології, регулярне оновлення й доступність великої кількості бібліотек, які пришвидшують розробку;
- велика база знань та документації, що суттєво знижує поріг входу й дає змогу швидко вирішувати технічні труднощі під час розробки;
- гарна інтеграція з REST API, що дозволяє легко обмінюватися даними з мікросервісами та серверною частиною;
- підтримка адаптивного дизайну за допомогою бібліотек стилізації та гнучкого підходу до верстки.

Таким чином, обрання JavaScript і React є обґрунтованим як з технічної, так і з економічної точки зору. Вони забезпечують необхідну швидкодію, масштабованість, гнучкість і зручність для реалізації сучасного інформаційного порталу з кіноконтентом.

2.3 Інструменти для роботи з базами даних та API

Розробка сучасного вебзастосунку вимагає використання надійних, гнучких та перевірених рішень для роботи з базами даних і серверною логікою. У рамках реалізації інформаційного порталу, присвяченого кінематографу, ключовими інструментами стали PostgreSQL як система керування базами даних, pgAdmin як засіб адміністрування й управління СКБД, а також Postman – платформа для тестування REST API. Усі ці компоненти забезпечують надійний технологічний стек, який дозволяє ефективно реалізувати як серверну логіку, так і взаємодію між клієнтом і сервером у межах обраної мікросервісної архітектури.

2.3.1 PostgreSQL – сучасна система керування базами даних

PostgreSQL – це потужна об'єктно-реляційна система керування базами даних з відкритим вихідним кодом, яка розвивається з 1986 року в рамках проєкту POSTGRES Каліфорнійського університету в Берклі [18]. Сьогодні PostgreSQL є однією з найпопулярніших і найбільш надійних СКБД у світі. Вона підтримує як традиційні реляційні структури, так і роботу з об'єктно-орієнтованими та документно-орієнтованими даними завдяки підтримці JSONB, XML, hstore тощо.

У дипломному проєкті PostgreSQL виконує роль основного сховища даних. Зокрема, у базі зберігаються:

- інформація про користувачів (реєстрація, ролі, вподобання);
- метадані про фільми (назва, рік, країна, опис, жанр);
- оцінки, коментарі, історія перегляду;
- зв'язки між фільмами та жанрами;
- дані для адміністрування контенту.

Основними перевагами PostgreSQL у рамках цього проєкту є [19]:

- підтримка ACID-властивостей, що гарантує цілісність даних навіть у разі збою системи;
- широкі можливості індексації (btree, hash, GiST, GIN, BRIN), які забезпечують високу продуктивність при запитах до великих обсягів інформації;
- масштабованість, що дозволяє в подальшому розширювати обсяги даних або кількість користувачів;
- гнучка робота з типами даних, включаючи власні типи, масиви, таблиці записів та JSON.

Крім того, PostgreSQL підтримує тригери, збережені процедури (на PL/pgSQL, PL/Python тощо), що дає змогу реалізовувати частину логіки прямо в базі [19].

2.3.2 pgAdmin – інтерфейс для візуального керування базою

PgAdmin є офіційним графічним клієнтом для адміністрування PostgreSQL. Це повноцінне вебзастосування, що дозволяє розробникам, адміністраторам баз даних та аналітикам зручно взаємодіяти з базами без необхідності постійного використання командного рядка [20].

У рамках реалізації інформаційного порталу pgAdmin активно використовувався для:

- створення й редагування бази даних – таблиць, схем, зв'язків;
- візуалізації структури БД – перегляд зв'язків «один-до-багатьох», «багато-до-багатьох»;
- моніторингу активності – журналів, процесів, транзакцій;
- налаштування користувачів – додавання ролей, прав доступу;
- виконання та збереження SQL-запитів – що дозволяло оперативно тестувати коректність запитів під час розробки;
- резервного копіювання – створення дампів і відновлення даних на локальній машині.

Приклад роботи з pgAdmin, який демонструє рис. 2.1.

Особливою перевагою pgAdmin є можливість поєднувати роботу з кількома серверами одночасно, що дозволяє розгорнути середовище для розробки, тестування й продакшну [20]. Також pgAdmin підтримує зручну побудову ER-діаграм і дозволяє бачити структуру бази в ієрархічному вигляді, що важливо для візуального проєктування та навігації [20].

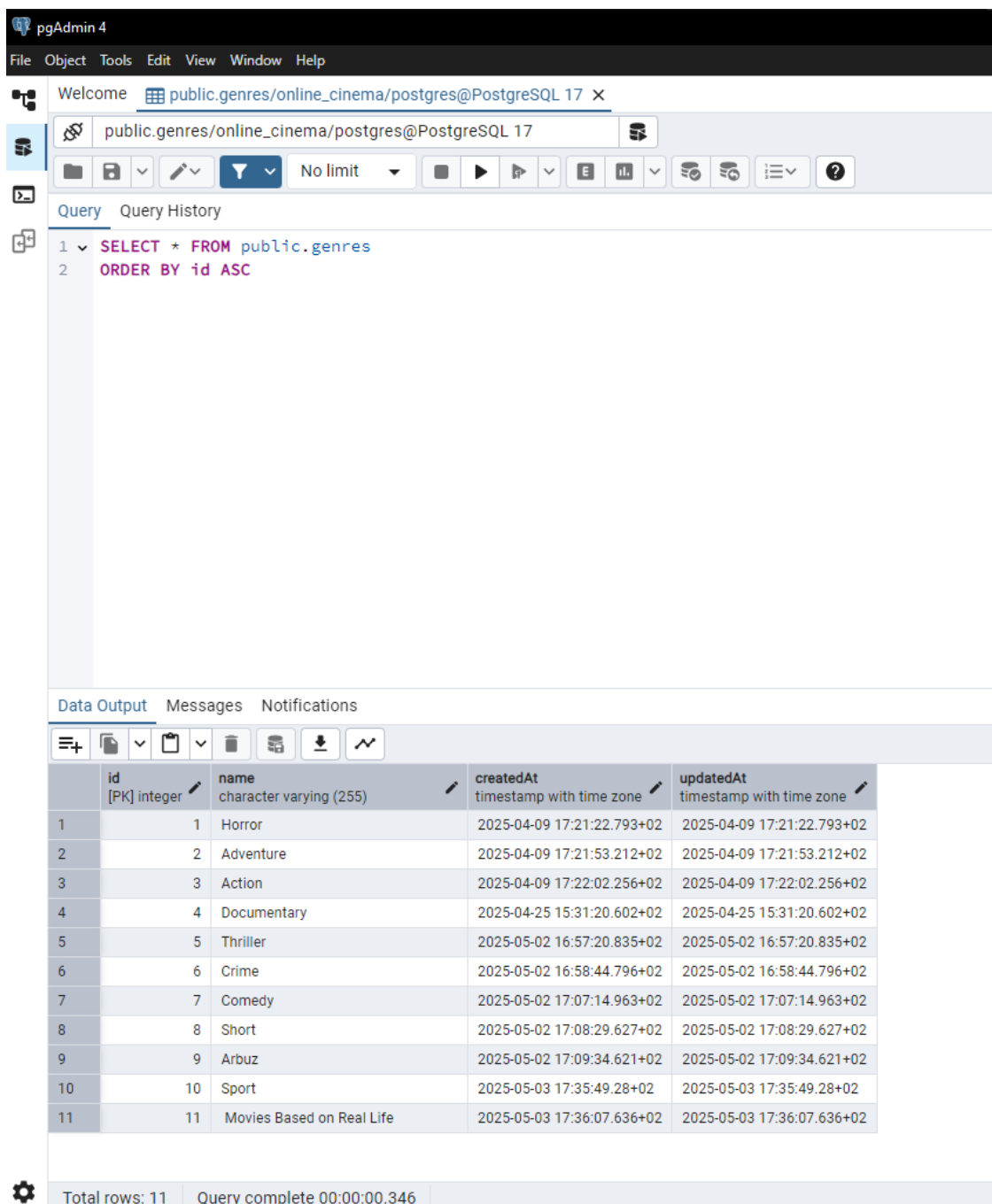


Рисунок 2.1 – Приклад роботи з pgAdmin [20]

2.3.3 Postman – платформа для тестування API-запитів

Postman – це інструмент для тестування API, який дозволяє виконувати HTTP-запити до серверів і перевіряти відповідь, що повертається. Його активно використовують у всіх етапах життєвого циклу API – від створення та налагодження до тестування й документації [21].

У контексті даного проєкту, де React-фронтенд взаємодіє з бекендом через REST API, Postman виконував критичну роль при:

- створенні й перевірці запитів до бекенду: запити до ендпоінтів для отримання списку фільмів, фільтрації за жанрами, реєстрації/авторизації користувачів;
- тестуванні помилок: перевірка коректності обробки винятків (404, 401, 403, 500);
- перевірці логіки авторизації: використання токенів авторизації, перевірка доступу до закритих маршрутів;
- документуванні API: створення колекцій і опису до кожного запиту;
- автоматизованому тестуванні: написання простих скриптів у вкладках “Tests” для перевірки очікуваної структури відповіді.

Приклад роботи з API-клієнтом Postman, який демонструє процес аутентифікації користувача через REST API на рис. 2.2.

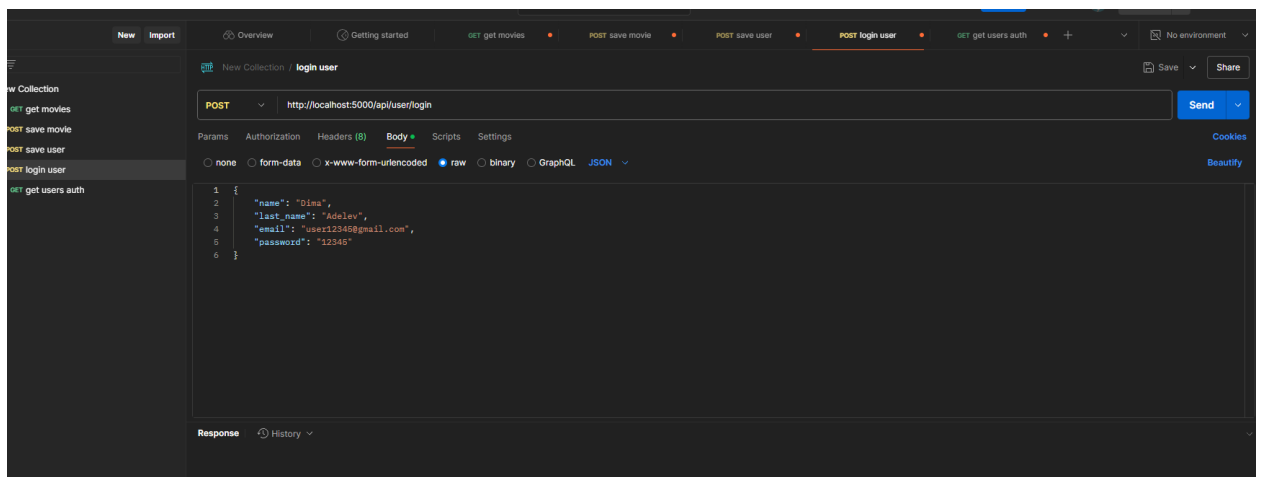


Рисунок 2.2 – Приклад роботи з Postman [21]

Postman дозволяє зберігати запити у вигляді наборів (collections), що значно пришвидшує перевірку системи при будь-яких змінах на боці API. Для дипломного проєкту це стало особливо корисним при додаванні нових

функцій: замість повторного написання запитів – достатньо оновити параметри у збереженому запиті.

2.3.4 Проєктування структури бази даних у draw.io

Для візуального проєктування структури бази даних було використано інструмент draw.io – популярний онлайн-редактор для створення схем і діаграм [22]. Цей сервіс дав змогу чітко спланувати архітектуру таблиць, їхні зв'язки та атрибути ще до реалізації на рівні SQL.

На основі вимог до функціональності порталу було створено модель бази даних, що враховує зберігання інформації про фільми, користувачів, жанри, коментарі та рейтинги. Особливу увагу було приділено правильній організації зв'язків між сутностями та уникненню надлишковості даних.

Було реалізовано такі основні таблиці:

- movie – зберігає інформацію про фільми: назва, рік, країна, режисер, актори, тривалість, постер, трейлер, опис тощо;
- genre та movie_genres – реалізують зв'язок багато до багатьох, оскільки фільм може мати кілька жанрів;
- user – містить дані зареєстрованих користувачів, включно з роллю для реалізації авторизації та модерації;
- comment – дозволяє зберігати коментарі користувачів до фільмів із зазначенням дати створення;
- rating – окрема таблиця для зберігання оцінок, що забезпечує можливість кожному користувачеві оцінити фільм лише один раз.

Структура бази даних для кінопорталу, яка включає таблиці фільмів, користувачів, жанрів, коментарів та оцінок, а також зв'язки між ними, демонструє логіку збереження інформації та взаємодії між основними об'єктами системи, зображено на рис. 2.3.

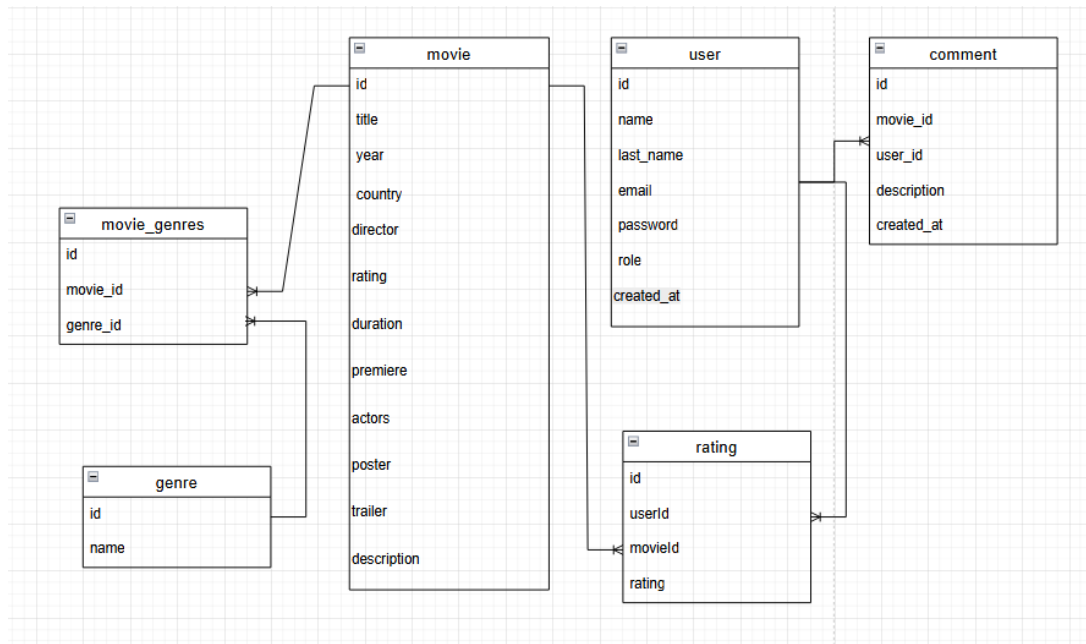


Рисунок 2.3 – Структура бази даних

Використання draw.io дозволило швидко та наочно спроектувати структуру, перевірити логічні зв'язки між таблицями, а також підготуватися до подальшої реалізації бази даних у системі PostgreSQL. Уся модель відповідає принципам нормалізації, що сприяє підтримуваності та масштабованості проєкту.

