МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«___» _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

Застосунок онлайн-кінотеатру

Спеціальність 121 Інженерія програмного забезпечення
Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ **Микола ХОМ'ЯНОВ**

«__» _____ 20__ р.

Керівник роботи
канд. техн. наук,
доцент _____ **Гліб ГОРБАНЬ**

«__» _____ 20__ р.

Миколаїв – 2026

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступінь	Бакалавр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«___» _____ 2026 р.

ЗАВДАННЯ
на кваліфікаційну бакалаврську роботу здобувача
Хом'янов Микола

1. Тема кваліфікаційної роботи «Застосунок онлайн-кінотеатру» затверджена наказом ректора ЧНУ ім. Петра Могили № _____ від «___» _____ 2025 р.

2. Строк представлення кваліфікаційної роботи «___» _____ 2026 р.

3. Очікуваним результатом є розробка вебзастосунку онлайн-кінотеатру, який дозволяє переглядати фільми, працювати з каталогом, обробку користувацьких запитів, а також збереження інформації про користувачів і їхні вподобання. Реалізація проєкту базується на аналізі аналогів, визначенні основних вимог та виборі відповідних технологій.

4. Перелік питань, що підлягають розробці:

– дослідження предметної області та визначення основних вимог до системи;

– розробка загальної структури та архітектури застосунку;

– проєктування та створення бази даних;

– реалізація серверної частини для обробки запитів користувачів;

– створення клієнтської частини інтерфейсу;

– проведення тестування та аналіз роботи застосунку.

5. Перелік графічних матеріалів: презентація.

6. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання «__24__» ____грудня____ 2025 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: **«Застосунок онлайн-кінотеатру»**

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КБР	24.12.2025	26.12.2025	*Виконано*
2.	Огляд літератури за темою роботи	10.02.2026	10.03.2026	*Виконано*
3.	Складання календарного плану КБР	11.03.2026	12.03.2026	*Виконано*
4.	Аналіз предметної області	14.03.2026	16.03.2026	*Виконано*
5.	Розробка проєктних рішень	10.04.2026	12.04.2026	*Виконано*
6.	Моделювання та конструювання ПЗ	20.04.2026	24.04.2026	*Виконано*
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	01.06.2026	11.06.2026	*Виконано*
8.	Відгук керівника КБР	18.05.2026	19.05.2026	*Виконано*
9.	Оформлення КБР та презентації	24.05.2026	26.05.2026	*Виконано*
10.	Попередній захист	27.05.2026	27.05.2026	*Виконано*
11.	Завершення оформлення КБР та презентації	27.05.2026	11.06.2026	*Виконано*
12.	Рецензування	12.06.2026	13.06.2026	*Виконано*
13.	Захист кваліфікаційної роботи			

Здобувач _____ **Микола ХОМ'ЯНОВ**

«__»_____20__р.

Керівник роботи

канд. техн. наук,

доцент _____ **Гліб ГОРБАНЬ**

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

«Застосунок онлайн-кінотеатру»

Здобувач 409 гр.: Хом'янов Микола

Керівник: канд. техн. наук, доцент Горбань Гліб

Актуальність теми полягає у стрімкому розвитку цифрових технологій та зростаючому попиті на онлайн-сервіси для перегляду відеоконтенту. Сучасні користувачі дедалі частіше обирають вебзастосунки, щоб отримувати доступ до фільмів у зручний для них час та з різних пристроїв.

Об'єкт роботи – процес розробки вебзастосунку для перегляду відеоконтенту.

Предмет роботи – методи та засоби створення онлайн-кінотеатру.

Мета роботи – розробка вебзастосунку онлайн-кінотеатру з базовим функціоналом для перегляду фільмів і управління контентом.

Кваліфікаційна робота складається із вступу, 4 розділів, висновків та переліку джерел посилання.

У вступі обґрунтовано актуальність теми, визначено мету, об'єкт та предмет дослідження.

У першому розділі проведено аналіз предметної області та існуючих аналогів.

У другому розділі присвячено аналізу сучасних технологій, інструментів та формуванню вимог до програмного забезпечення.

У третьому розділі виконано проєктування вебзастосунку онлайн-кінотеатру, розроблено архітектуру системи, структуру бази даних, UML-діаграми та макети інтерфейсу.

У четвертому розділі реалізовано вебзастосунок онлайн-кінотеатру, виконано програмну реалізацію основних функцій системи, проведено тестування та розроблено керівництво користувача.

У висновках підведено підсумки виконаної роботи та визначено результати дослідження.

Кваліфікаційна робота викладена на 79 сторінках машинописного тексту, складається із вступу, 4 розділів, загальних висновків, переліку джерел посилання з 16 найменувань та 3 додатків. Праця містить 10 таблиць та 23 рисунків.

Ключові слова: вебзастосунок, онлайн-кінотеатр, відеоконтент, база даних, бекенд, фронтенд, користувацький інтерфейс.

ABSTRACT

to the qualifying bachelor's thesis

"Online cinema application"

Student of 409 group: Khomianov Mykola

Supervisor: Candidate of Technical Sciences, Associate Professor Horban Hlib

Relevance of the topic lies in the rapid development of digital technologies and the growing demand for online services for viewing video content. Today's users are increasingly turning to web applications to access movies at their convenience and from various devices.

Object of the work – the process of developing a web application for viewing video content.

Subject of the work – methods and tools for creating an online movie theater.

Purpose of the work – to develop a web application for an online movie theater with basic functionality for watching movies and managing content.

The qualification work consists of an introduction, 4 sections, conclusions and a list of references.

In the introduction, the relevance of the topic is justified, and the purpose, object, and subject of the study are defined.

In the first section, an analysis of the subject area and existing counterparts is presented.

In the second section is devoted to an analysis of modern technologies and tools, as well as the development of software requirements.

In the third section, the design of a web application for an online movie theater was completed, and the system architecture, database structure, UML diagrams, and interface mockups were developed.

In the fourth section, the online movie theater web application is implemented, the main system functions are programmed, testing is conducted, and a user manual is developed.

In the conclusions, the work carried out is summarized and the results of the study are presented.

The qualification work is presented on 79 pages of typewritten text, consists of an introduction, 4 sections, general conclusions, a list of references with 16 titles and 3 appendices. The work contains 10 tables and 23 figures.

Keywords: web application, online movie theater, video content, database, backend, frontend, user interface.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API – інтерфейс програмування застосунків (Application Programming Interface)

Auth – автентифікація користувача (Authentication)

CRUD – базові операції з даними (Create, Read, Update, Delete)

DB – база даних (Database)

FTS – повнотекстовий пошук (Full Text Search)

HTTP – протокол передачі гіпертексту (HyperText Transfer Protocol)

HTTPS – захищений протокол передачі гіпертексту (HyperText Transfer Protocol Secure)

JSON – формат обміну даними (JavaScript Object Notation)

JWT – вебтокен JSON (JSON Web Token)

MVC – модель-представлення-контролер (Model-View-Controller)

ООР – об’єктно-орієнтоване програмування (Object-Oriented Programming)

ORM – об'єктно-реляційне відображення (Object-Relational Mapping)

REST – передача стану представлення (Representational State Transfer)

RegEx – регулярні вирази (Regular Expressions)

SQL – мова структурованих запитів (Structured Query Language)

UI – користувацький інтерфейс (User Interface)

UX – користувацький досвід (User Experience)

ВСТУП

Стрімкий розвиток інформаційних технологій та зростання швидкості доступу до Інтернету сприяли широкому поширенню цифрових сервісів для перегляду відеоконтенту. Сучасні користувачі дедалі частіше відмовляються від традиційного телебачення на користь онлайн-платформ, які надають доступ до фільмів та телесеріалів у зручний для них час і з різних пристроїв.

Навіть попри велику кількість онлайн-кінотеатрів, багато з них мають обмежену функціональність або незручний інтерфейс. Користувачі часто стикаються з труднощами у пошуку потрібного контенту, відсутністю ефективної навігації та недостатньою персоналізацією сервісу.

Зважаючи на це, виникає потреба у розробці вебзастосунку для онлайн-кінотеатру, який забезпечить:

– зручну реєстрацію та вхід користувачів;
– швидкий пошук та фільтрування відеоконтенту;
– ефективне управління контентом;
– інтуїтивно зрозумілий інтерфейс.

Підсумовуючи, розробка вебзастосунку для онлайн-кінотеатру є актуальним завданням у сфері веб-технологій та розробки програмного забезпечення.

Мета роботи – розробка вебзастосунку онлайн-кінотеатру з базовим функціоналом для перегляду та управління відеоконтентом.

Для реалізації поставленої мети в роботі необхідно виконати ряд завдань, зокрема: провести аналіз сфери діяльності онлайн-кінотеатрів, дослідити існуючі подібні платформи та визначити їхні сильні та слабкі сторони, проаналізувати сучасні технології та інструменти для розробки вебзастосунків, визначити вимоги до програмного забезпечення; а також розробити структуру та архітектуру застосунку.

Об’єкт роботи – процес розробки вебзастосунку для перегляду відеоконтенту.

Предмет роботи – методи та засоби створення онлайн-кінотеатру.

Обґрунтування доцільності розробки

Існуючі онлайн-кінотеатри не завжди забезпечують достатній рівень зручності та інтуїтивності. Під час їх використання користувачі часто стикаються з труднощами у пошуку потрібного контенту, навігації по сайту та непідходящими рекомендаціями.

У зв’язку з цим виникає потреба у створенні сучасного вебзастосунку, який забезпечить зручний доступ до відеоконтенту, простий інтерфейс, ефективну систему пошуку та базові функції управління контентом. Розробка такого рішення є своєчасною в контексті еволюції цифрових медіа-сервісів та веб-технологій.