2.3.5 Сумісна робота інструментів

Інтеграція PostgreSQL, pgAdmin, Postman та draw.io створює єдину ефективну екосистему для роботи з даними на всіх етапах – від проєктування бази даних до реалізації бізнес-логіки та тестування API.

На початковому етапі для візуального проєктування структури БД використовувався сервіс draw.io, який дав змогу графічно змодельювати таблиці, їх атрибути та зв'язки. Це допомогло наочно оцінити логіку системи та уникнути помилок ще до реалізації SQL-схеми.

Після затвердження структури база даних була реалізована в PostgreSQL, а для керування й адміністрування використовувався інтерфейс

pgAdmin, який забезпечує зручну навігацію по таблицях, перегляд вмісту та виконання SQL-запитів.

На завершальному етапі для перевірки роботи API-запитів і коректності взаємодії з базою даних застосовувався Postman, який дозволяє тестувати маршрути, перевіряти отримання, створення, оновлення та видалення даних.

Таким чином, узгоджене використання цих інструментів забезпечило чітке проєктування, стабільну реалізацію та ефективне тестування всієї системи.

3 ОСНОВНІ РІШЕННЯ З РЕАЛІЗАЦІЇ СИСТЕМИ

У цьому розділі розглянуто ключові рішення, що були прийняті під час розробки інформаційного порталу, присвяченого кінематографу. Було визначено засоби реалізації, сформульовано функціональні вимоги, спроектовано структуру бази даних, а також побудовано загальну архітектуру програмного забезпечення, відповідно до обраних технологій.

3.1 Вибір засобів реалізації

Під час реалізації інформаційного порталу, присвяченого кіно, було ухвалено низку ключових рішень щодо використання технологій і засобів розробки. Основним критерієм при виборі технологічного стеку стали зручність розробки, гнучкість, розширюваність, швидкість завантаження, підтримка модульності, а також наявність великої спільноти користувачів та документації.

Для фронтенду було обрано React – одну з найпопулярніших бібліотек JavaScript для створення інтерфейсів користувача. React забезпечує компонентно-орієнтований підхід до розробки, що дозволяє легко масштабувати систему. Також React добре підходить для побудови SPA (Single Page Application), що є оптимальним рішенням для інтерактивного кінопорталу.

Для зберігання даних у базі було вирішено використовувати PostgreSQL – потужну об'єктно-реляційну систему управління базами даних з відкритим вихідним кодом. PostgreSQL забезпечує високу продуктивність, надійність і масштабованість, що робить її ідеальним вибором для вебзастосунків та інформаційних порталів із складною структурою даних.

В якості середовища виконання JavaScript обрано Node.js, а для створення REST API – Express.js, що дозволяє швидко та ефективно будувати серверну частину проєкту. Цей стек забезпечує зручну комунікацію між

клієнтом і сервером, дозволяючи реалізувати усі необхідні запити до бази даних.

Для керування залежностями використано прт. Для тестування запитів – Postman. Для розміщення зображень – зберігання реалізовано локально на сервері, що спрощує доступ до них через URL.

3.2 Функціональні вимоги до системи

Система повинна реалізовувати наступні функціональні можливості:

- а) реєстрація та авторизація користувачів:
 - 1) забезпечення безпечного зберігання паролів;
 - 2) валідація унікальності email;
 - 3) надання ролей (USER, ADMIN) з відповідними правами доступу;
- б) перегляд списку фільмів та детальної інформації про кожен фільм:
 - 1) виведення заголовку, року, тривалості, опису, трейлеру, постера, акторського складу;
 - 2) відображення середньої оцінки та жанрів;
- в) пошук і фільтрація фільмів:
 - 1) за назвою, жанром, роком виходу, країною, режисером;
- г) система оцінювання:
 - 1) користувачі можуть оцінювати фільми по шкалі (наприклад, від 1 до 10);
 - 2) автоматичне обчислення середньої оцінки;
- д) система коментування:
 - 1) користувачі можуть залишати текстові коментарі до фільмів;
 - 2) адміністратор має змогу модерації (видалення) неприйнятнього контенту;
- е) адмін-панель:
 - 1) можливість додавання, редагування та видалення фільмів;

- 2) можливість керування жанрами;
- 3) завантаження постера й трейлера;
- ж) система рекомендацій (базова реалізація):
 - 1) виведення схожих фільмів за жанрами чи рейтингом.

3.3 Розробка бази даних

Для реалізації функціоналу інформаційного порталу про кіно було спроектовано реляційну базу даних за допомогою ORM Sequelize (JavaScript). Усі таблиці мають чітко визначені атрибути, типи даних, обов'язковість полів і ключові зв'язки. Нижче наведено опис кожної таблиці бази даних.

База даних є основою функціонування інформаційного порталу. У межах цього проєкту реалізовано нормалізовану структуру даних, що дозволяє ефективно зберігати, оновлювати та запитувати інформацію.

Основні таблиці:

а) `user` – зберігає інформацію про користувачів. Обов'язкові поля – `email`, `password`, а також `role` (за замовчуванням – `USER`). `name` та `surname` – необов'язкові для реєстрації, але можуть використовуватися для відображення в інтерфейсі чи коментарях;

б) `movie` – основна таблиця з інформацією про фільми. Ключові атрибути:

- 1) `title` – унікальний заголовок;
- 2) `description`, `poster`, `trailer` – забезпечують наповнення картки фільму;
- 3) `rating` – оновлюється автоматично на основі оцінок користувачів;

в) `genre` – окрема таблиця для жанрів, що дозволяє розширювати список без дублювання;

г) `movie_genres` – проміжна таблиця для реалізації зв'язку "багато-до-багатьох" між фільмами й жанрами. Один фільм може мати кілька жанрів, і один жанр може належати багатьом фільмам;

д) `comment` – зберігає коментарі користувачів до фільмів. Присутні зовнішні ключі до таблиць `user` та `movie`, що забезпечує відстеження, хто і до якого фільму залишив відгук;

е) `rating` – таблиця для зберігання оцінок користувачів. Для запобігання дублюванню оцінок передбачається логіка перевірки унікальності (наприклад, один користувач може оцінити фільм лише раз, але може оновити свою оцінку).

В таблиці 3.1 наведено атрибути таблиці `user`.

Таблиця 3.1 – Атрибути таблиці `user`

Атрибут	Тип даних	Обов'язковий	Опис
<code>id</code>	INTEGER	так	Первинний ключ, автоінкремент
<code>name</code>	STRING	ні	Ім'я користувача
<code>last_name</code>	STRING	ні	Прізвище користувача
<code>email</code>	STRING	так	Унікальний email
<code>password</code>	STRING	так	Пароль
<code>role</code>	STRING	ні	Роль (USER або ADMIN), за замовчуванням USER

В таблиці 3.2 наведено атрибути таблиці `movie`.

Таблиця 3.2 – Атрибути таблиці `movie`

Атрибут	Тип даних	Обов'язковий	Опис
<code>id</code>	INTEGER	так	Первинний ключ, автоінкремент
<code>title</code>	STRING	так	Назва фільму (унікальна)
<code>year</code>	INTEGER	так	Рік виходу
<code>country</code>	STRING	ні	Країна
<code>director</code>	STRING	ні	Режисер
<code>rating</code>	FLOAT	ні	Середня оцінка, за замовчуванням 0
<code>duration</code>	INTEGER	так	Тривалість у хвилинах

Продовження таблиці 3.2

Атрибут	Тип даних	Обов'язковий	Опис
premier	DATE	ні	Дата прем'єри
actors	STRING	ні	Список акторів
poster	STRING	так	Посилання на постер
trailer	STRING	так	Посилання на трейлер
description	TEXT	так	Опис фільму

В таблиці 3.3 наведено атрибути таблиці genre.

Таблиця 3.3 – Атрибути таблиці genre

Атрибут	Тип даних	Обов'язковий	Опис
id	INTEGER	так	Первинний ключ, автоінкремент
name	STRING	так	Назва жанру (унікальна)

В таблиці 3.4 наведено атрибути таблиці movie_genres.

Таблиця 3.4 – Атрибути таблиці movie_genres

Атрибут	Тип даних	Обов'язковий	Опис
id	INTEGER	так	Первинний ключ, автоінкремент
movie_id	INTEGER	так	Зовнішній ключ до movie
genre_id	INTEGER	так	Зовнішній ключ до genre

Ця таблиця є проміжною для зв'язку "багато-до-багатьох" між фільмами та жанрами.

В таблиці 3.5 наведено атрибути таблиці comment.

Таблиця 3.5 – Атрибути таблиці comment

Атрибут	Тип даних	Обов'язковий	Опис
id	INTEGER	так	Первинний ключ, автоінкремент
user_id	INTEGER	так	Зовнішній ключ до user
movie_id	INTEGER	так	Зовнішній ключ до movie
description	TEXT	так	Текст коментаря

В таблиці 3.6 наведено атрибути таблиці rating.

Таблиця 3.6 – Атрибути таблиці rating

Атрибут	Тип даних	Обов'язковий	Опис
id	INTEGER	так	Первинний ключ, автоінкремент
userId	INTEGER	так	Зовнішній ключ до user
movieId	INTEGER	так	Зовнішній ключ до movie
rating	INTEGER	так	Оцінка (1–10, наприклад)

Завдяки чітко визначеним типам і зв'язкам ця база даних дозволяє гнучко керувати інформацією про фільми, користувачів, жанри, оцінки та коментарі. Уся логіка була реалізована за допомогою ORM Sequelize, що дозволяє взаємодіяти з базою без написання сирих SQL-запитів.

3.4 Структура програмного забезпечення

ПЗ поділено на два основні модулі: фронтенд (інтерфейс користувача) та бекенд (серверна логіка та API). Такий підхід дозволяє розвивати та обслуговувати кожен компонент окремо:

а) фронтенд:

- 1) реалізовано за допомогою React;
- 2) компоненти розділені за функціональністю: Header, Footer, MovieCard, MovieList, MovieDetails, Rating, Comment, AdminPanel, Login, Register, тощо;
- 3) впроваджено систему маршрутизації (React Router);
- 4) використано контекст (Context API) для зберігання даних авторизації та ролі користувача;
- 5) дані отримуються через REST API;

б) бекенд:

- 1) реалізовано на Node.js із використанням Express.js;
- 2) роутери поділені за ресурсами: /api/user, /api/movie, /api/genre, /api/comment, /api/rating;

- 3) створено моделі для взаємодії з базою даних;
- 4) реалізовано middleware для аутентифікації (JWT) та авторизації (перевірка ролей);
- в) завантаження контенту:
 - 1) пости зберігаються у локальній директорії сервера, а шляхи до них записуються в БД;
 - 2) трейлери –посилання на YouTube;
- г) безпека:
 - 1) хешування паролів;
 - 2) перевірка автентичності запитів користувача через JWT-токени;
 - 3) захист від SQL-ін'єкцій і XSS;
- д) масштабованість:
 - 1) можливість у майбутньому винести бекенд на окремий сервер;
 - 2) можливість переходу на повноцінну СУБД у разі зростання обсягів даних.

В таблиці 3.7 наведено структуру проєкту інформаційного порталу, присвяченого кіно, за каталогами та файлами.

Таблиця 3.7 – Структура проєкту за каталогами та файлами

Назва	Опис
AppContent/	Містить основний вміст сторінок додатку. Відповідає за компонування сторінки: заголовки, навігацію, текстові блоки та інші інформаційні частини.
ArticleFooter/	Модуль для виводу футера в рамках статей або інформаційних блоків — використовується для завершення описів.

Продовження таблиці 3.7

Назва	Опис
ArticleNavigation/	Відповідає за перемикання між розділами контенту або сторінками статей. Реалізує локальну навігацію.
ArticlePages/	Містить компоненти окремих сторінок (статей, публікацій, блоків). Зручно для динамічної генерації сторінок.
assets/	Зберігає статичні ресурси: зображення, іконки, SVG, шрифти тощо. Імпортуються в компоненти.
components/	Містить багаторазові, незалежні інтерфейсні компоненти.
components/App/	Головний компонент застосунку, що ініціалізує маршрути, обгортки (напр., контексти).
components/MoviesCarousel/	Компонент слайдера для перегляду фільмів (новинки, рекомендації).
components/MoviesList/	Відображає список фільмів з фільтрацією, сортуванням, взаємодією.
components/PageTitle/	Відповідає за уніфіковане відображення заголовків сторінок.
fixtures/	Тестові дані, що використовуються під час розробки (фільми, коментарі тощо).
Footer/	Глобальний футер додатку з контактами, логотипом, посиланнями.
Header/	Шапка сайту з логотипом, навігацією, пошуком та елементами користувача.
hooks/	Користувацькі React-хуки (API-запити, збереження стану, логіка форм тощо).
http/	Містить логіку взаємодії з API (налаштування запитів, окремі сервіси).
pages/	Відповідає за сторінки додатку, які відповідають окремим маршрутам (головна, фільм, профіль тощо).
index.css	Головний файл стилів: шрифти, кольори, адаптивність.
index.js	Точка входу застосунку. Налаштування ReactDOM, маршрутизації, стилів та контекстів.