Практичне значення одержаних результатів

Практичне значення одержаних результатів полягає в тому, що вони можуть бути використані:

– під час розробки вебзастосунків у сфері цифрових сервісів;

– як основа для навчальних проєктів у сфері веброзробки;

– для вдосконалення функціоналу онлайн-платформ для перегляду відеоконтенту;

– при проєктуванні користувацьких інтерфейсів вебзастосунків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області онлайн-кінотеатрів

Онлайн-кінотеатри – це один із найдинамічніших сегментів сучасної цифрової індустрії, який активно розвивається під впливом зростання швидкості інтернет-з'єднання, поширення мобільних пристроїв та змін у поведінці користувачів. Сьогодні відеоконтент дедалі частіше переглядають через вебзастосунки та стрімінгові платформи, які дають користувачам можливість отримувати доступ до фільмів та мультимедійного контенту в будь-який час і в будь-якому місці [6].

Разом із тим, дана предметна область має низку специфічних особливостей і викликів. До них належать необхідність ефективної організації великих обсягів відеоданих, забезпечення швидкого та зручного пошуку контенту, підтримка стабільної потокової передачі відео та впровадження інтуїтивно зрозумілого користувацького інтерфейсу. Крім того, ключовим аспектом є персоналізація надання користувачам рекомендацій на основі їхніх уподобань, історії переглядів та поведінки в системі.

Сучасні онлайн-кінотеатри приділяють особливу увагу персоналізації користувацького досвіду. На основі аналізу поведінки користувачів, історії переглядів та їхніх уподобань розробляються системи рекомендацій, покликані підвищити зацікавленість користувачів та покращити зручність користування сервісом.

Актуальність розробки спеціалізованого вебзастосунку онлайн-кінотеатру обумовлена тим, що існуючі рішення не завжди повною мірою задовольняють потреби користувачів. Вони часто мають перевантажений інтерфейс, неефективні алгоритми пошуку або обмежені можливості управління контентом. Крім того, деякі платформи не забезпечують достатньої адаптивності на різних пристроях або мають проблеми з продуктивністю під час значного навантаження. Тому існує потреба у створенні сучасного програмного продукту,

який поєднує в собі зручність використання, функціональність та високу продуктивність.

Об'єктом роботи є функціонування вебзастосунків для перегляду відеоконтенту, що включає взаємодію користувача з системою – від реєстрації та авторизації до вибору та перегляду фільмів, а також взаємодію з додатковими сервісами, такими як системи рекомендацій та обробки даних.

Предметом роботи є методи, моделі та технології, що застосовуються при розробці онлайн-кінотеатрів і забезпечують ефективну організацію контенту, обробку запитів користувачів, інтеграцію з базами даних та підтримку стабільної роботи вебзастосунку. Особлива увага приділяється питанням масштабованості, безпеки та оптимізації продуктивності системи, які мають вирішальне значення для сучасних веб-сервісів із великою кількістю користувачів.

Проєктування вебзастосунку онлайн-кінотеатру потребує комплексного підходу до створення програмної системи, яка має одночасно забезпечувати стабільну роботу сервісу, високу продуктивність та зручність у користуванні. Під час розробки необхідно враховувати як архітектурні особливості системи, так і вимоги до обробки та зберігання великих обсягів мультимедійних даних.

Одним із ключових аспектів є вибір архітектурної моделі системи. Найбільш доцільним у сучасних умовах є використання клієнт-серверного підходу з можливим поділом на окремі логічні модулі або мікросервіси. Така структура забезпечує масштабованість системи, спрощує її обслуговування та гарантує гнучкість при розробці нових функцій [3, 4].

Не менш важливим є проєктування структури бази даних, яка повинна забезпечувати зберігання інформації про користувачів, відеоконтент, жанри, рейтинги та історію переглядів [2, 5]. При цьому необхідно враховувати вимоги щодо швидкого доступу до даних та можливості їх подальшої аналітичної обробки. Реалізація серверної частини застосунку базується на сучасних технологіях, що підтримують розробку API та забезпечують безпечний обмін даними між компонентами системи [7, 8]. Клієнтська частина розроблена з використанням сучасних фронтенд-технологій, зокрема React та TypeScript, що

дозволяє створити динамічний та адаптивний користувацький інтерфейс [1, 9]. Важливим функціональним елементом системи є механізм персоналізації контенту. Він базується на аналізі дій користувача, його вподобань та історії переглядів, що дозволяє генерувати персоналізовані рекомендації та підвищує залученість користувачів до платформи.

Окрему роль відіграє підсистема роботи з відеоконтентом, яка відповідає за його ефективну передачу та відтворення в режимі потокового відео. Вона повинна забезпечувати адаптацію якості відео до швидкості інтернет-з'єднання та мінімізувати затримки під час відтворення. Таким чином, усі розглянуті технологічні та архітектурні рішення спрямовані на створення ефективного, масштабованого та зручного у користуванні вебзастосунку для онлайн-кінотеатру, який відповідає сучасним вимогам до цифрових медіасервісів.

1.2 Аналіз існуючих аналогів

У процесі розробки вебзастосунку для перегляду фільмів та серіалів важливим етапом є аналіз існуючих рішень у даній предметній області. Сучасний ринок онлайн-кінотеатрів характеризується високою конкуренцією, що вимагає детального вивчення найпопулярніших платформ. Такий аналіз дає змогу виявити ключові функції, які користуються попитом серед користувачів, а також визначити напрямки вдосконалення власного програмного продукту. У цьому розділі розглядаються такі платформи, як Netflix, Megogo та Sweet.tv. Одним із найбільш відомих представників вебзастосунків даного типу є платформа Netflix (рис. 1.1).

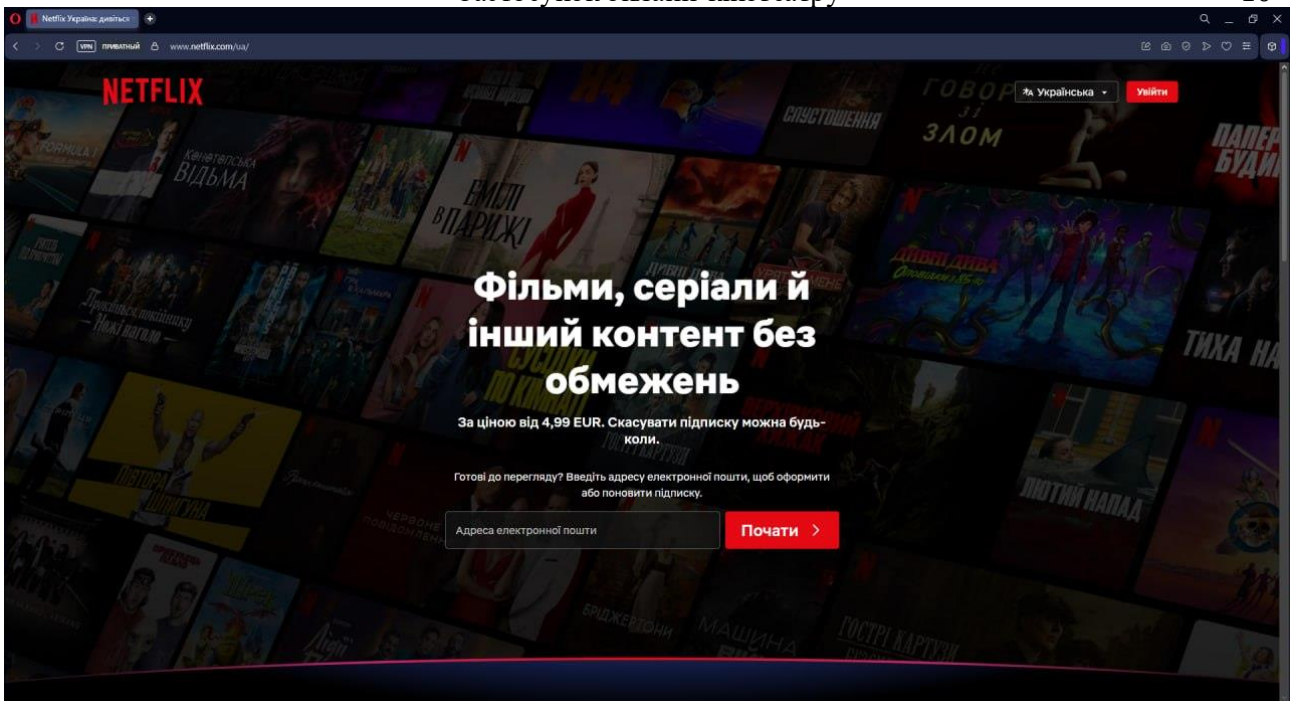

Рисунок 1.1 – Вебсайт Netflix

Netflix є одним з провідних платформ на світовому ринку стримінгу, пропонуючи користувачам широкий вибір фільмів, телесеріалів, документальних фільмів та ексклюзивного оригінального контенту. Платформа прагне надавати високоякісні послуги та відрізняється зручним та інтуїтивно зрозумілим інтерфейсом.

Основними перевагами є:

– широкий вибір контенту: платформа надає доступ до значного обсягу відеоматеріалів різних жанрів, зокрема фільмів, серіалів, документальних програм та дитячого контенту;

– наявність оригінального контенту: платформа створює власні фільми та серіали, що є його конкурентною перевагою;

– висока якість відео: підтримка форматів 4K, HDR та Dolby Vision забезпечує високу якість відтворення відео;

– персоналізація контенту: використання системи рекомендацій на основі історії переглядів дозволяє пропонувати користувачам релевантний контент;

– мультиплатформеність: доступ до платформи можливий з різних пристроїв, зокрема телевізорів, смартфонів та комп’ютерів.