3.5 Висновок до розділу

У цьому розділі було детально розглянуто ключові аспекти реалізації інформаційного порталу, присвяченого кіно. Здійснено обґрунтований вибір технологій, зокрема використання стеку JavaScript з React для фронтенду та PostgreSQL для зберігання даних. Такий підхід забезпечує високу гнучкість, масштабованість і зручність у розробці.

Сформульовано функціональні вимоги до системи, що охоплюють основні сценарії взаємодії користувача з платформою: перегляд фільмів і трейлерів, авторизацію, коментування, оцінювання, а також можливості адміністративного управління контентом. Це дозволяє створити повноцінну екосистему для взаємодії з кіноконтентом.

Було спроектовано логічну структуру бази даних із врахуванням зв'язків між сутностями, що забезпечує ефективне зберігання та обробку інформації. Також описано файлову структуру проєкту, яка відповідає сучасним підходам до організації компонентів у React-додатках, що сприяє підтримуваності та розширюваності системи.

Таким чином, прийняті рішення щодо вибору засобів реалізації, структури даних та архітектури програмного забезпечення формують надійну основу для подальшої розробки, тестування та впровадження системи.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРЕМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

У цьому розділі наведено особливості експлуатації створеного інформаційного порталу про кіно, розглянуто процеси тестування функціоналу, виявлення помилок, а також експериментальну перевірку ефективності розробленої системи.

4.1 Характеристики програми

У розробленому вебзастосунку було реалізовано усі потрібні складові для коректної роботи сервісу. Основні серед них є наступні:

- реєстрація та авторизація користувачів;
- можливість додавання коментарів ;
- перегляд списку фільмів;
- перегляд додаткової інформації про фільм;
- оцінювання фільмів;
- додавання фільму адміністратором;
- видалення фільму адміністратором;
- фільтрація фільмів.

4.2 Застосування програми

Система призначена для перегляду, завантаження фільмів, постерів та коментування. Основна направленість це звичайні користувачі і адміністратори.

4.2.1 Сценарії використання (Use Case діаграми)

На рисунку 4.1 зображено діаграму прецеденту «Створення облікового запису».

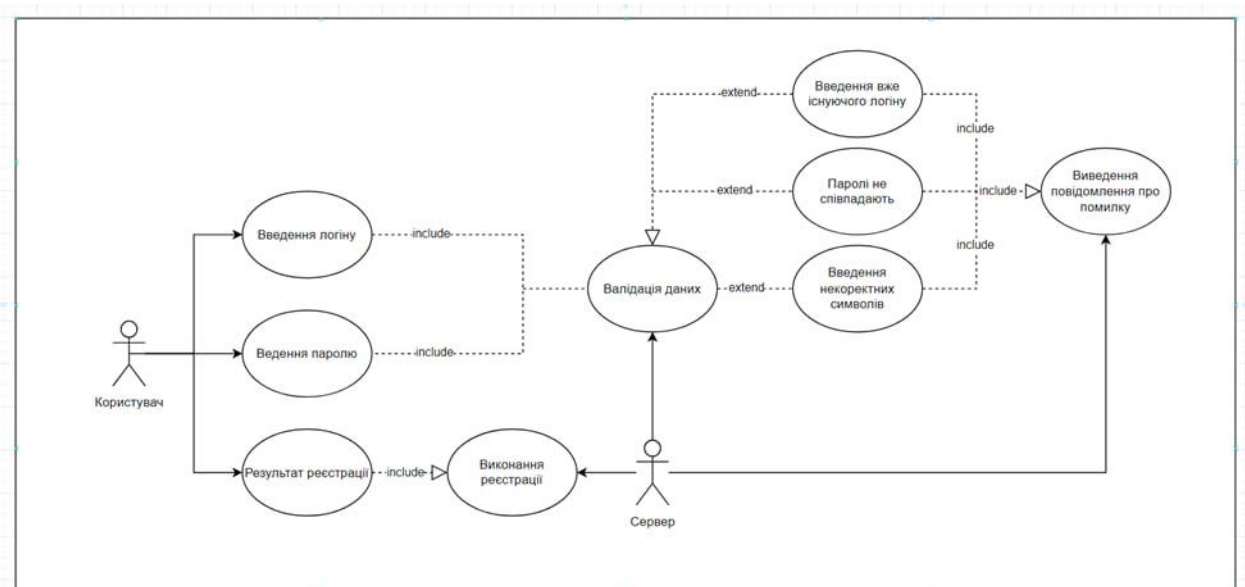


Рисунок 4.1– Схема прецеденту «Створення облікового запису»

Потік подій для розробленої діаграми прецеденту «Створення облікового запису».

Передумова: доступ до мережі інтернет, користувач увійшов у додаток.

Основний виконавець – авторизований користувач.

Основний потік:

а) введення логіну:

1) валідація даних. У разі, якщо користувач ввів вже існуючий логін – А1. Введення вже існуючого логіну;

2) валідація даних. У разі, якщо користувач ввів некоректні символи – А3. Введення некоректних символів;

б) введення паролю:

1) валідація даних. У разі, якщо перший та другий паролі користувача не співпадають – А2. Паролі не співпадають;

2) валідація даних. У разі, якщо користувач ввів некоректні символи – А3. Введення некоректних символів;

в) результат реєстрації:

1) успішне виконання реєстрації.

Альтернативний потік:

а) А1. Введення вже існуючого логіну:

1) користувач отримує сповіщення про невдалу спробу реєстрації;

2) потік повертається до пункту 1 основного потоку;

б) А2. Паролі не співпадають:

1) користувач отримує сповіщення про невдалу спробу реєстрації;

2) потік повертається до пункту 2 основного потоку;

в) А3. Введення некоректних символів:

1) користувач отримує сповіщення про невдалу спробу реєстрації;

2) потік повертається до певного пункту, в залежності в якому була помилка, якщо у логіну, то до пункту 1, у паролі – 2.

Постумова: успішна реєстрація.

На рисунку 4.2 зображено діаграму прецеденту «Оцінювання фільму».

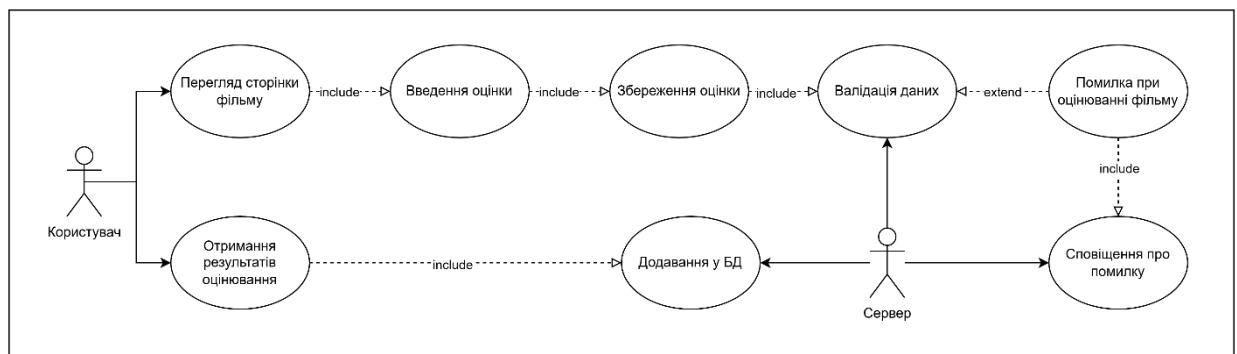


Рисунок 4.2– Схема прецеденту «Оцінювання фільму»

Потік подій для розробленої діаграми прецеденту «Оцінювання фільму».

Передумова: доступ до мережі інтернет, користувач увійшов у додаток. Користувач відкрив сторінку фільму. Основний виконавець – авторизований користувач.

Основний потік:

а) вибір фільму:

1) валідація даних. У разі помилки при оцінюванні фільму – А1.

Помилка при оцінюванні фільму;

б) отримання результатів оцінювання:

1) успішне додавання у БД.

Альтернативний потік:

а) А1. Помилка при оцінюванні фільму:

1) користувач отримує сповіщення про невдале оцінювання фільму;

2) потік повертається до пункту 1 основного потоку.

Постумова: отримання результатів оцінювання.

4.2.2 Діаграма бізнес-процесу

На рисунку 4.3 зображено діаграму бізнес-процесу створення облікового запису, яка відображає ключові етапи взаємодії користувача з системою, включаючи введення даних та їх валідацію.

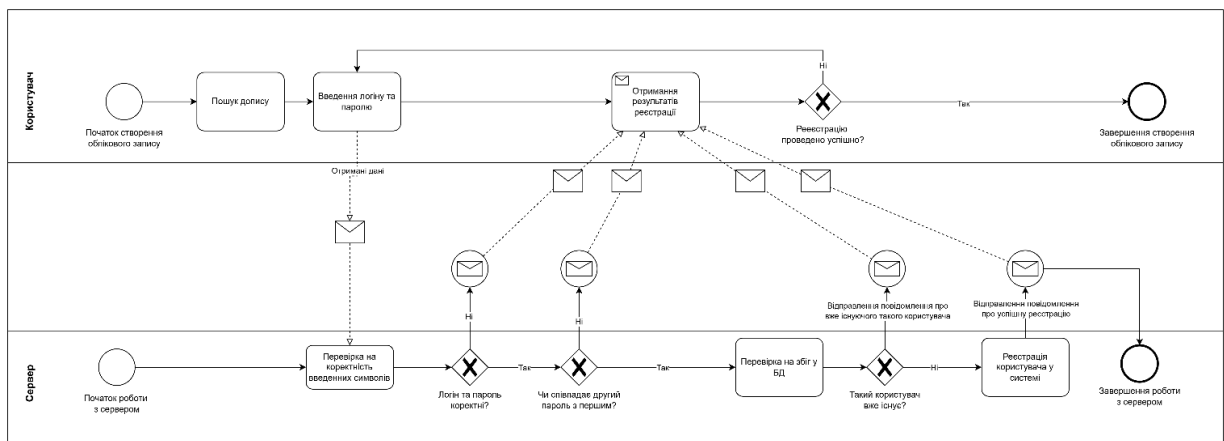


Рисунок 4.3– Діаграма бізнес-процесу «створення облікового запису»

4.3 Тестування програми

У таблиці 4.1 наведено чек-ліст перевірки функціоналу вебзастосунку на трьох різних браузерях: Google Chrome (версія 128.0.6613.138), Microsoft Edge (версія 129.0.2792.79) та Opera GX (версія 112.0.5197.60). Метою тестування було переконатися в коректності роботи ключових функцій порталу незалежно від середовища перегляду.

Перевірка охоплює основні дії користувача, зокрема: реєстрацію, авторизацію, пошук фільмів, взаємодію з коментарями та адмін-панеллю. Для кожної функції зафіксовано результат проходження тесту (успішний – *Passed*) в кожному з браузерів.

Таким чином, таблиця демонструє, що всі основні функціональні можливості застосунку стабільно працюють у найпопулярніших браузерах, що підтверджує кросбраузерну сумісність системи.

Таблиця 4.1 – Чек-ліст до функціоналу

Сайт	Браузер версія: Google Chrome 128.0.6613.138	Браузер версія: Microsoft Edge 129.0.2792.79	Браузер версія: Opera GX 112.0.5197.60
1	2	3	4
Реєстрація та Особистий профіль			
Перевірити реєстрацію на сайті	Passed	Passed	Passed
Валідаційні повідомлення для обов'язкових полів:			
Порожні поля	Passed	Passed	Passed
Пароль, який містить менше 6 символів	Passed	Passed	Passed

Продовження таблиці 4.1

1	2	3	4
Заповнення полів валідними даними	Passed	Passed	Passed
Авторизація			
Повторний вхід після реєстрації	Passed	Passed	Passed
Валідаційні повідомлення:			
Невалідна пошта (неzareєстрований користувач)	Passed	Passed	Passed
Невалідний пароль	Passed	Passed	Passed
Пошук			
Перевірити роботу пошуку за назвами фільмів	Passed	Passed	Passed
Перевірити перехід за підказками у випадіючому списку поля пошуку після введення ключового слова	Passed	Passed	Passed
Перевірити роботу пошуку по різноманітним параметрам	Passed	Passed	Passed
Перевірити сортування товарів за назвою	Passed	Passed	Passed
Перевірити фільтрування за категоріями	Passed	Passed	Passed
Коментарі			
Перевірити додавання коментаря-відгуку до фільму	Passed	Passed	Passed
Перевірити наявність порожніх полів	Passed	Passed	Passed
Адмін панель			
Перевірити додавання нового жанру	Passed	Passed	Passed
Перевірити видалення фільму	Passed	Passed	Passed
Перевірити коректне додавання постеру	Passed	Passed	Passed

4.4 Структура програми

Розроблений вебзастосунок побудований за компонентною структурою з використанням бібліотеки React. Кожна окрема частина інтерфейсу реалізована у вигляді окремого компонента, що забезпечує зручність у підтримці та масштабуванні проєкту.

На рисунку 4.4 зображено спрощену логіку функціонування вебзастосунку, яка ілюструє основні етапи переміщення користувача між сторінками програми.