Недоліки:

– вартість підписки: платні тарифні плани можуть бути недоступними для частини користувачів;

– регіональні обмеження контенту: через ліцензійні умови обсяг доступного контенту може відрізнятися залежно від країни;

– залежність від інтернет-з’єднання: для перегляду відео у високій якості необхідне стабільне та швидкісне підключення до мережі Інтернет;

Megogo (рис. 1.2) є популярною українською платформою для перегляду фільмів, серіалів і телевізійних каналів. Платформа надає доступ до широкого спектру вітчизняного контенту, зокрема українських фільмів та серіалів, а також пропонує значний каталог міжнародного відеоконтенту.

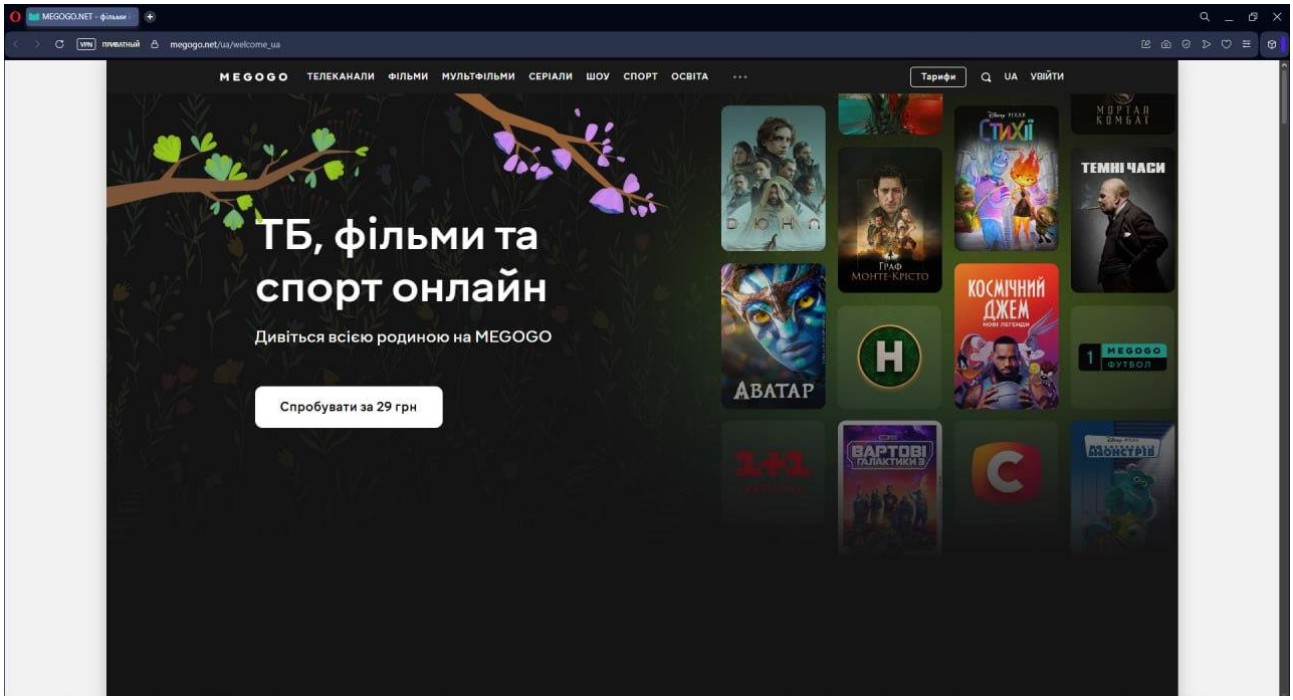

Рисунок 1.2 – Вебсайт Megogo

Переваги:

– широкий асортимент контенту: платформа надає доступ не лише до фільмів і серіалів, але й до телевізійних каналів, спортивних трансляцій та інших мультимедійних матеріалів;

– наявність українського контенту: платформа містить значну кількість українських фільмів та серіалів;

– адаптивність інтерфейсу: платформа забезпечує зручне використання на різних пристроях, зокрема смартфонах, планшетах, персональних комп'ютерах і телевізорах;

– частково безкоштовний доступ: окремий контент доступний безкоштовно за умови перегляду рекламних матеріалів.

Недоліки:

– якість окремого контенту: у порівнянні з міжнародними сервісами якість деяких відеоматеріалів може бути нижчою;

– наявність реклами: у безкоштовній версії присутні рекламні вставки, що може негативно впливати на користувацький досвід;

– обмежена доступність HD-контенту: не всі матеріали представлені у високій роздільній здатності.

Sweet.tv (рис. 1.3) є ще однією популярною українською платформою, що надає користувачам доступ до фільмів, серіалів і телевізійних каналів у режимі онлайн.

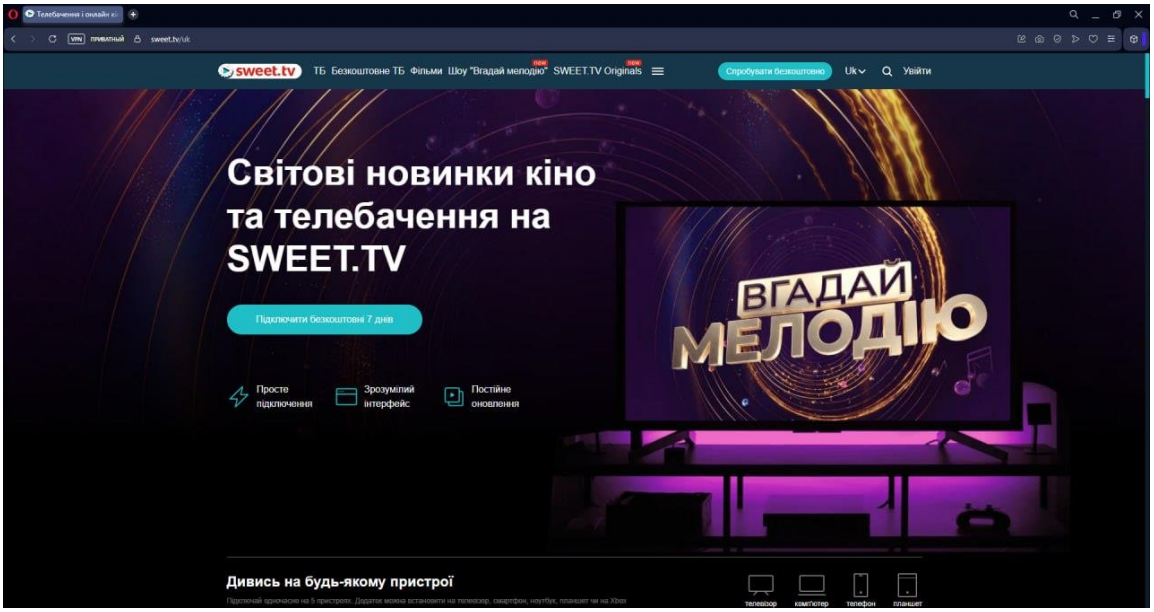

Рисунок 1.3 – Вебсайт Sweet.tv

Платформа є одним із провідних вітчизняних рішень у цій сфері та відрізняється активним розвитком і розширенням функціоналу.

Перевага:

– широкий вибір телеканалів: платформа надає доступ до значної кількості телеканалів різних жанрів, зокрема новинних, спортивних і розважальних, що підвищує її універсальність;

– наявність спортивних трансляцій: користувачі мають можливість переглядати спортивні події в режимі прямого ефіру;

– зручний інтерфейс: інтерфейс платформи є інтуїтивно зрозумілим, що спрощує навігацію та використання сервісу;

– мультиплатформеність: підтримується одночасний доступ із різних пристроїв, що забезпечує зручність для користувачів.

Недоліки:

– обмежений обсяг контенту: у порівнянні з міжнародними сервісами вибір фільмів і серіалів може бути меншим, особливо у високій якості;

– наявність реклами: у безкоштовних тарифах присутні рекламні матеріали, що може знижувати комфорт користування;

– технічні обмеження: у деяких випадках спостерігається нестабільна якість трансляцій, зокрема під час перегляду прямих ефірів.

Для більш наочного порівняння основних функціональних можливостей вищезгаданих платформ доцільно сформувати узагальнюючу таблицю. Вона дозволяє систематизувати ключові характеристики кожного сервісу та наочно відобразити їхні сильні й слабкі сторони. Такий підхід спрощує аналіз і дає змогу більш обґрунтовано визначити вимоги до розроблюваного вебзастосунку. Враховуючи наявні переваги та недоліки розглянутих платформ, сформовано порівняльну таблицю програмних аналогів, що дозволяє узагальнити отримані результати аналізу.

Таблиця 1.1 – Порівняльна характеристика платформ

Характеристики	Netflix	Megogo	Sweet.tv
Вибір контенту	Широкий	Широкий	Широкий (телеканали, спорт)
Якість контенту	Висока (4K, HDR)	Здебільшого стандартна	Стандартна
Безкоштовний перегляд	Ні	Так (з рекламою)	Так (з рекламою)
Персоналізація	Є (рекомендації)	Є (рекомендації)	Обмежена
Адаптивність	Так	Так	Так
Трансляції в реальному часі	Ні	Ні	Так (спорт, новини)
Реклама	Немає	Є (на безкоштовному пакеті)	Є (на безкоштовному пакеті)

Проведений аналіз дозволяє визначити ключові вимоги до розробки вебзастосунку. Враховуючи досвід існуючих конкурентів, важливо забезпечити високу якість відеоконтенту, зручну навігацію та можливість персоналізації. Також необхідна підтримка різних типів контенту, зокрема фільмів, серіалів, спортивних трансляцій та телеканалів. Крім того, платформа повинна мати інтуїтивно зрозумілий інтерфейс та адаптуватися до різних типів пристроїв, щоб забезпечити комфортну роботу користувачів.