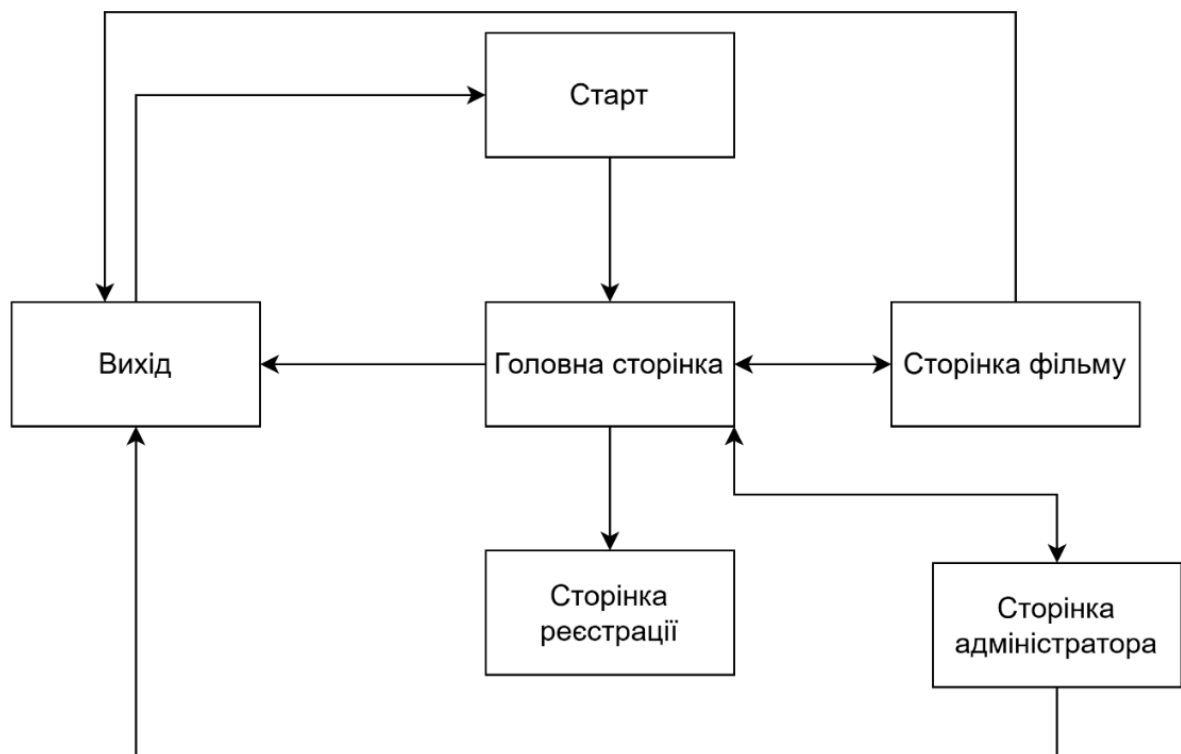


Рисунок 4.4– Схема функціонування програми

Головними складовими програми є:

- головна сторінка – містить список фільмів, панель навігації, а також доступ до фільтрації та пошуку;
- сторінка фільму – відображає повну інформацію про стрічку: опис, постер, трейлер, список акторів, можливість залишити коментар або оцінку;

- реєстрація та авторизація – сторінки для створення акаунту та входу користувача в систему;
- особистий профіль – містить інформацію про користувача;
- панель адміністратора – окремий інтерфейс для керування фільмами, жанрами та контентом.

4.5 Експлуатація програми

Взаємодія користувача з вебзастосунком є простою та зрозумілою. Інтерфейс однаковий для всіх, окрім адміністратора, який має додаткові інструменти для керування контентом. Щоб розпочати роботу із системою, новому користувачу потрібно спочатку зареєструватися, після чого авторизуватися у власному акаунті.

Після входу користувач отримує повідомлення про успішну авторизацію та можливість оцінювання та коментування фільмів. Доступ до основного функціоналу порталу мають як зареєстровані, так і не зареєстровані користувачі: перегляд каталогу фільмів, пошук за назвою, фільтрація за жанрами та сортування. Всі елементи інтерфейсу клікабельні — користувач може вільно переглядати інформацію про кожен фільм, включаючи постер, опис і трейлер.

Центральним елементом інтерфейсу є головна сторінка з переліком фільмів та зручною панеллю навігації, яка забезпечує швидкий доступ до ключових розділів сайту, таких як реєстрація, вхід, профіль, фільми. (рис. 4.5).

Головна сторінка також надає коротку інформацію про кожен фільм, включаючи жанр, рік випуску, рейтинг та тривалість. Це дозволяє користувачам швидко ознайомитися з доступним контентом і обрати те, що їх зацікавило. Такий підхід допомагає покращити зручність пошуку та підвищує інтерес до перегляду фільмів.

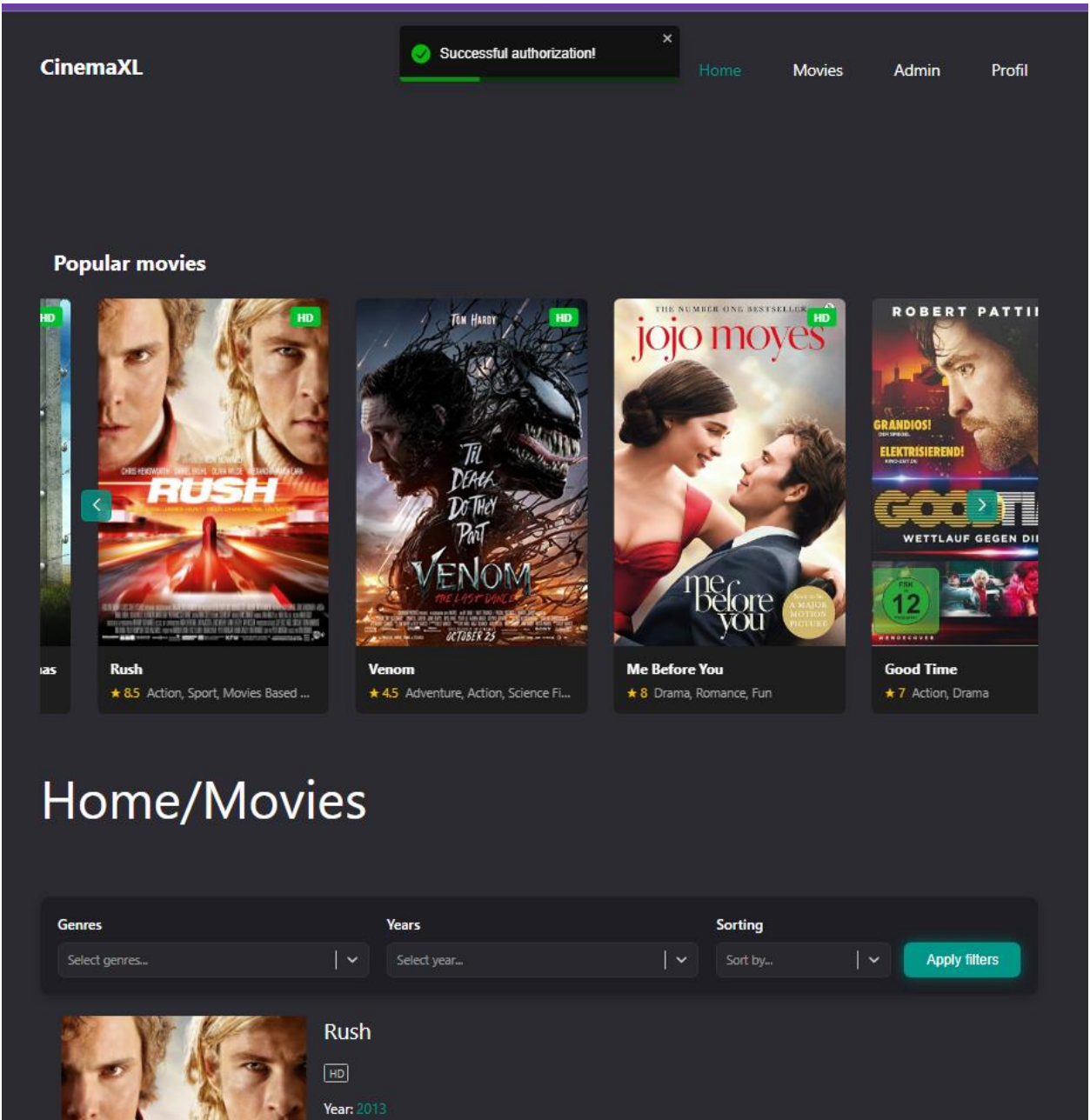


Рисунок 4.5 – Вікно головної сторінки

При натисканні на кнопку Movies можна побачити весь список фільмів (рис. 4.6), а також коротку інформацію про кожен фільм — жанр, рік випуску, рейтинг та тривалість. Це дає змогу швидко зорієнтуватися у доступному контенті, обрати цікавий фільм для перегляду та ознайомитися з його основними характеристиками без необхідності переходу на сторінку детального опису.

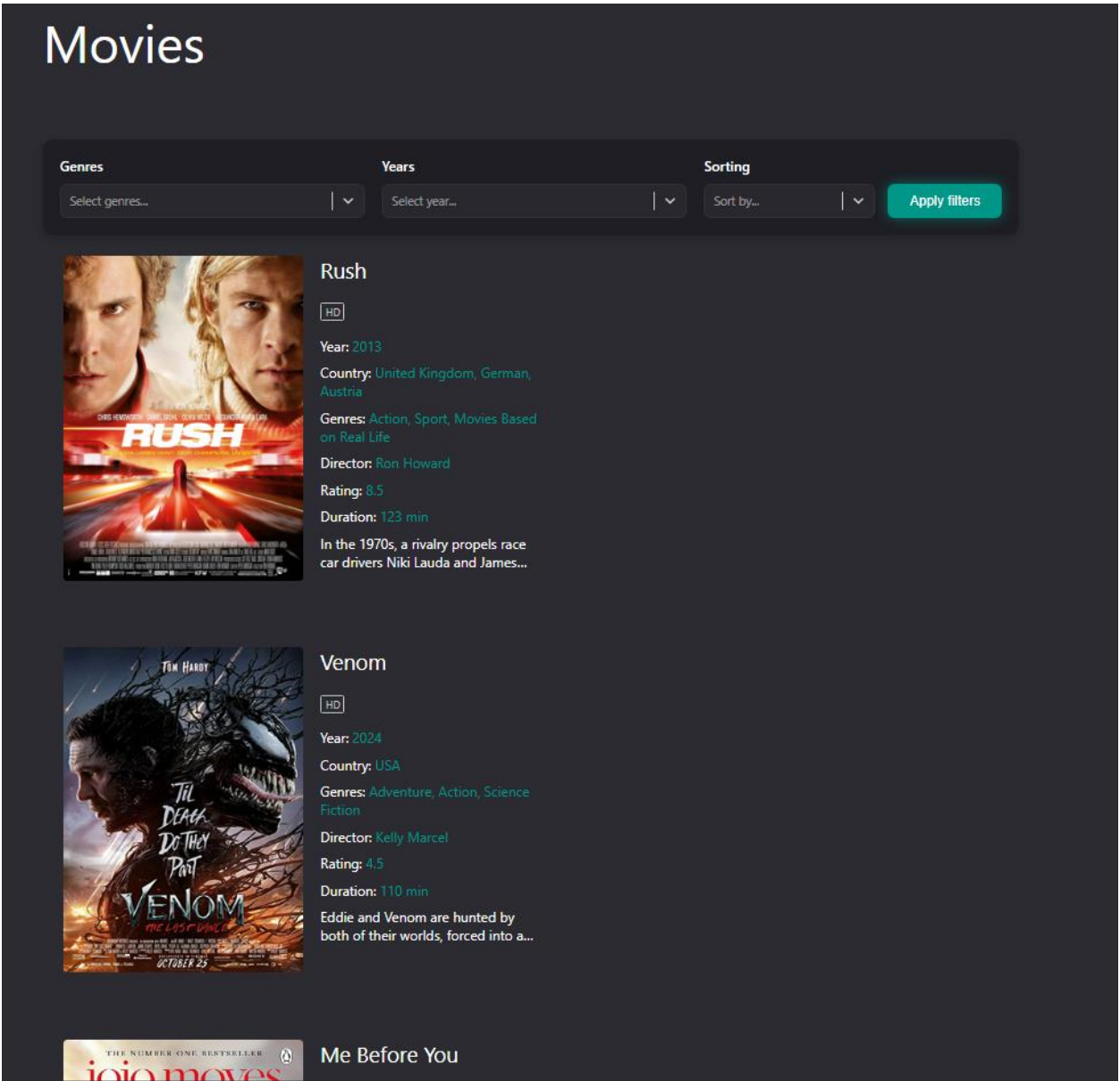


Рисунок 4.6 – Вікно сторінки Movies

Усі фільми є клікабельними та при натисканні на них можна побачити додаткову інформацію фільму з описом, трейлером, можливістю коментування та оцінювання. Також можна побачити кількість користувачів, які оцінили фільм, на основі чого автоматично розраховується його середня оцінка (рис. 4.7).

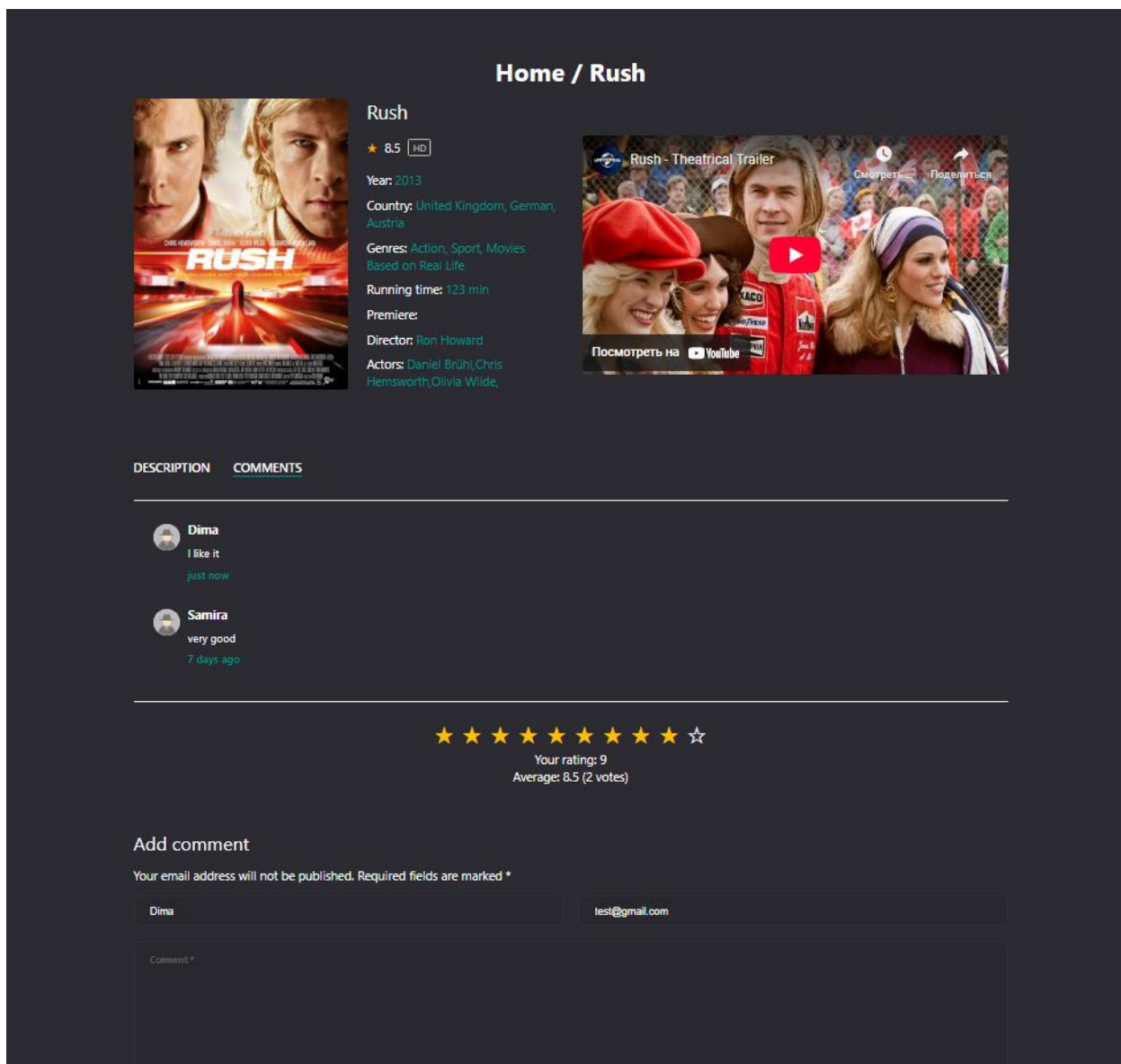
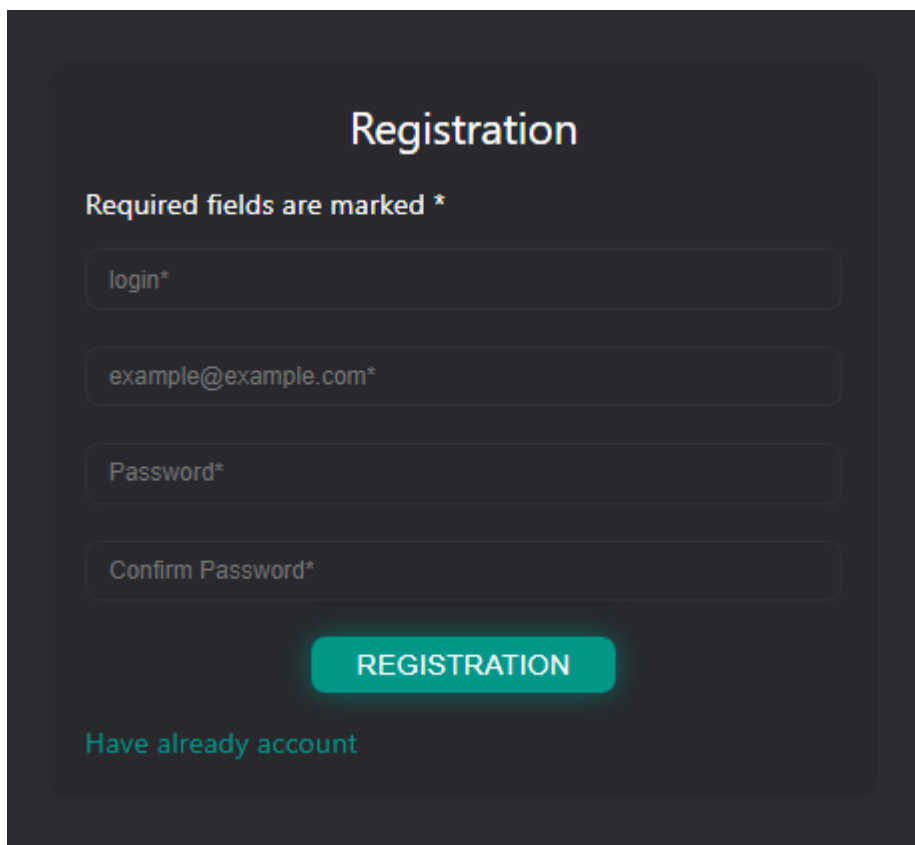


Рисунок 4.7 – Сторінка фільму

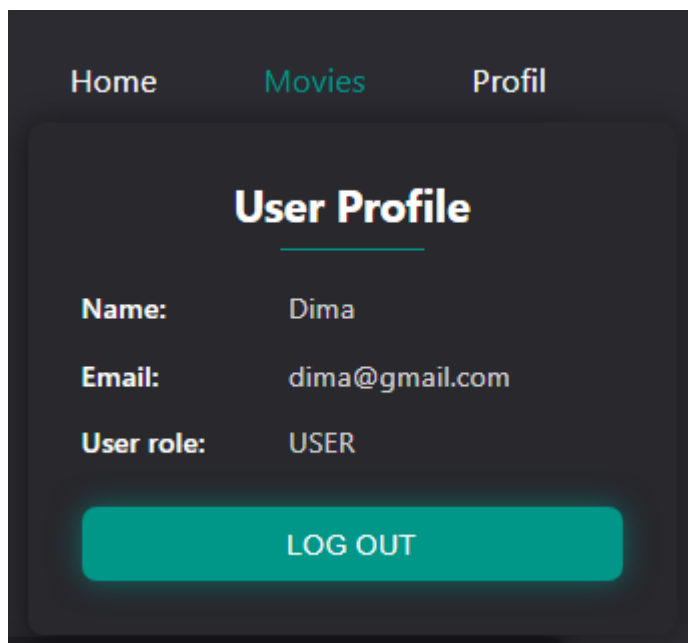
Для реєстрації потрібно ввести логін, пошту та пароль. Якщо користувач вже зареєстрований, він може натиснути на кнопку Have already account, що перенаправляє його на сторінку авторизації. Ім'я користувача може повторюватися, однак електронна пошта має бути унікальною – інакше система повідомить про помилку, що користувач з такою поштою вже існує (рис. 4.8).



The image shows a registration form on a dark background. At the top, the word "Registration" is centered in a large, white, sans-serif font. Below it, a smaller line of text reads "Required fields are marked *". There are four input fields stacked vertically, each with a light gray border and a small asterisk at the end of the placeholder text: "login*", "example@example.com*", "Password*", and "Confirm Password*". Below the input fields is a prominent red button with the word "REGISTRATION" in white, uppercase letters. At the bottom left of the form area, there is a link that says "Have already account" in a light blue color.

Рисунок 4.8 – Сторінка реєстрації

Натиснувши на кнопку **Profil**, користувач може побачити свої дані та кнопку **Log out**, що дає змогу вийти з акаунту (рис. 4.9).

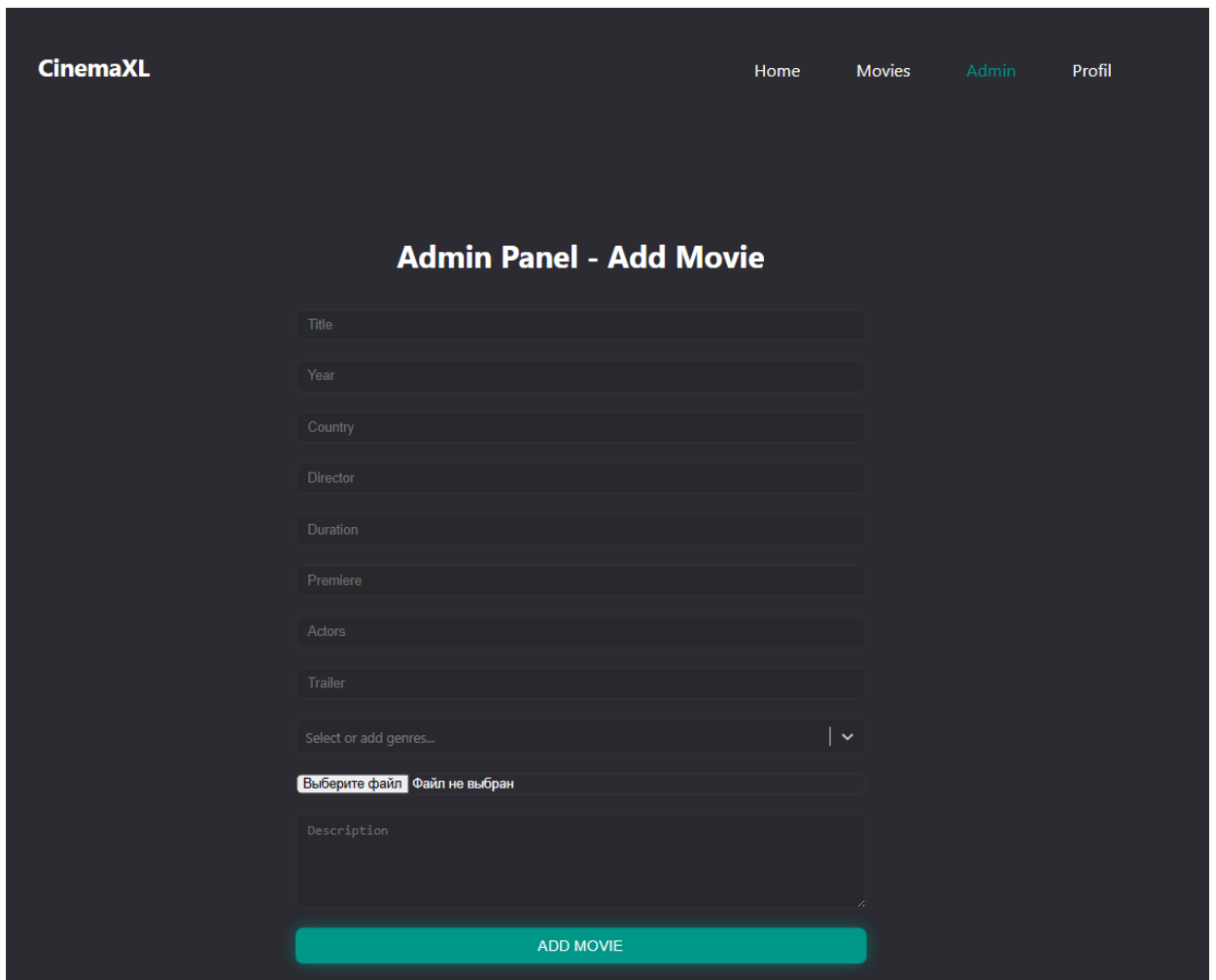


The image shows a user profile window. At the top, there is a navigation bar with three items: "Home", "Movies" (highlighted in red), and "Profil". Below the navigation bar, the title "User Profile" is centered in a large, white, sans-serif font. Underneath the title, there is a table-like structure with three rows of user information: "Name: Dima", "Email: dima@gmail.com", and "User role: USER". At the bottom of the profile section, there is a prominent red button with the text "LOG OUT" in white, uppercase letters.

Рисунок 4.9 – Вікно профілю з можливістю виходу

Адміністративна панель дозволяє додавати нові фільми до каталогу (рис. 4.10). Інтерфейс складається з форми, де адміністратор може ввести назву фільму, рік виходу, країну, режисера, тривалість, дату прем'єри, акторів, опис, посилання на трейлер, обрати жанри або створити нові, а також завантажити постер фільму. Після заповнення всіх полів потрібно натиснути кнопку Add Movie, щоб зберегти інформацію.

Після форми додавання розташований список уже доданих фільмів (рис. 4.11), кожен з яких має відображену основну інформацію: назву, рік, жанр, режисера, акторський склад та короткий опис. Також є кнопки Watch Trailer для перегляду трейлера та Delete для видалення відповідного фільму з бази. Ця функціональність дає змогу зручно редагувати та оновлювати каталог фільмів у системі.



The screenshot displays the 'Admin Panel - Add Movie' form within the CinemaXL application. The interface is dark-themed with teal accents. At the top, the CinemaXL logo is on the left, and navigation links for Home, Movies, Admin (highlighted in teal), and Profil are on the right. The form itself is titled 'Admin Panel - Add Movie' and contains several input fields: Title, Year, Country, Director, Duration, Premiere, Actors, Trailer, a dropdown menu for 'Select or add genres...', a file upload section with a button 'Выберите файл' and the text 'Файл не выбран', and a large text area for 'Description'. At the bottom of the form is a prominent teal button labeled 'ADD MOVIE'.

Рисунок 4.10 – Сторінка адміністративної панелі

Movies List

Rush (2013)

Genre: Action, Sport, Movies Based on Real Life

Director: Ron Howard

Actors: Daniel Brühl,Chris Hemsworth,Olivia Wilde,

Description: In the 1970s, a rivalry propels race car drivers Niki Lauda and James Hunt to fame and glory — until a horrible accident threatens to end it all.



Delete

Venom (2024)

Genre: Adventure, Action, Science Fiction

Director: Kelly Marcel

Actors: Tom Hardy, Chiwetel Ejiofor, Juno Temple, Stephen Graham, Rhys Ifans, Peggy Lu, Alanna Ubach

Description: Eddie and Venom are hunted by both of their worlds, forced into a decision ending their last dance.



Delete

Me Before You (2016)

Genre: Drama, Romance, Fun

Director: Thea Sharrock

Actors: Emilia Clarke, Sam Claflin, Charles Dance, Jenna Coleman, Matthew Lewis, Vanessa Kirby, Stephen Peacocke, Brendan Coyle, Janet McTeer

Description: A paralysed man develops a bond with his caregiver.

Рисунок 4.11 – Сторінка адміністративної панелі

ВИСНОВКИ

У процесі виконання дипломної роботи було досягнуто поставленої мети — розроблено сучасний, функціональний та інтерактивний інформаційний вебпортал, присвячений кіно, який об'єднує користувачів навколо теми кінематографу, забезпечує зручний доступ до фільмів, дозволяє ділитися враженнями, оцінювати контент та взаємодіяти із системою через зручний інтерфейс.

У межах проєкту було проведено аналіз сучасних інформаційних систем, призначених для поширення кіно- та мультимедійного контенту. Визначено ключові вимоги до подібних порталів, серед яких — інтуїтивність інтерфейсу, швидкодія, можливість персоналізації, простота адміністрування та безпека обробки даних користувачів. На основі проведеного аналізу було спроектовано архітектуру майбутнього застосунку, визначено функціональні модулі та інтерфейси взаємодії.