1.3 Функціональні можливості вебзастосунку

Розроблюваний вебзастосунок онлайн-кінотеатру являє собою клієнт-серверну систему, яка забезпечує взаємодію користувача з мультимедійним контентом через вебінтерфейс. Архітектура застосунку базується на поділі на фронтенд і бекенд частини, що дозволяє розділити відповідальність за

відображення даних та їх обробку, підвищуючи масштабованість і підтримуваність системи. На стороні клієнта передбачається використання сучасного односторінкового застосунку (SPA), реалізованого за допомогою React та TypeScript [1, 9]. Такий підхід забезпечує швидку навігацію між сторінками без повного перезавантаження сторінки, а також дозволяє створювати компоненти інтерфейсу, що можуть використовуватися повторно. Для маршрутизації сторінок використовується React Router, що дає змогу організувати логічну структуру додатка (головна сторінка, каталог фільмів, сторінка перегляду, профіль користувача тощо) та забезпечує зручну навігацію.

Бекенд частина системи відповідає за обробку бізнес-логіки, авторизацію користувачів та взаємодію з базою даних. В її основі лежить REST API, через який клієнтська частина отримує доступ до даних [7, 8]. Для реалізації серверної логіки можна використовувати сучасні фреймворки на базі Node.js або подібних технологій, що підтримують асинхронні запити та аутентифікацію користувачів за допомогою JWT.

Система зберігає дані в реляційній базі даних, яка містить такі об’єкти, як користувачі, відеоконтент, жанри, рейтинги та історія переглядів [2, 5]. Для оптимізації доступу до інформації система використовує механізми індексації та швидкого пошуку, що дозволяє користувачам швидко знаходити потрібні фільми за назвою, жанром або іншими критеріями. З розвитком інтернет додатків на тлі загального підвищення кількості користувачів основна трирівнева клієнт-серверна модель була розширена шляхом введення додаткових рівнів [10].

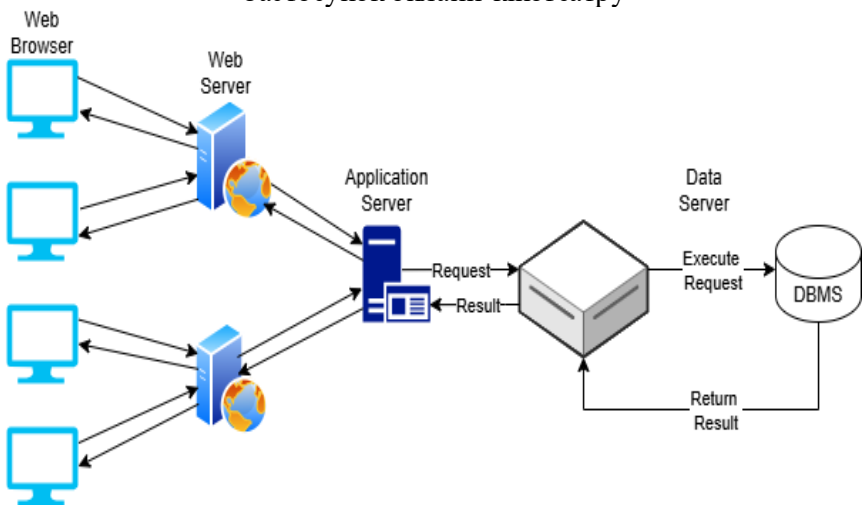

Рисунок 1.4 – Системна архітектура «клієнт-сервер»

Функціонально вебзастосунок онлайн-кінотеатру включає такі основні можливості:

– реєстрація та авторизація користувачів із захищеним доступом до персонального кабінету;

– перегляд каталогу фільмів із можливістю фільтрації за жанром, роком випуску та рейтингом;

– пошук відеоконтенту за назвою або ключовими словами;

– сторінка перегляду фільму з потоковим відтворенням відео;

– система оцінювання та коментування контенту;

– формування списку «обраного» для збереження контенту;

– персоналізовані рекомендації на основі історії переглядів користувача;

– адміністративна панель для додавання, редагування та видалення відеоматеріалів.

Варто також відзначити нефункціональні вимоги до системи, серед яких адаптивний дизайн для різних пристроїв (настільних комп'ютерів, планшетів та мобільних телефонів), висока продуктивність за умови одночасного підключення великої кількості користувачів та стабільна передача відео. Важливим аспектом є також безпека даних користувачів та захист від несанкціонованого доступу [3, 4].

Таким чином, функціональна структура вебзастосунку онлайн-кінотеатру охоплює весь цикл взаємодії з користувачем – від авторизації та пошуку контенту до перегляду та персоналізації досвіду забезпечуючи цілісність платформи та зручність її використання.

Висновки до розділу 1

У результаті проведеного системного аналізу було досліджено предметну область розробки вебзастосунку онлайн-кінотеатру та визначено ключові технологічні підходи до реалізації подібних систем. Встановлено, що сучасні платформи для перегляду відеоконтенту базуються на клієнт-серверній архітектурі, що забезпечує гнучкість, масштабованість і зручність подальшого розвитку. Дослідження існуючих рішень, зокрема Netflix, Megogo та Sweet.tv, дозволило виділити їхні основні особливості та функціональні акценти. Зокрема, Netflix орієнтується на якість контенту та персоналізацію, Megogo – на місцевий та спортивний контент, а Sweet.tv надає доступ до телеканалів та прямих трансляцій. Проведений аналіз результатів показує, що вебзастосунок, який розробляється, має підтримувати різні типи контенту, бути адаптованим для різних пристроїв та містити механізм надання персоналізованих рекомендацій користувачам. Окрім того, було визначено сучасні підходи до впровадження таких систем, зокрема багатоплатформна підтримка, стабільна передача відео та орієнтація на зручність користування. Врахування цих факторів дозволить підвищити якість розроблюваного рішення та забезпечити його відповідність сучасним вимогам ринку. Таким чином, результати проведеного аналізу формують основу для подальшого проєктування архітектури системи та деталізації функціональних вимог до вебзастосунку онлайн-кінотеатру.

2 АНАЛІЗ СУЧАСНОГО СТАНУ ІНСТРУМЕНТАРІЮ, МОДЕЛЕЙ ТА МЕТОДІВ

2.1 Аналіз сучасного стану інструментарію розробки вебзастосунків

Сучасна розробка вебзастосунків характеризується стрімким розвитком технологій, орієнтованих на забезпечення високої продуктивності, масштабованості, безпеки та зручності використання програмного забезпечення. Зокрема, це стосується систем потокового передавання мультимедійного контенту, до яких належать онлайн-кінотеатри. Такі системи повинні забезпечувати швидке завантаження сторінок, стабільне відтворення відео, ефективну обробку великої кількості одночасних запитів, а також підтримку різних типів пристроїв.

У сучасних умовах більшість вебзастосунків створюються на основі архітектури «клієнт-сервер» та підходу SPA (Single Page Application) [1]. Даний підхід передбачає динамічне оновлення вмісту сторінки без її повного перезавантаження, що значно прискорює взаємодію користувача з системою та забезпечує більш зручний інтерфейс. Для онлайн-кінотеатрів архітектура SPA є особливо ефективною, оскільки дозволяє швидко оновлювати розклад сеансів, результати пошуку, рекомендації та мультимедійний контент.

Також одним із найбільш поширених інструментів для розробки клієнтської частини сучасних вебзастосунків є React [1]. Дана бібліотека була розроблена компанією Meta та активно використовується для створення інтерактивних інтерфейсів. Основною перевагою React є компонентний підхід, який дозволяє повторно використовувати елементи інтерфейсу, спрощує обслуговування коду та покращує масштабованість проєкту. Окрім того, React використовує механізм Virtual DOM, який забезпечує ефективне оновлення сторінок і позитивно впливає на продуктивність вебзастосунку. Компонентний підхід React дозволяє розділити інтерфейс вебзастосунку на незалежні функціональні елементи, що забезпечує повторне використання компонентів та

спрощує підтримку програмного коду. На рис. 2.1 представлено приклад компонентної структури вебзастосунку онлайн-кінотеатру.