У якості технічної реалізації обрано стек сучасних технологій: клієнтська частина реалізована за допомогою бібліотеки React, що забезпечує гнучкий компонентний підхід; серверна частина створена з використанням Node.js та Express, що дозволяє реалізовувати REST-архітектуру та зручно обробляти запити; базу даних реалізовано на PostgreSQL, а для зручної взаємодії з нею застосовано Sequelize. Такий підхід дозволив створити масштабовану, модульну систему, яку в майбутньому легко оновлювати та доповнювати.

У результаті виконання проєкту було реалізовано реєстрацію та авторизацію користувачів, перегляд фільмів, їх оцінювання, коментування, групування за жанрами, а також створено адміністративну панель для керування вмістом. Інтерфейс розроблений у відповідності до сучасних принципів UX/UI, що забезпечує комфортне користування на різних пристроях. Особливу увагу приділено валідації даних, повідомленням про помилки та забезпеченню стабільної роботи сервісу.

Отриманий застосунок відповідає сучасним вимогам до вебплатформ, орієнтованих на мультимедійний контент. Особлива увага приділялася зручності користувача, логічній структурі інтерфейсу та простоті навігації. Система дозволяє швидко знаходити потрібні фільми, оцінювати їх, обговорювати в коментарях та ділитися власною думкою з іншими користувачами, що формує активну кіноспільноту.

У подальшому проєкті можна покращити функціонал рекомендаційної системи, додати: редагування облікового запису користувача, відповіді на коментарі та інтегрувати платіжні модулі для реалізації преміум-контенту. Також можливо розглянути інтеграцію мобільних додатків для Android та iOS та удосконалення системи безпеки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Про sweet.tv [Електрон. ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Sweet.tv>.
2. sweet.tv [Електрон. ресурс]. – Режим доступу: <https://sweet.tv/uk>.
3. Про Netflix [Electronic resource]. – Access mode: <https://marketing4ecommerce.net/en/history-of-netflix/>.
4. Netflix [Electronic resource]. – Access mode: <https://www.netflix.com/de-en/>.
5. Про IMDb [Electronic resource]. – Access mode: <https://en.wikipedia.org/wiki/IMDb>.
6. IMDb [Electronic resource]. – Access mode: <https://www.imdb.com/>.
7. Про Amazon Prime Video [Electronic resource]. – Access mode: <https://www.businessinsider.com/guides/streaming/what-is-prime-video>.
8. Amazon Prime Video [Electronic resource]. – Access mode: <https://www.amazon.com/gp/video/storefront>.
9. Django or Flask? Which Python Web Framework Is Right for Me in 2024? [Electronic resource]. – Access mode: <https://www.cloudways.com/blog/django-or-flask/>.
10. Advantages And Disadvantages Of Laravel Development [Electronic resource]. – Access mode: <https://dolphinwebsolution.com/blog/advantages-and-disadvantages-of-laravel-development/>.
11. Pros & Cons of Ruby on Rails You Should Know Before Choosing the Technology for Your Startup [Electronic resource]. – Access mode: <https://sloboda-studio.com/blog/pros-and-cons-of-ruby-on-rails/>.
12. Spring Boot: Advantages, Disadvantages [Electronic resource]. – Access mode: <https://medium.com/@jayeshwarke011/getting-started-with-spring-boot-advantages-disadvantages-and-use-cases-497b0f04fb86>.
13. C#: Advantages, Disadvantages [Electronic resource]. – Access mode: <https://www.coolcodecompany.co.uk/technologies/microsoft-technologies/c->

development/discover-the-pros-and-cons-of-c-expert-insights.

14. Pros and Cons of JavaScript Development: Advantages, Disadvantages [Electronic resource]. – Access mode: <https://pangea.ai/resources/javascript-best-practices>.

15. Why Use React? Top Benefits for Web Development [Electronic resource]. – Access mode: <https://www.netguru.com/blog/why-use-react>.

16. History of ReactJS [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/history-of-javascript/>.

17. History of ReactJS [Электрон. ресурс]. – <https://www.mygreatlearning.com/react-js/tutorials/history-of-reactjs>.

18. PostgreSQL [Electronic resource]. – Access mode: <https://www.postgresql.org/about/>.

19. PostgreSQL [Electronic resource]. – Access mode: <https://www.ibm.com/think/topics/postgresql>.

20. pgAdmin 4 [Electronic resource]. – Access mode: <https://www.adservio.fr/post/what-is-pgadmin>.

21. Postman [Electronic resource]. – Access mode: <https://apidog.com/blog/what-is-postman/>.

22. Draw.io [Electronic resource]. – Access mode: <https://fotc.com/blog/draw-io-online-guide/#:~:text=Draw.io%20is%20a%20free,Google%20Workspace%20or%20Gmail%20account>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для створення інформаційного порталу, що надає користувачам доступ до фільмів та рецензій.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмна реалізація інформаційного порталу, присвяченого кіно», затверджене наказом Національного університету «Запорізька політехніка» №209 від 28 квітня 2025 р.

A.2 Основні вимоги до програми, що розробляється

A.2.1 Вимоги до функціональних характеристик

Вимоги до функціональних характеристик:

- система повинна надавати можливість реєстрації та авторизації користувачів з різними ролями (адміністратор, звичайний користувач);
- користувач повинен мати можливість переглядати інформацію про фільми: назва, опис, жанр, рік виходу, країна, актори, трейлер, постер;
- реалізувати можливість оцінювання фільмів зареєстрованими користувачами;
- реалізувати функціонал коментування фільмів із подальшою модерацією коментарів;
- система повинна надавати можливість додавання, редагування та видалення фільмів через адміністративну панель;
- забезпечити сортування фільмів за жанрами, роками та іншими критеріями;
- реалізувати систему рекомендацій фільмів на основі історії переглядів користувача.

A.2.2 Вимоги до інтерфейсу програми

Вимоги до інтерфейсу програми:

- інтерфейс користувача повинен бути реалізований у веб-форматі з використанням сучасних фреймворків (React);
- інтерфейс повинен бути зрозумілим, інтуїтивно простим для користувача, з навігацією за категоріями;
- реалізувати розмежування доступу до функціоналу залежно від ролі користувача.

A.2.3 Вимоги до надійності

Вимоги до надійності:

- система повинна бути стійкою до помилкових дій користувача (введення некоректних даних, порожніх полів тощо);
- реалізувати обробку помилок на стороні сервера та клієнта;
- забезпечити захист від несанкціонованого доступу до адміністративного функціоналу.

A.2.4 Вимоги до складу та параметрів технічних засобів

Вимоги до складу та параметрів технічних засобів:

- клієнтська частина повинна працювати в сучасних браузерях (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari);
- середовище виконання клієнтської частини браузер користувача;
- серверна частина програми має бути сумісна з ОС Windows/Linux;
- середовище виконання серверної частини Node.js;
- СУБД: PostgreSQL.

ДОДАТОК Б
Текст програми

AdminPanelPage

```
import "./styles.scss";
import routeMain from "./routes";
import React, { useState, useEffect } from "react";
import Select from "react-select";
import CreatableSelect from "react-select/creatable";
import { createGenre, getGenres } from "../http/genreApi";

import { fetchGenres } from "../http/genreApi";
import { createMovie } from "../http/movieApi";
import { fetchMovies, getMovies, deleteMovie } from "../http/movieApi";
```

```
const AdminPanelPage = () => {
```

```
  const [movie, setMovie] = useState({
    title: "",
    year: "",
    country: "",
    genreIds: [],
    director: "",
    rating: "",
    duration: "",
    premiere: "",
    actors: "",
    poster: null, // Це файл!!!
    trailer: "",
    description: "",
  });
```

```
const handleGenreChange = (selectedOptions) => {
```

```
  const selectedGenreIds = selectedOptions.map((option) => option.value);
```

```
    setMovie((prev) => ({ ...prev, genreIds: selectedGenreIds }));
  };
```

```
const handleCreateGenre = async (inputValue) => {
  try {
    const newGenre = await createGenre(inputValue);
    setGenres((prev) => [...prev, newGenre]);
    return { value: newGenre.id, label: newGenre.name };
  } catch (error) {
    console.error("Помилка при створенні жанру:", error);
    return null;
  }
};
```

```
const [previewPoster, setPreviewPoster] = useState(null);
const [movies, setMovies] = useState([]);
const [genres, setGenres] = useState([]);
const [loading, setLoading] = useState(false);
const [error, setError] = useState(null);
```

```
useEffect(() => {
  const loadData = async () => {
    setLoading(true);
    try {
      const [moviesData, genresData] = await Promise.all([
        fetchMovies(),
        fetchGenres(),
      ]);

      setMovies(
```

```

        Array.isArray(moviesData) ? moviesData : moviesData?.rows || []
    );
    setGenres(
        Array.isArray(genresData) ? genresData : genresData?.rows || []
    );
} catch (err) {
    setError(err.message);
    console.error("Error loading data:", err);
} finally {
    setLoading(false);
}
};

```

```

    loadData();
}, []);

```

```

const loadGenres = async () => {
    try {
        const data = await fetchGenres();
        setGenres(data);
    } catch (error) {
        console.error("✖ Error loading genres:", error);
    }
};

```

```

const loadMovies = async () => {
    try {
        const data = await fetchMovies();
        setMovies(data);
    } catch (error) {

```

```

        console.error("❌ Error loading movies:", error);
    }
};

const handleDeleteMovie = async (movieId) => {
    if (window.confirm("Are you sure you want to delete this movie?")) {
        try {
            await deleteMovie(movieId);
            setMovies((prevMovies) =>
                prevMovies.filter((movie) => movie.id !== movieId)
            );
            alert("The film was successfully deleted");

            fetchMovies();
        } catch (error) {
            console.error("Error deleting movie:", error);
            alert("Failed to delete movie");
        }
    }
};

const handleChange = (e) => {
    const { name, value, files, options } = e.target;

    if (name === "poster" && files.length > 0) {
        setMovie((prev) => ({ ...prev, poster: files[0] }));
        setPreviewPoster(URL.createObjectURL(files[0]));
    } else if (name === "genre") {
        // multiple select
        const selectedGenres = Array.from(options)

```

```

        .filter((option) => option.selected)
        .map((option) => option.value);
    setMovie((prev) => ({ ...prev, genre: selectedGenres }));
  } else {
    setMovie((prev) => ({ ...prev, [name]: value }));
  }
};

const addMovie = async () => {
  try {
    const formMovie = {
      ...movie,
      year: parseInt(movie.year),
      rating: parseFloat(movie.rating),
      actors: movie.actors.split(",").map((a) => a.trim()),
      genreIds: movie.genreIds, // !!! важливо: genre -> genres
    };

    const newMovie = await createMovie(formMovie);
    setMovies((prevMovies) => [...prevMovies, newMovie]);
    // Очищення форми
    setMovie({
      title: "",
      year: "",
      country: "",
      genreIds: [],
      director: "",
      rating: "",
      duration: "",
      premiere: "",
    });
  } catch (error) {
    console.error(error);
  }
};

```

```

        actors: "",
        poster: null,
        trailer: "",
        description: "",
    });
    setPreviewPoster(null);
  } catch (error) {
    console.error("❌ Помилка при створенні фільму:", error);
  }
};

```

```

useEffect(() => {
  loadGenres();
  loadMovies();
}, []);

```

```

return (
  <div style={{ padding: "20px" }} className="admin">
    <h1>Admin Panel - Add Movie</h1>
    <div className="adminPanel">
      {[
        "title",
        "year",
        "country",
        "director",
        "duration",
        "premiere",
        "actors",
        "trailer",
      ]}.map((field) => (

```

```

<input
  key={field}
  name={field}
  value={movie[field]}
  onChange={handleChange}
  placeholder={field.charAt(0).toUpperCase() + field.slice(1)}
/>
)))}

```

```

<CreatableSelect
  isMulti
  name="genreIds"
  options={genres.map((g) => ({ value: g.id, label: g.name })))}
  value={movie.genreIds
    .map((id) => {
      const genre = genres.find((g) => g.id === id);
      return genre ? { value: genre.id, label: genre.name } : null;
    })
    .filter(Boolean)}
  onChange={(selectedOptions) => {
    const selectedIds = selectedOptions.map((option) => option.value);
    setMovie((prev) => ({ ...prev, genreIds: selectedIds }));
  }}
  onCreateOption={async (inputValue) => {
    try {
      console.log("Creating genre:", inputValue);
      const { genre: newGenre } = await createGenre({
        name: inputValue,
      });
      console.log("Created genre:", newGenre);
    }
  }}

```

```

    setGenres((prev) => [...prev, newGenre]);
    setMovie((prev) => ({
      ...prev,
      genreIds: [...prev.genreIds, newGenre.id],
    }));
    return { value: newGenre.id, label: newGenre.name };
  } catch (error) {
    console.error("Full error:", error);
    console.error("Error response:", error.response);
    console.error("Error creating genre:", error);

    return null;
  }
}
}
className="basic-multi-select"
classNamePrefix="select"
placeholder="Select or add genres..."
noOptionsMessage={() => "Enter a name for the new genre"}
formatCreateLabel={(inputValue) => `Create "${inputValue}"`}
/>
<input
  type="file"
  name="poster"
  accept="image/*"
  onChange={handleChange}
/>
<textarea
  name="description"
  value={movie.description}
  onChange={handleChange}

```

```

        placeholder="Description"
    />
    <button onClick={addMovie}>Add Movie</button>
</div>

```

```

    {previewPoster && (
      <div style={{ marginTop: "10px" }}>
        <p>Preview poster:</p>
        <img
          src={previewPoster}
          alt="Preview"
          style={{ width: "150px", height: "auto" }}
        />
      </div>
    )}

```

```

    {loading && <p>Loading...</p>}
    {error && <p style={{ color: "red" }}>Error: {error}</p>}
    <div style={{ marginTop: "40px" }}>
      <h2>Movies List</h2>

```

```

    {!loading && Array.isArray(movies) && movies.length === 0 && (
      <p>No movies found</p>
    )}

```

```

    {Array.isArray(movies)
      ? movies.map((m) => {
          console.log("Movie data:", m);
          return (
            <div

```

```

key={m.id}
style={{
  border: "1px solid #ccc",
  padding: "10px",
  marginBottom: "10px",
}}
>
<h3>
  {m.title} ({m.year})
</h3>
<p>
  <strong>Genre:</strong>{" "}
  {m.genres?.length > 0
    ? m.genres.map((g) => g.name).join(", ")
    : "No genres specified"}
</p>
<p>
  <strong>Director:</strong> {m.director}
</p>
<p>
  <strong>Actors:</strong>{" "}
  {typeof m.actors === "string"
    ? m.actors
    : Array.isArray(m.actors)
      ? m.actors.join(", ")
      : "No actors"}
</p>
<p>
  <strong>Description:</strong> {m.description}
</p>

```

```

    {m.poster && (
      <img
        src={process.env.REACT_APP_API_URL + m.poster}
        alt={m.title}
        style={{ width: "150px", height: "auto" }}
      />
    )}
    <p>
      <a
        href={m.trailer}
        target="_blank"
        rel="noopener noreferrer"
      >
        Watch Trailer
      </a>
    </p>
    <button
      onClick={() => handleDeleteMovie(m.id)}
      className="delete-button"
    >
      Delete
    </button>
  </div>
);
})
: !loading && <p>Failed to load movies data</p>}
</div>
</div>
);
};

```

```
export { routeMain };
export default AdminPanelPage;
```

Authorization

```
import "./styles.scss";
import { registration, login } from "../../http/userApi";
import { useState } from "react";
import { useNavigate } from "react-router-dom";
import { routeMain as routeMainPage } from "pages/MainPage";
import { toast, Bounce } from "react-toastify";
import useCurrentUser from "../../hooks/useCurrentUser";
```

```
const Authorization = () => {
  const [activeTab, setActiveTab] = useState("Auth");
  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
```

```
  const { setCurrentUser } = useCurrentUser();
  const navigateToMain = useNavigate();
  const notify = (message) => toast(message);
```

```
  const click = async (e) => {
    console.log(e);
    e.preventDefault();
```

```
    if (activeTab === "Auth") {
      const response = await login(name, email, password);
      if (!response.success) return;
```

```

setCurrentUser(response.data);
toast.success("Successful authorization!", {
  position: "top-center",
  autoClose: 5000,
  hideProgressBar: false,
  closeOnClick: false,
  pauseOnHover: true,
  draggable: true,
  progress: undefined,
  theme: "dark",
  transition: Bounce,
});

//notify("Successful authorization!");
console.log(response);
} else {
  const response = await registration(name, email, password);
  if (!response.success) return;

  setCurrentUser(response.data);

  toast.success("Successful registration!");

  console.log(response);
}
navigateToMain(routeMainPage());
};
return (
  <div className="authentication">

```

```

<form className="authenticationForm">
  <div className="authenticationTitle">
    {activeTab === "Auth" ? "Authorization" : "Registration"}
  </div>
  <div>
    <div className="authenticationDescription">
      Required fields are marked *
    </div>
    <div className="usersInputs">
      <input
        type="text"
        id="username"
        placeholder="login*"
        value={name}
        onChange={(e) => setName(e.target.value)}
      />
      <input
        type="email"
        id="email"
        placeholder="example@example.com*"
        value={email}
        onChange={(e) => setEmail(e.target.value)}
      />
      <input
        type="password"
        id="password"
        placeholder="Password*"
        required
        value={password}
        onChange={(e) => setPassword(e.target.value)}
      />
    </div>
  </div>
</form>

```

```

/>
{activeTab === "Registr" ? (
  <input
    type="password"
    id="confirmPassword"
    placeholder="Confirm Password*"
    required
  />
) : null}
</div>
</div>
<div className="authenticationFormButton">
  <button onClick={click}>
    {activeTab === "Auth" ? "Authorization" : "Registration"}
  </button>
</div>
{activeTab === "Auth" ? (
  <div      onClick={()      =>      setActiveTab("Registr")}
className="activeTab">
    No account yet
  </div>
) : (
  <div      onClick={()      =>      setActiveTab("Auth")}
className="activeTab">
    Have already account
  </div>
)}}
</form>
</div>
);

```

```
};
```

```
export default Authorization;
```

ДОДАТОК В
Слайди презентації

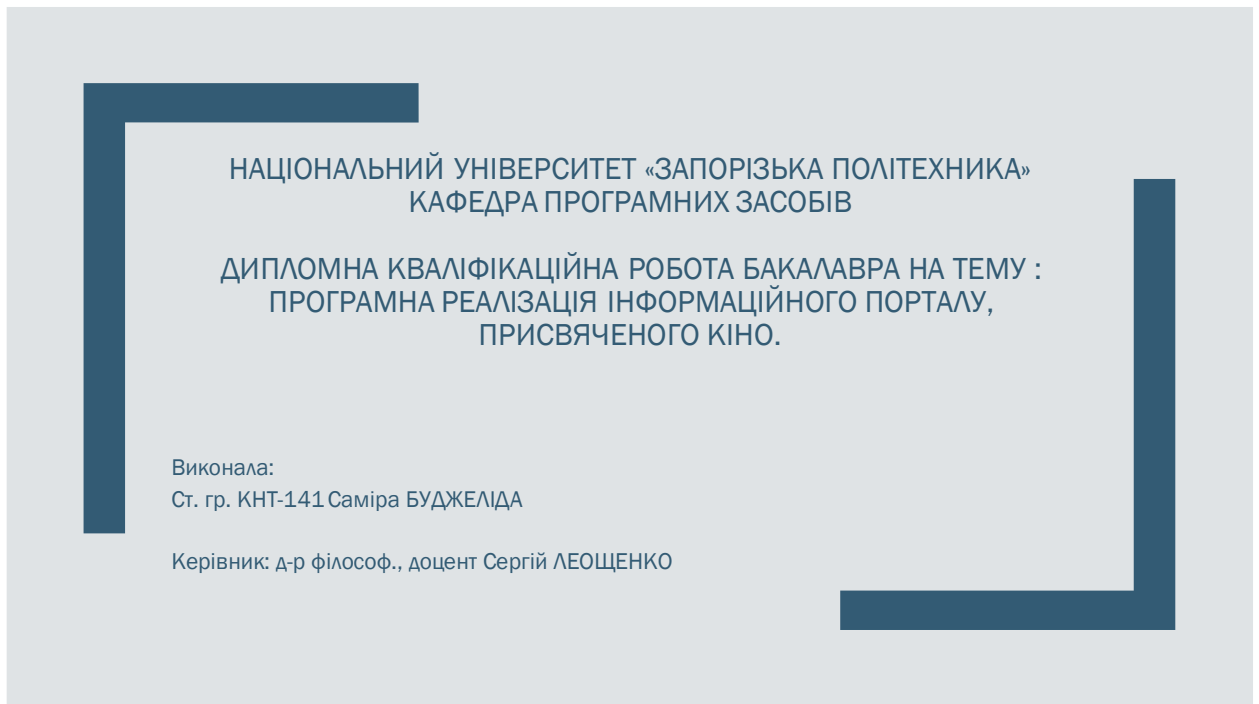


Рисунок Б.1 – Слайд № 1

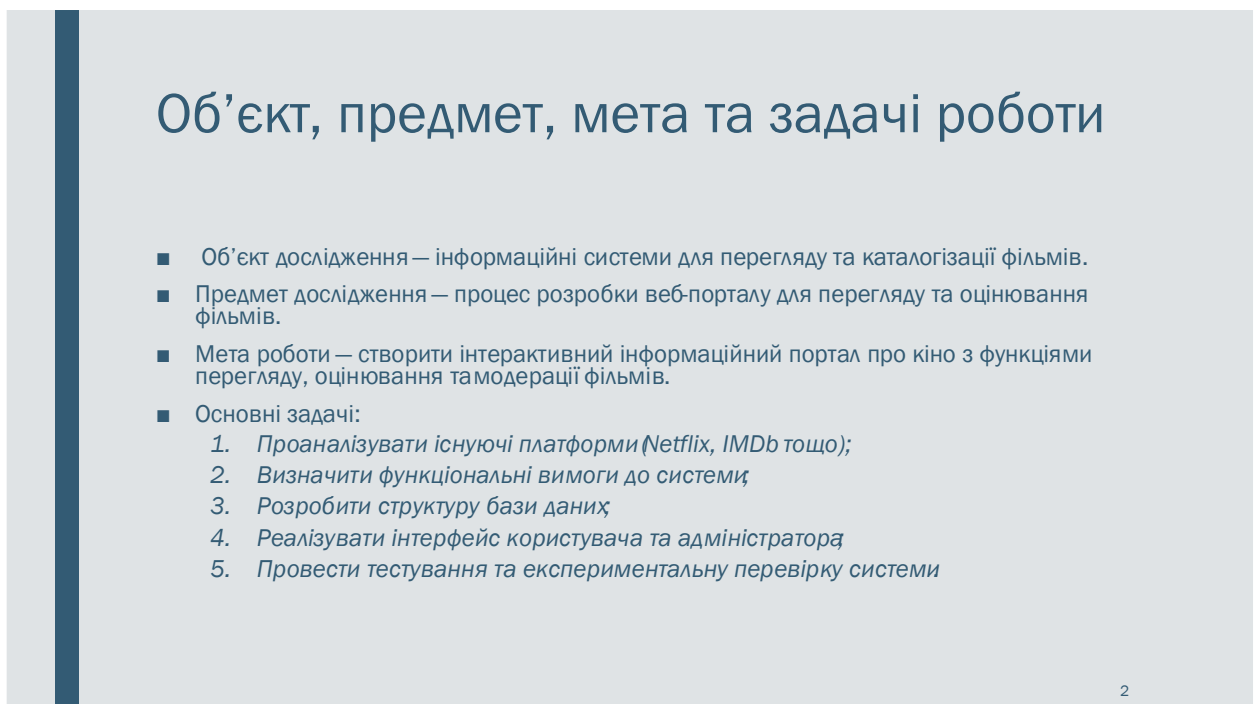


Рисунок Б.2 – Слайд № 2

Аналіз вимог до роботи

Функціональні вимоги:

- Реєстрація та авторизація користувачів
- Пошук і фільтрація фільмів за жанрами
- Перегляд описів, трейлерів, постерів фільмів
- Оцінювання та коментування контенту
- Адміністрування порталу (для адміністратора)

Нефункціональні вимоги:

- Інтуїтивний та адаптивний інтерфейс
- Висока швидкодія та продуктивність
- Захист даних користувачів та безпека
- Легке масштабування та оновлення системи

3

Рисунок Б.3 – Слайд № 3

Актуальність роботи

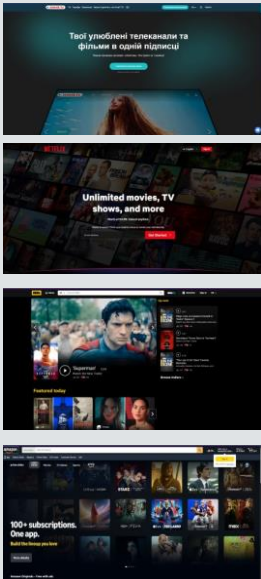
- Стрімке зростання популярності онлайн-ресурсів для перегляду кіно
- Потреба користувачів у швидкому доступі до фільмів, рейтингах та коментарях
- Важливість зручного інтерфейсу та залучення спільноти для обговорень
- Необхідність надійного адміністрування та стабільної роботи платформи

4

Рисунок Б.4 – Слайд № 4

Сервіси та сайти аналоги

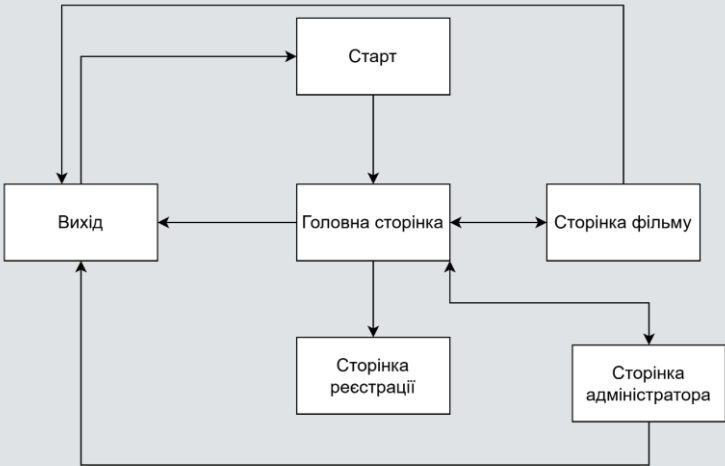
Платформа	Швидкодія	Безпека	Перегляд фільмів	Якість фільмів	Колекції фільмів
Sweet.tv	Висока	Висока	+	Висока	Обмежено
Netflix	Висока	Висока	+	Дуже висока	+
IMDb	Висока	Висока	-	-	+
Amazon Prime Video	Середня	Висока	+	Висока	Обмежено



5

Рисунок Б.5 – Слайд № 5

Структура програми



6

Рисунок Б.6 – Слайд № 6

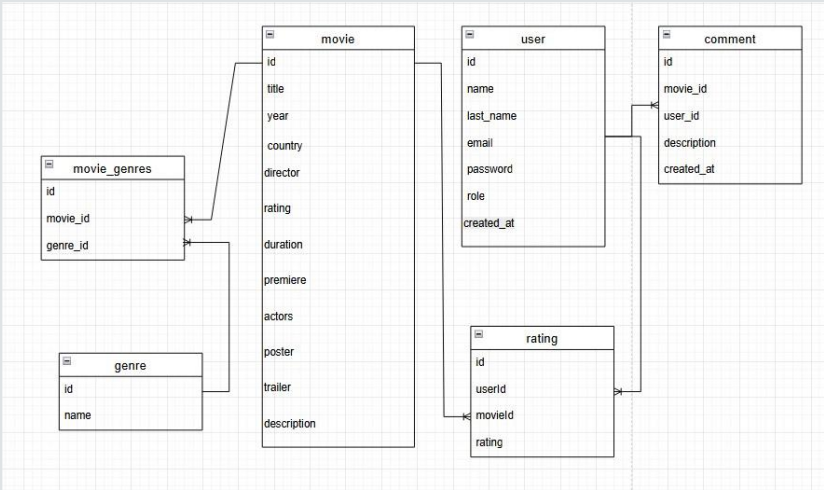
Порівняння мов програмування

Критерій	JavaScript + React	Python + Django	PHP + Laravel	Ruby on Rails	Java + Spring Boot	C# + ASP.NET Core
Сфера застосування	Frontend / Full-stack	Backend	Backend / Full-stack	Backend / Full-stack	Backend (Enterprise)	Backend (Enterprise)
Переваги	Єдина мова для клієнта й сервера (через Node.js), SPA, велика спільнота, адаптивність, компонентний підхід	Простота, вбудована адмін-панель, ORM, безпека, швидка розробка	Швидкий старт, Blade-шаблони, велика кількість ресурсів	Швидка розробка MVP, зрозумілий синтаксис	Потужність, надійність, масштабованість	Висока продуктивність, безпека, підтримка Microsoft
Недоліки	Потребує налаштувань для SSR, залежність від бібліотек	Слабкий frontend, потреба в окремому UI-фреймворку	Менша гнучкість інтерфейсу, застарілість підходів	Менша популярність, менша підтримка в Україні	Високий поріг входу, складність інтеграції з frontend	Складність хостингу на Linux, великий об'єм системи
Висновок	Оптимальний вибір для динамічного, інтерактивного та SPA-застосунку	Підходить для бекенду, але потребує окремого frontend-стеку	Можна використовувати, але менш гнучко для SPA	Не рекомендується через зменшення спільноти	Добрий вибір для великих систем, але надлишковий для порталу	Хороший варіант, але переважаний для SPA-порталу