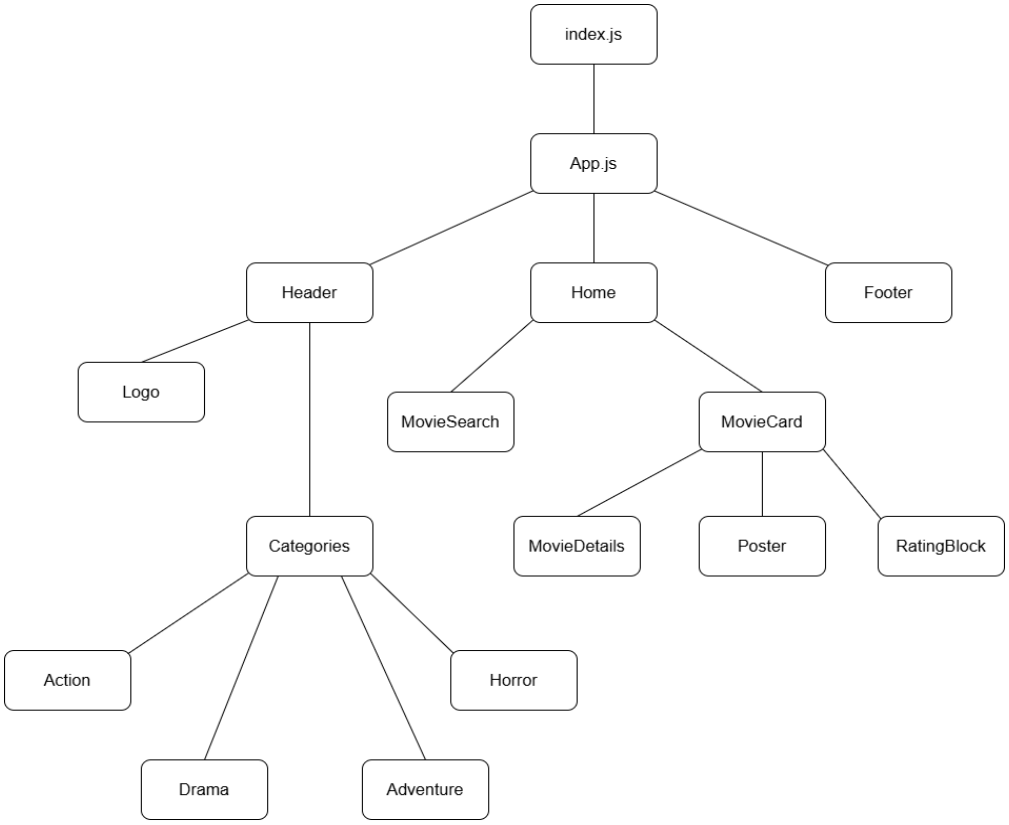

Рисунок 2.1 – Компонентна структура React-застосунку онлайн-кінотеатру

Представлена структура демонструє взаємозв'язок між основними компонентами інтерфейсу та відображає принципи компонентної архітектури React. Використання такого підходу забезпечує масштабованість додатка та спрощує процес модифікації його функціоналу в майбутньому.

Важливим напрямом розвитку сучасної веброзробки є використання TypeScript [9]. Мова програмування TypeScript є надмножиною JavaScript і підтримує статичну типізацію. Використання TypeScript дозволяє значно зменшити кількість помилок у коді, покращити процес тестування та спростити обслуговування великих програмних систем. У останніх наукових публікаціях зазначається, що використання TypeScript позитивно впливає на якість програмного забезпечення та скорочує час, необхідний для виявлення помилок під час розробки.

Для реалізації серверної частини вебзастосунків активно використовується Node.js [14]. Дане середовище виконання дозволяє застосовувати JavaScript на сервері та підтримує асинхронну модель обробки запитів [6]. Асинхронна обробка є ключовою перевагою для систем онлайн-кінотеатрів, оскільки такі системи одночасно обробляють значну кількість запитів користувачів, пов'язаних із відтворенням відео, пошуком контенту, автентифікацією та формуванням рекомендацій.

Для забезпечення взаємодії між клієнтською та серверною частинами застосунку найчастіше використовується REST API [8]. Такий підхід дозволяє стандартизувати обмін даними між компонентами системи та гарантує передачу інформації у форматі JSON. REST-API відрізняються простотою реалізації, незалежністю клієнта від серверної платформи та легкістю інтеграції із зовнішніми сервісами.

У сучасних вебзастосунках особлива увага приділяється зберіганню даних. Для онлайн-кінотеатрів надзвичайно важливо забезпечити безпечне зберігання інформації про користувачів, фільми, жанри, рейтинги, історію переглядів та коментарі. З цією метою доцільним є використання реляційних систем керування базами даних, таких як PostgreSQL або MySQL. PostgreSQL[2, 5] характеризується високою стабільністю, підтримкою складних SQL-запитів та здатністю обробляти великі обсяги даних. MySQL, з іншого боку, простіший у налаштуванні та широко використовується у веб-розробці завдяки високій швидкості виконання запитів.

Системи контролю версій відіграють ключову роль у сучасній розробці програмного забезпечення [11]. Найпопулярнішим рішенням є Git, який дозволяє відстежувати зміни в вихідному коді, організовувати роботу в команді та забезпечувати безпечне внесення змін до проєкту. Використання Git також допомагає автоматизувати процеси тестування та розгортання програмного забезпечення.

Крім основних технологій розробки, сучасні вебзастосунки активно використовують хмарні сервіси та технології контейнеризації. Хмарні

платформи забезпечують масштабованість системи та підвищують доступність сервісів для користувачів. Контейнеризація за допомогою Docker [12] спрощує процес розгортання програмного забезпечення та гарантує єдине середовище виконання на різних серверах.

Важливим аспектом сучасної веброзробки є також забезпечення безпеки застосунків. Для цього використовуються такі механізми, як аутентифікація за допомогою JWT [13], шифрування паролів, а також захист від SQL-ін'єкцій, XSS-атак та CSRF-запитів. У системах онлайн-кінотеатрів безпека має вирішальне значення, оскільки система обробляє особисті дані та облікові записи користувачів.

Для зручності аналізу основні інструменти розробки наведено у вигляді узагальнюючої таблиці 2.1.

Таблиця 2.1 – Порівняння сучасних технологій розробки вебзастосунків

Технологія	Призначення	Переваги	Недоліки
React	Frontend	Висока продуктивність, компонентний підхід	Відносна складність для початківців
TypeScript	Типізація JavaScript	Надійність коду, зручність підтримки	Потребує додаткового налаштування
Node.js	Backend	Асинхронність, масштабованість	Складність роботи з CPU-задачами
PostgreSQL	База даних	Надійність, підтримка складних запитів	Вища складність адміністрування
Git	Контроль версій	Зручність командної роботи	Потребує знань системи версій

Кінець таблиці 2.1

Docker	Контейнеризація	Простота розгортання, ізольованість	Додаткове навантаження на систему

Отже, сучасний стан розвитку вебтехнологій дає змогу створювати високопродуктивні, масштабовані та безпечні вебзастосунки для перегляду мультимедійного контенту. Використання сучасних інструментів розробки забезпечує простоту обслуговування програмного забезпечення, стабільність системи та можливості для подальшого розширення функціоналу онлайн-кінотеатру.

2.2 Аналіз моделей, методів та архітектур розробки вебзастосунків

У процесі розробки сучасних вебзастосунків застосовуються різноманітні архітектурні підходи, моделі та методи, що забезпечують стабільність, масштабованість і підтримуваність програмного забезпечення. Для вебзастосунків онлайн-кінотеатрів особливо важливо ефективно обробляти велику кількість запитів, швидко доставляти мультимедійний контент та забезпечувати безперебійну взаємодію між клієнтською та серверною частинами системи онлайн-кінотеатру.

Одним з основних підходів до створення веб-систем є архітектура «клієнт-сервер» [3, 4]. У рамках цієї моделі система поділяється на дві основні частини: клієнтську та серверну. Клієнтська частина відповідає за взаємодію з користувачем та відображення інтерфейсу, тоді як серверна частина обробляє запити, реалізує бізнес-логіку та керує базою даних. Такий підхід підвищує гнучкість системи та забезпечує можливість незалежного масштабування її компонентів.

Також із основних методів організації взаємодії між клієнтом і сервером є REST API [8]. Такий підхід ґрунтується на принципах клієнт-серверної взаємодії та використовує стандартні методи HTTP для обміну даними. Інформація

зазвичай передається у форматі JSON. Основними перевагами REST API є простота реалізації, висока масштабованість та незалежність клієнтської сторони від реалізації на стороні сервера. Важливим архітектурним шаблоном, що використовується при розробці вебзастосунків, є MVC (Model–View–Controller) [16]. Цей підхід передбачає поділ системи на три логічні компоненти:

– Model – відповідає за роботу з даними та бізнес-логіку;

– View – забезпечує відображення інформації користувачу;

– Controller – обробляє запити користувача та координує взаємодію між іншими компонентами.

Застосування MVC покращує структуру коду, спрощує технічне обслуговування та підвищує масштабованість програмної системи. При розробці сучасного програмного забезпечення особлива увага приділяється архітектурі мікросервісів [15]. На відміну від монолітного підходу, архітектура мікросервісів передбачає поділ системи на набір незалежних сервісів, кожен з яких відповідає за певну функціональність.

Такий підхід підвищує відмовостійкість системи та спрощує її розробку, але вимагає більш складної організації взаємодії між сервісами та додаткових механізмів моніторингу. Для наочного порівняння основних моделей та архітектур подано узагальнюючу таблицю 2.2.

Таблиця 2.2 – Порівняння моделей та архітектур вебзастосунків

Модель / архітектура	**Особливості**	**Переваги**	**Недоліки**
Клієнт-серверна	Поділ на frontend і backend	Простота реалізації	Залежність від сервера
REST API	Обмін даними через HTTP	Гнучкість і масштабованість	Потребує чіткої документації
MVC	Розділення логіки системи	Чітка структура та підтримка	Складність архітектури

Кінець таблиці 2.2

Мікросервіси	Набір незалежних сервісів	Масштабованість і відмовостійкість	Складність адміністрування

Проведений аналіз показує, що при розробці вебзастосунку для онлайн-кінотеатру доцільно використовувати архітектуру «клієнт-сервер». Такий підхід забезпечує чіткий розподіл обов'язків між клієнтськими та серверними компонентами, що покращує структуру системи та спрощує її обслуговування. Використання даної архітектури дозволяє ефективно обробляти запити користувачів, централізовано керувати даними та забезпечувати стабільну роботу застосунку навіть при зростанні навантаження. Загалом це створює основу для високопродуктивної, масштабованої та зручної у користуванні системи онлайн-кінотеатру.