7

Рисунок Б.7 – Слайд № 7

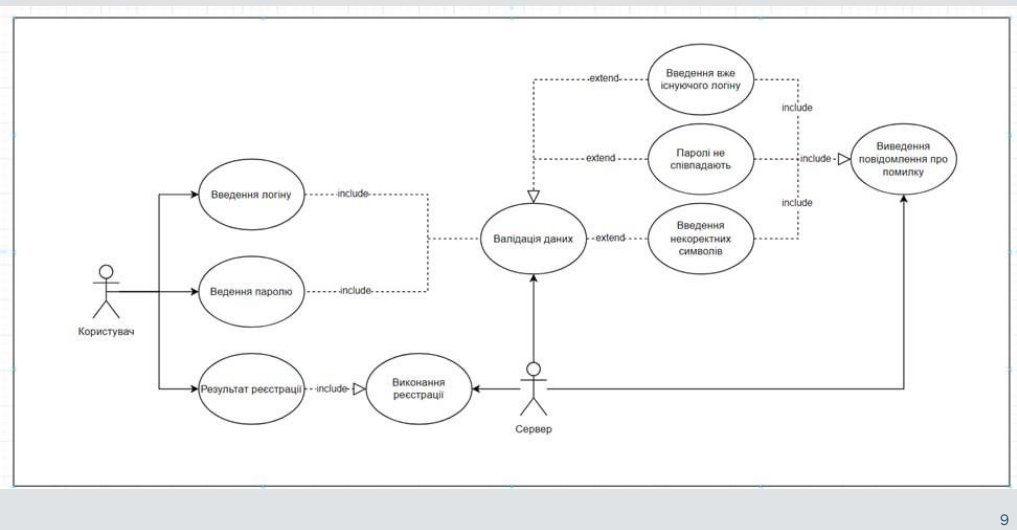
Схема бази даних



8

Рисунок Б.8 – Слайд № 8

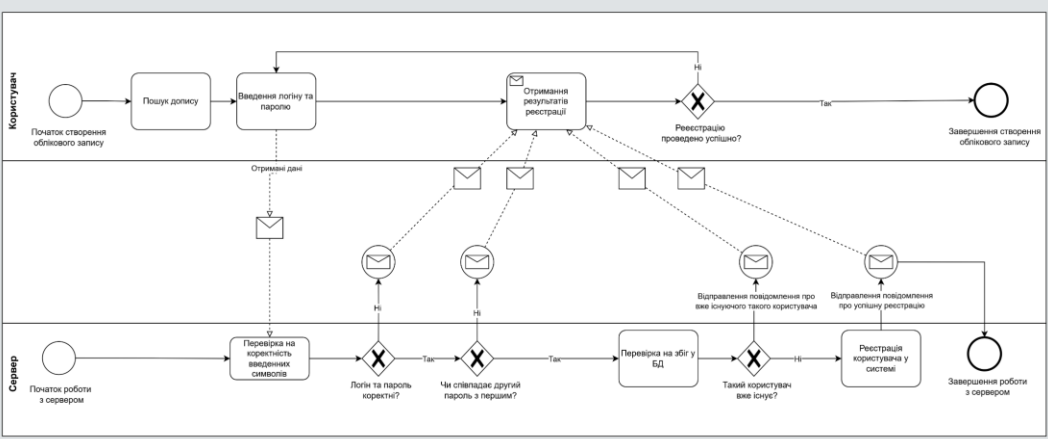
Діаграма прецедентів



9

Рисунок Б.9 – Слайд № 9

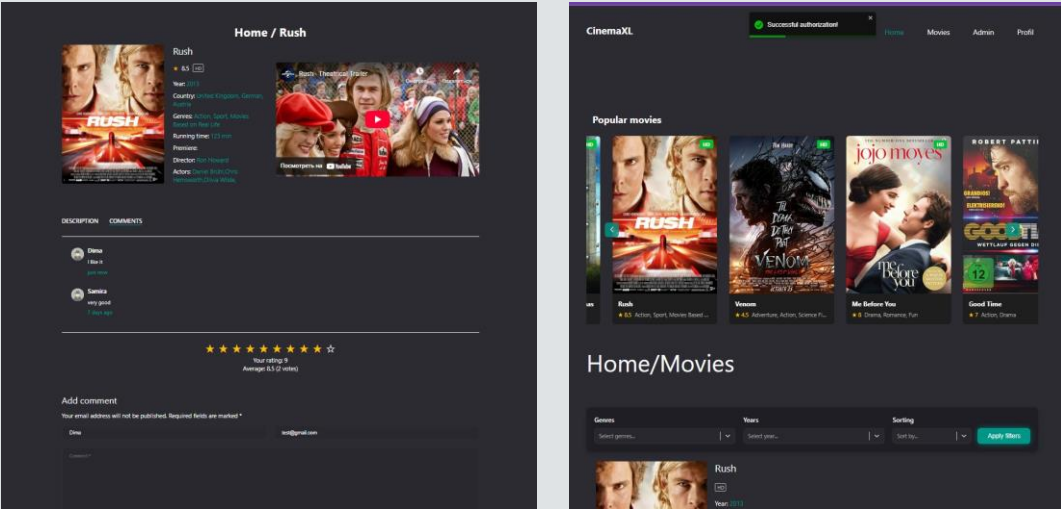
Діаграма бізнес-процесу створення облікового запису



10

Рисунок Б.10 – Слайд № 10

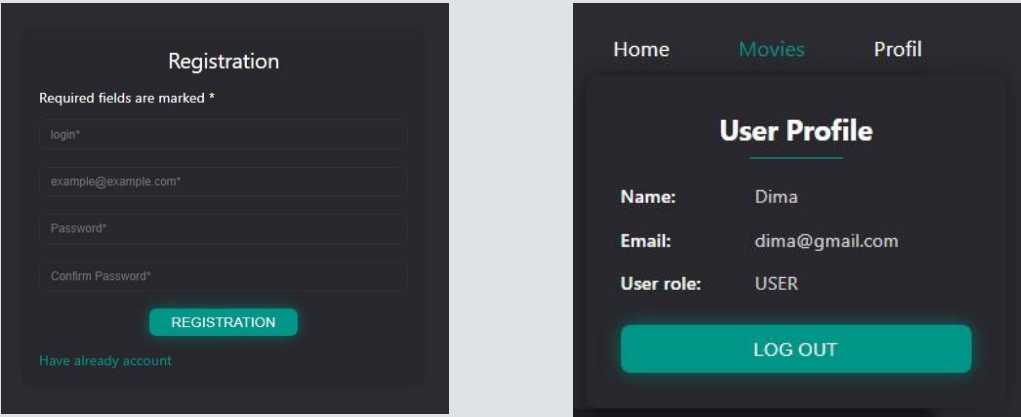
Интерфейс системы



11

Рисунок Б.11 – Слайд № 11

Интерфейс системы



12

Рисунок Б.12 – Слайд № 12

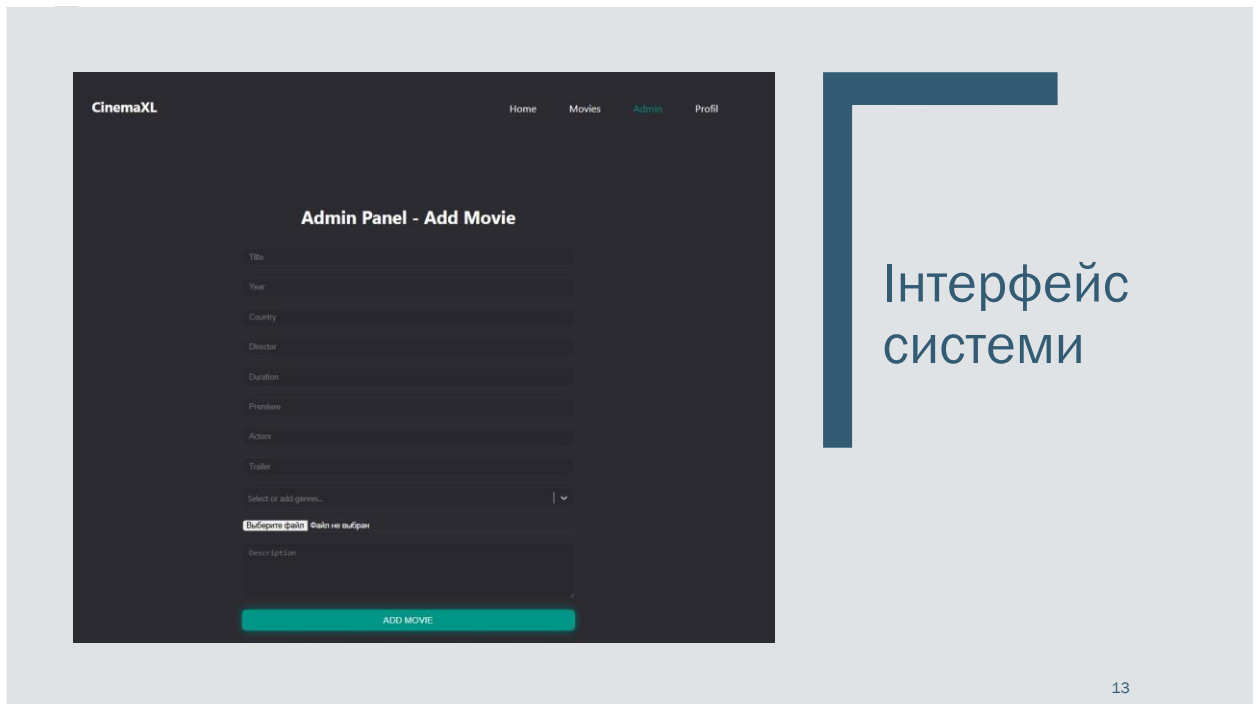


Рисунок Б.13 – Слайд № 13

Висновки

Розроблено повноцінний веб-портал про кіно з авторизацією, переглядом, оцінюванням та коментуванням фільмів.

Використано сучасні технології: React, Node.js, Express, PostgreSQL, Sequelize, REST-архітектура.

Забезпечено зручний інтерфейс, стабільну роботу та валідацію даних.

Платформа дозволяє швидко знаходити фільми, ділитися думками та оцінювати фільми.

Проект готовий до подальшого розвитку і може бути використаний як основа комерційного продукту.

Рисунок Б.14 – Слайд № 14

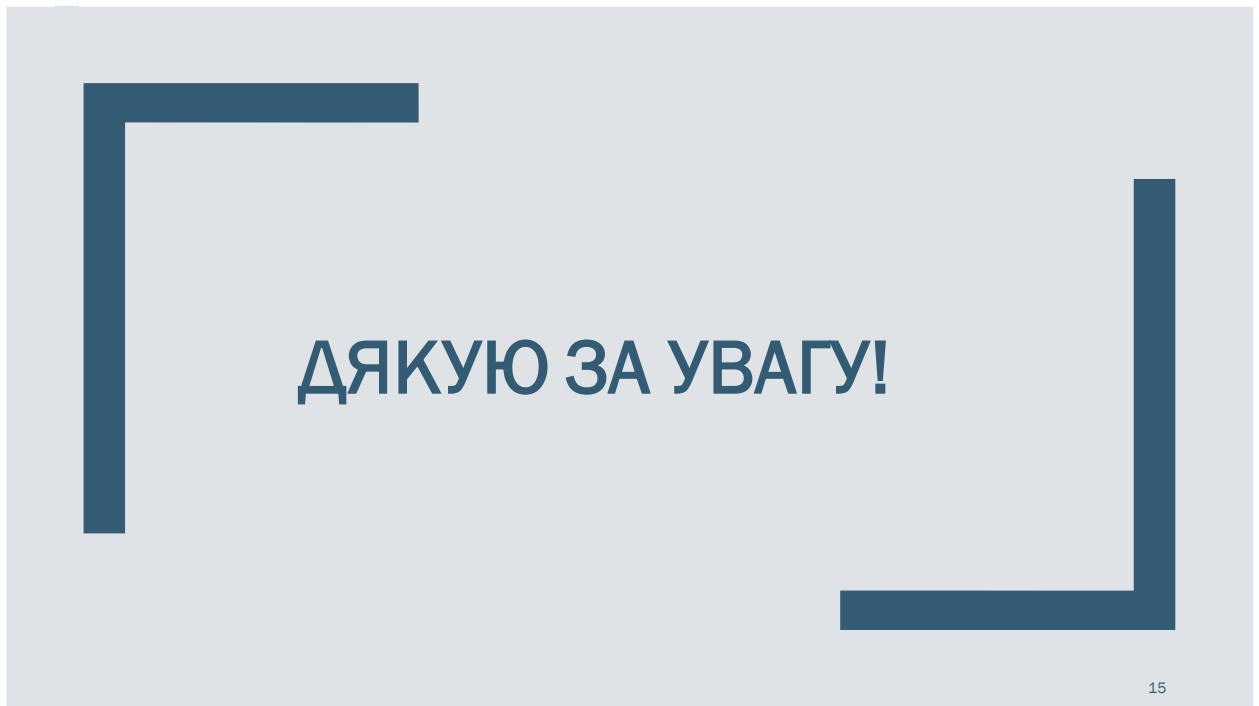


Рисунок Б.15 – Слайд № 15