2.3 Специфікація вимог до програмного забезпечення

На основі проведеного аналізу предметної області та сучасних технологічних рішень було сформовано комплекс вимог до вебзастосунку онлайн-кінотеатру. Дані вимоги відображають необхідний рівень функціональності системи, її поведінку під час роботи, а також технічні характеристики, що забезпечують стабільність та зручність у використанні. Формування вимог є одним із ключових етапів проєктування програмного забезпечення, оскільки саме на цьому етапі визначається перелік можливостей майбутньої системи, її обмеження та умови експлуатації. Для вебзастосунку онлайн-кінотеатру особливо важливо знайти баланс між функціональністю, продуктивністю та зручністю використання.

1. Призначення та межі проєкту

1.1 Призначення системи

Вебзастосунок онлайн-кінотеатру призначений для перегляду фільмів та іншого відеоконтенту через мережу Інтернет. Система дозволяє користувачам шукати контент, переглядати відео, створювати списки для перегляду та

отримувати персоналізовані рекомендації. Також система надає інструменти для управління відеоконтентом та даними користувачів.

1.2 Погодження, що ухвалені в програмній документації

– специфікацію сформовано відповідно до стандартів ISO/IEC/IEEE 29148:2018 (вимоги до системної та програмної інженерії) та ISO/IEC 25010:2023 (моделі якості програмного забезпечення);

– для реалізації системи було обрано стек React + TypeScript для клієнтської частини, Node.js для серверної логіки та PostgreSQL як систему управління базами даних;

– проєкт передбачає адаптивний макет, підтримку української мови та використання REST-API для взаємодії між компонентами системи.

1.3 Межі проєкту

Проєкт включає розробку вебзастосунку для онлайн-кінотеатру, зокрема серверних компонентів та бази даних. Передбачається реалізація системи авторизації, функцій перегляду відеоконтенту, пошуку та персоналізації. Проєкт не передбачає розробки окремих мобільних застосунків або інтеграції з телевізійними приставками та смарт-телевізорами.

2. Загальний опис:

2.1 Сфера застосування

Система орієнтована на користувачів, які переглядають відеоконтент через веббраузер. Вебзастосунок можна використовувати як онлайн-платформу для перегляду фільмів, трейлерів та іншого мультимедійного контенту.

2.2 Характеристики користувачів

Користувачі системи поділяються на такі категорії:

– гості – користувачі, які переглядають доступний відеоконтент без реєстрації;

– зареєстровані користувачі мають доступ до персонального кабінету, списку обраного та рекомендацій;

– адміністратори здійснюють керування контентом та модерацію системи.

2.3 Загальна структура і склад системи

Система складається з клієнтської частини: React + TypeScript SPA-застосунок; серверної частини: REST API на базі Node.js; бази даних PostgreSQL для зберігання інформації про користувачів, фільми, жанри та історію переглядів.

2.4 Загальні обмеження

Для надійної роботи системи необхідне постійне підключення до Інтернету. Система повинна працювати за протоколом HTTPS і бути сумісною з сучасними веббраузерами. Передбачається підтримка великої кількості одночасних користувачів без істотного зниження продуктивності.

3. Функції системи

3.1. Функція авторизації та аутентифікації

3.1.1 Опис функції

Система підтримує реєстрацію користувачів, доступ до особистих облікових записів та безпечну аутентифікацію.

3.1.2 Вхідна і вихідна інформація

– Вхідні дані: email, пароль.

– Вихідні дані: повідомлення про успішний вхід або помилку, дані користувача.

3.1.3 Функціональні вимоги

Паролі користувачів повинні зберігатися у зашифрованому вигляді. Система повинна підтримувати безпечну передачу даних через HTTPS.

3.2 Функція каталогу та пошуку

3.2.1 Опис функції

Система надає користувачам можливість переглядати каталог фільмів, а також здійснювати пошук і фільтрувати контент.

3.2.2 Вхідна і вихідна інформація

– вхідні дані: назва фільму, жанр, рейтинг.

– вихідні дані: перелік знайденого контенту, інформація про фільм.

3.2.3 Функціональні вимоги

Система повинна забезпечувати швидкий пошук контенту та підтримувати фільтрування за різними критеріями.

3.3 Функція перегляду відеоконтенту

3.3.1 Опис функції

Система забезпечує потокове відтворення відео через вебінтерфейс.

3.3.2 Вхідна і вихідна інформація

– вхідні дані: ID відеоматеріалу.

– вихідні дані: потокове відео, інформація про контент.

3.3.3 Функціональні вимоги

Система повинна забезпечувати стабільну передачу відео та регулювати якість зображення відповідно до швидкості інтернет-з'єднання.

3.4 Функція особистого кабінету

3.4.1 Опис функції

Користувачі можуть переглядати історію переглядів, список улюблених та персоналізовані рекомендації.

3.4.2 Вхідна і вихідна інформація

– вхідні дані: дані користувача (user_id, інформація профілю).

– вихідні дані: історія переглядів, список обраного, персоналізовані дані.

3.4.3 Функціональні вимоги

Система повинна відстежувати історію переглядів та підтримувати персоналізацію контенту.

4. Вимоги до інформаційного забезпечення

4.1 Джерела і зміст вхідної інформації

Основними джерелами інформації для функціонування вебзастосунку онлайн-кінотеатру є дані, що вводяться користувачами та адміністраторами системи. До вхідної інформації належать:

– реєстраційні дані користувачів;

– дані авторизації;

– інформація про фільми;

– жанри, рейтинги та описи контенту;
– коментарі та оцінки користувачів;
– історія переглядів;
– медіафайли та постери.

Інформація повинна оброблятися в режимі реального часу та бути доступною для швидкого пошуку, фільтрації та сортування.

4.2 Нормативно-довідкова інформація

– перелік жанрів фільмів;
– класифікація вікового рейтингу контенту;
– довідники ролей користувачів;
– системні налаштування платформи;
– перелік статусів користувачів та контенту.

4.3 Вимоги до способів організації, збереження та ведення інформації

Зберігання даних повинно здійснюватися у реляційній базі даних PostgreSQL. Для забезпечення цілісності інформації необхідно використовувати механізми резервного копіювання та відновлення даних.

Конфіденційна інформація користувачів повинна передаватися виключно через захищене HTTPS-з’єднання. Паролі користувачів мають зберігатися у зашифрованому вигляді із застосуванням сучасних алгоритмів хешування.

5. Вимоги до технічного забезпечення

Для стабільної роботи вебзастосунку необхідне використання серверного обладнання з постійним доступом до мережі Інтернет. Сервер повинен підтримувати HTTPS-з'єднання та бути здатним обробляти велику кількість одночасних запитів.

6. Вимоги до програмного забезпечення

6.1 Архітектура програмної системи

Для розробки вебзастосунку використовується архітектура «клієнт-сервер». Взаємодія між клієнтськими та серверними компонентами здійснюється через REST API з використанням формату передачі даних JSON.

6.2 Системне програмне забезпечення

Серверна частина системи повинна функціонувати на базі:

– Node.js;

– Nginx;

– PostgreSQL.

6.3 Мережне програмне забезпечення

Для передачі даних використовується протокол HTTPS із підтримкою шифрування TLS. Для реалізації певних інтерактивних функцій системи використовуються з'єднання WebSocket.

6.4 Програмне забезпечення ведення інформаційної бази

– зберігання інформації про користувачів;

– зберігання інформації про контент;

– обробку запитів пошуку;

– підтримку транзакцій.

– підтримку резервного копіювання;

6.5 Мова і технологія розробки

Для реалізації вебзастосунку використовуються:

– React;

– TypeScript;

– Node.js;

– PostgreSQL;

– Tailwind CSS.

Застосування сучасних технологій дозволяє забезпечити масштабованість, продуктивність та зручність підтримки програмного забезпечення.

7 Вимоги до зовнішніх інтерфейсів

7.1 Інтерфейс користувача

Інтерфейс системи має бути інтуїтивно зрозумілим, швидко реагувати на дії користувача та зручним у користуванні для користувачів з будь-яким рівнем

підготовки. Він підтримує різні роздільні здатності екрану та коректно відображається на мобільних пристроях.

7.2 Апаратний інтерфейс

Система не потребує спеціального обладнання і може використовуватися на звичайних комп'ютерах та мобільних пристроях.

7.3 Програмний інтерфейс

Для взаємодії між компонентами системи використовується REST API. Система розроблена з урахуванням можливості інтеграції із зовнішніми сервісами в майбутньому.

7.4 Комунікаційний протокол

Усі запити між клієнтом і сервером повинні передаватися за протоколом HTTPS, щоб забезпечити безпеку даних користувачів.

8. Властивості програмного забезпечення

8.1 Доступність

Система повинна забезпечувати безперебійну роботу та бути доступною для користувачів у будь-який час доби.

8.2 Супроводжуваність

Програмне забезпечення повинно мати модульну структуру коду, що полегшить його обслуговування, тестування та майбутні оновлення системи.

8.3 Переносимість

Система повинна бути сумісною з різними серверними платформами та підтримувати роботу в хмарних середовищах.

8.4 Продуктивність

Вебзастосунок має забезпечувати швидке завантаження сторінок і стабільну роботу навіть за умови великої кількості одночасних користувачів.

8.5 Надійність

Система повинна підтримувати механізми резервного копіювання даних та швидке відновлення у разі можливих збоїв.

8.6 Безпека

Система підтримує HTTPS-з'єднання, захищає особисті дані користувачів та шифрує конфіденційну інформацію.

9. Інші вимоги

Система повинна забезпечувати реєстрацію дій користувачів, підтримку оновлень програмного забезпечення без втрати даних, а також можливість інтеграції із зовнішніми сервісами та платформами в майбутньому.

Висновки до розділу 2

У другому розділі було проведено комплексний аналіз сучасного стану інструментарію, моделей, методів та архітектур розробки вебзастосунків. Встановлено, що сучасні вебсистеми переважно будуються на основі клієнт-серверної архітектури, використання REST API та SPA-підходу, що забезпечує високу продуктивність, масштабованість, гнучкість та зручність підтримки програмного забезпечення. Було розглянуто та проаналізовано сучасні технології розробки вебзастосунків, зокрема React, TypeScript, Node.js, PostgreSQL, Git та Docker. Також у межах розділу проведено аналіз моделей, методів та архітектур розробки вебзастосунків. Розглянуто особливості клієнт-серверної архітектури, SPA-моделі, REST API та MVC-підходу, які забезпечують ефективну організацію взаємодії між компонентами системи та підвищують якість програмного забезпечення. Крім того, було сформовано специфікацію функціональних і нефункціональних вимог до програмного забезпечення онлайн-кінотеатру. Визначені вимоги охоплюють основні функції системи, а також вимоги щодо продуктивності, безпеки, надійності, масштабованості та супроводжуваності програмного забезпечення. Отримані результати створюють основу для подальшого проєктування архітектури системи, побудови UML-діаграм та реалізації вебзастосунку онлайн-кінотеатру.

3 ПРОЄКТУВАННЯ ЗАСТОСУНКУ ОНЛАЙН-КІНОТЕАТРУ

3.1 Розробка UML-діаграм

Для проєктування вебзастосунку онлайн-кінотеатру UML-діаграми використовуються як засіб візуального моделювання структури та поведінки системи. Вони дозволяють розробникам формалізувати вимоги до програмного забезпечення, визначити взаємодію користувачів із системою та спроєктувати ключові програмні компоненти ще до початку реалізації.

Використання UML дає можливість створити цілісне уявлення про майбутній застосунок, описати основні функціональні сценарії, структуру даних та послідовність виконання операцій. Це також спрощує процес розробки, покращує якість проєктних рішень та забезпечує узгодженість між системними вимогами та реалізацією.

Для моделювання вебзастосунку онлайн-кінотеатру планується використання UML-діаграм, наведених у таблиці 3.1.

Таблиця 3.1 – Перелік UML-діаграм, запланованих для застосунку онлайн-кінотеатру

Тип діаграми	**Мета використання**	**Охоплення в системі**
Діаграма прецедентів (Use Case)	Визначення акторів та основних сценаріїв взаємодії з системою	Гість, Користувач, Адміністратор
Діаграма послідовності	Моделювання взаємодії між користувачем і системою під час виконання основних операцій	Авторизація, пошук фільму, перегляд відеоконтенту, додавання до обраного
Діаграма класів	Відображення структури основних об’єктів системи та взаємозв’язків між ними	User, Movie, Genre, Review, Favorite, WatchHistory та інші

Кінець таблиць 3.1

Діаграма діяльності	Опис послідовності дій та логіки виконання бізнес-процесів системи	Пошук фільму, перегляд інформації про фільм, запуск відтворення відеоконтенту
Діаграма розгортання	Відображення фізичної архітектури та взаємодії компонентів системи	Клієнтський застосунок, вебсервер, сервер застосунку, база даних

Запропонований набір UML-діаграм охоплює основні аспекти проєктування вебзастосунку онлайн-кінотеатру. Спочатку за допомогою діаграми прецедентів визначаються учасники та випадки використання системи. Далі за допомогою діаграми класів формується статична структура системи, а потім деталізуються процеси взаємодії між окремими компонентами шляхом побудови діаграм послідовностей. Для моделювання логіки основних функцій системи використовується діаграма активності, яка дозволяє відобразити послідовність дій користувача та реакцію системи на його запити. Завершальним етапом проєктування є побудова діаграми розгортання, яка демонструє фізичну структуру системи, розміщення її компонентів та взаємодію між клієнтською частиною, сервером застосунку та базою даних. Такий підхід забезпечує комплексне уявлення про архітектуру та функціонування вебзастосунку онлайн-кінотеатру та закладає основу для його подальшої реалізації.

3.1.1 Діаграма прецедентів

Діаграма прецедентів для застосунку онлайн-кінотеатру відображає трьох основних учасників системи: Гість, Користувач та Адміністратор. Кожен із них взаємодіє з системою відповідно до наданих йому прав доступу та виконує певні функції. Гість має можливість здійснювати пошук фільмів, переглядати

доступний контент та реєструватися в системі. Після входу в систему користувачі отримують доступ до додаткових функцій, зокрема до можливості додавати фільми до списку обраного, оцінювати переглянутий контент та залишати коментарі. Загальну структуру взаємодії акторів із системою наведено на рис. 3.1.

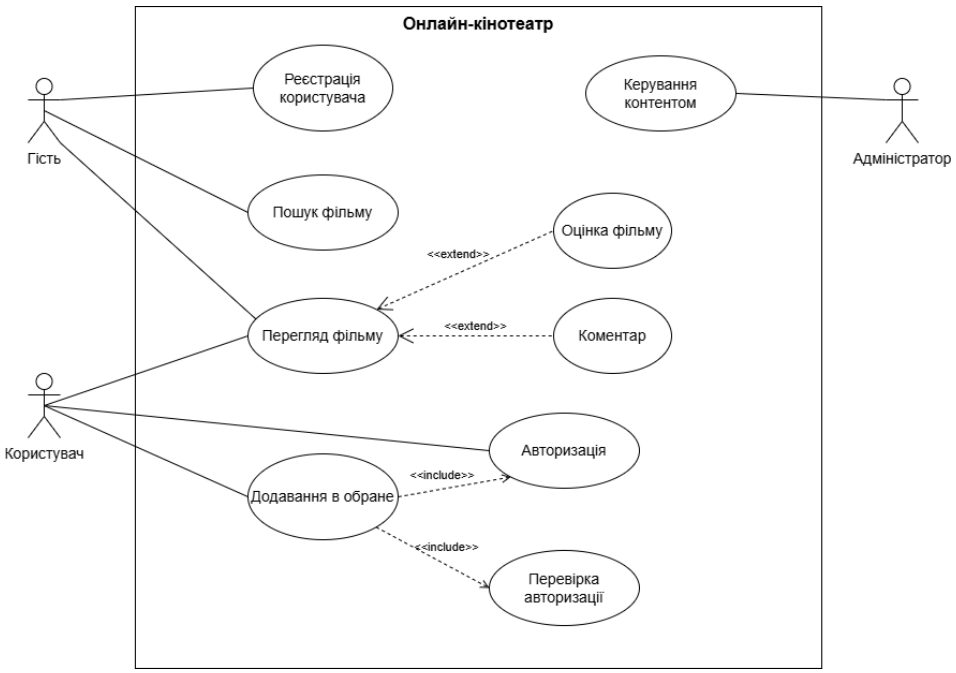

Рисунок 3.1 – Діаграма прецедентів застосунку онлайн-кінотеатру

На діаграмі центральним прецедентом є перегляд фільму, який може бути розширений додатковими функціями, такими як оцінювання фільму та коментування. Для відображення цих залежностей використовуються зв’язки extend, що вказують на необов’язкові сценарії взаємодії користувача з системою. Для виконання окремих дій, зокрема додавання фільмів до обраного, необхідною умовою є проходження авторизації та перевірки облікових даних користувача.

Такі залежності реалізовано за допомогою зв'язків include, які відображають обов’язкові етапи виконання відповідних сценаріїв використання. Адміністратор виконує функції керування контентом системи, включаючи додавання, редагування та видалення інформації про фільми. Діаграма прецедентів допомагає визначити основні функції застосунку для онлайн-

кінотеатру та встановити взаємозв'язки між учасниками та варіантами використання системи.

3.1.2 Діаграма послідовності

Діаграми послідовності (Sequence Diagrams) використовуються для відображення взаємодії між об'єктами системи у межах конкретного сценарію. Вони дають змогу відстежувати обмін повідомленнями між користувачем, програмними компонентами та базою даних, а також визначати послідовність операцій під час реалізації функціональних можливостей вебзастосунку.

Для застосунку онлайн-кінотеатру діаграма послідовності використовується для моделювання основних робочих процесів системи, зокрема авторизації користувачів, пошуку фільмів, перегляду інформації про контент та запуску відтворення відео. Використання таких діаграм дозволяє перевірити правильність взаємодії між клієнтською частиною, сервером та базою даних ще на етапі проєктування.

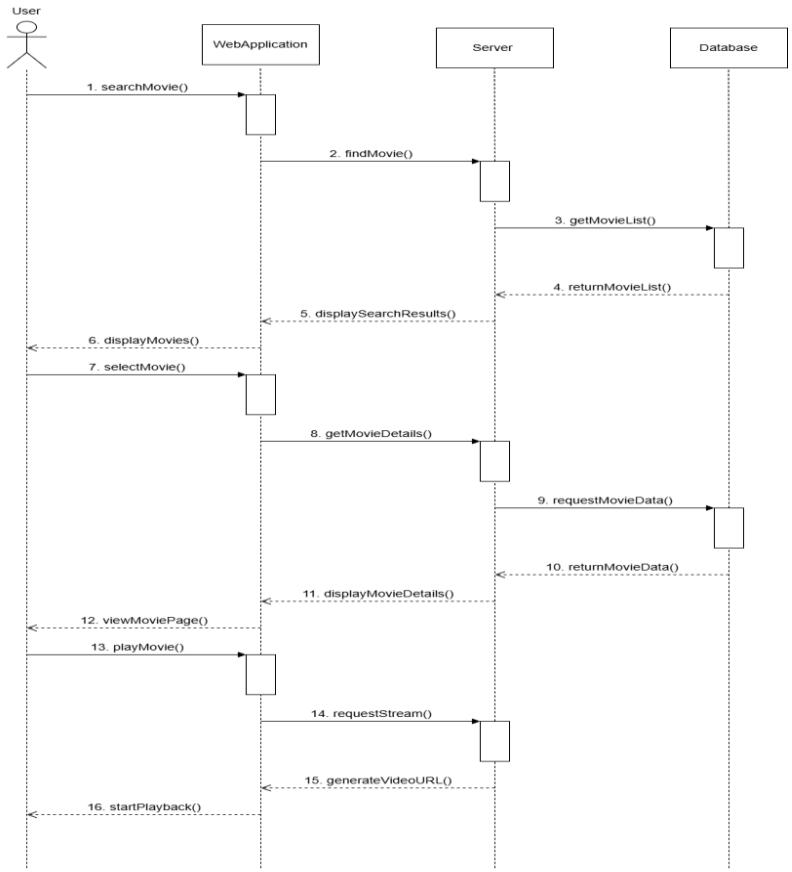

Рисунок 3.2 – Діаграма послідовності процесу перегляду фільму

На діаграмі послідовності відображено процес пошуку та перегляду фільму в системі онлайн-кінотеатру. Користувач здійснює пошук контенту через вебзастосунок, який надсилає відповідні запити на сервер. Сервер отримує необхідні дані з бази даних і повертає результати пошуку. Після вибору фільму користувач отримує інформацію про нього і може розпочати перегляд. Для запуску відтворення вебзастосунок надсилає запит на сервер, який повертає посилання на відеопотік, після чого користувач отримує доступ до контенту.

3.1.3 Діаграма класів

Діаграма класів відображає статичну структуру вебзастосунку онлайн-кінотеатру та визначає основні сутності системи, їхні атрибути, методи та взаємозв'язки (рис. 3.3). Вона використовується для моделювання предметної області та слугує основою для подальшого проєктування бази даних і програмної реалізації застосунку. У межах проєкту було виділено основні класи системи: User, Admin, Movie, Genre, Favorite, WatchHistory, Review та Recommendation.

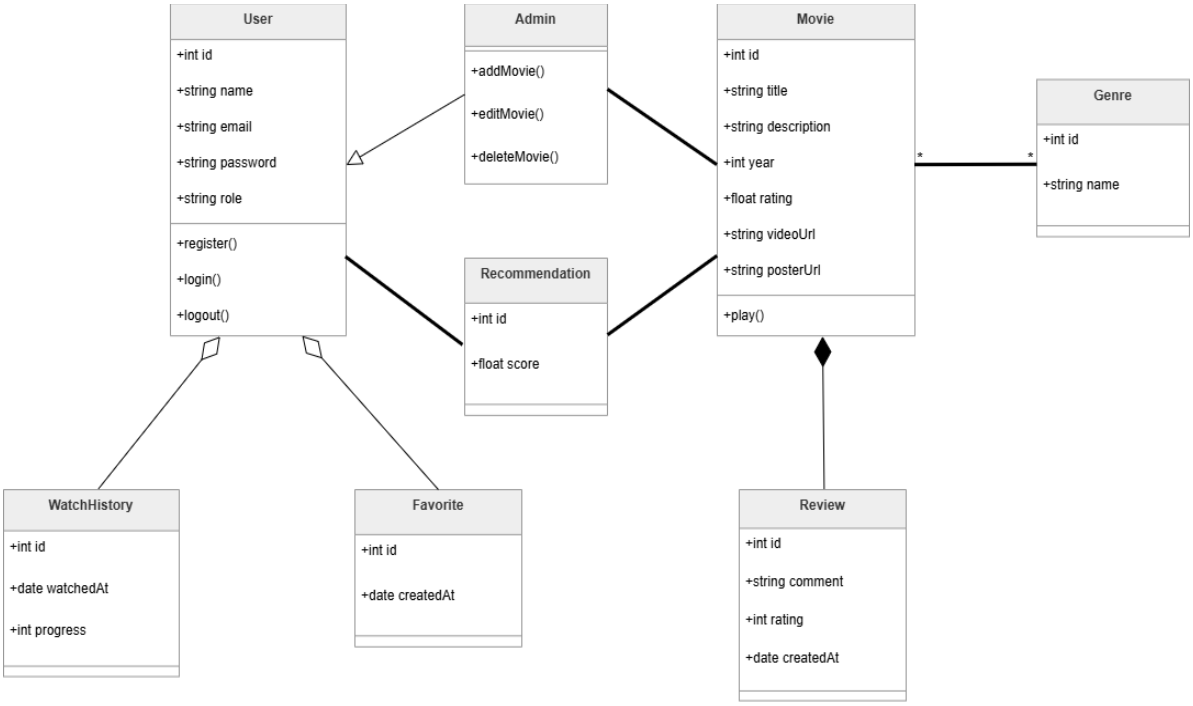

Рисунок 3.3 – Діаграма класів застосунку онлайн-кінотеатру

Клас User містить інформацію про користувачів системи та забезпечує виконання основних операцій, пов'язаних із реєстрацією, авторизацією та

завершенням сеансу роботи. Клас Admin є спеціалізованим типом користувача і дозволяє керувати відеоконтентом шляхом додавання, редагування та видалення інформації про фільми. Клас Movie представляє основний об’єкт системи та містить інформацію про фільми, зокрема назву, опис, рік випуску, рейтинг, посилання на відеофайл та постер. Клас Genre використовується для класифікації контенту, що дозволяє віднести фільми до певних жанрів.

Для реалізації персоналізованого функціоналу застосунку використовуються класи Favorite та WatchHistory. Вони забезпечують можливість зберігати улюблені фільми та відстежувати історію переглядів. Клас Review використовується для зберігання оцінок та коментарів до фільмів, а клас Recommendation призначений для формування рекомендацій на основі уподобань користувачів та рейтингу контенту. Між класами застосовано різні типи зв’язків, зокрема наслідування, асоціацію та композицію. Використання цих взаємозв'язків допомагає забезпечити логічну структуру системи та зберегти цілісність даних під час роботи вебзастосунку.

3.1.4 Діаграма діяльності

Діаграма діяльності відображає послідовність дій, які виконуються користувачем та системою під час реалізації певного функціонального сценарію. Вона надає наочне уявлення про робочий процес програми, точки прийняття рішень, можливі результати та взаємозв’язки між окремими етапами процесу. Для застосунку онлайн-кінотеатру діаграма діяльності використовується для моделювання процесів пошуку та перегляду відеоконтенту, авторизації користувача, додавання фільмів до списку обраного та залишення відгуків. Використання даного типу UML-діаграм дозволяє виявляти потенційні проблеми в логіці системи ще на етапі проєктування та забезпечувати коректну реалізацію функціональних вимог. На рис. 3.4 наведено діаграму діяльності процесу перегляду фільму користувачем.

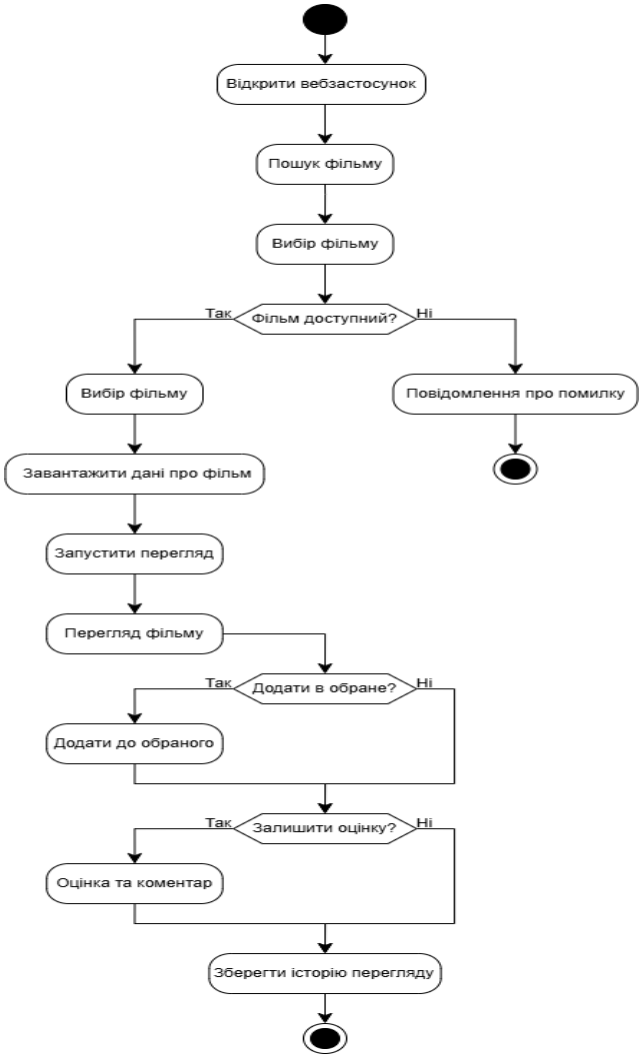

Рисунок 3.4 – Діаграма діяльності процесу перегляду фільму

Процес розпочинається з відкриття вебзастосунку та пошуку необхідного фільму. Після вибору контенту система перевіряє наявність відеофайлу. Якщо фільм доступний, користувач може розпочати його перегляд. У разі відсутності контенту або виникнення помилки відображається відповідне повідомлення.

Після початку відтворення користувачі можуть переглядати фільм, додавати його до списку улюблених, а також залишати оцінки та коментарі. Після завершення перегляду фільму система зберігає цю інформацію в історії переглядів користувача. Таким чином діаграма діяльності відображає повний цикл взаємодії користувача з відеоконтентом та демонструє основні етапи роботи застосунку онлайн-кінотеатру.

3.1.5 Діаграма розгортання

Діаграма розгортання відображає фізичну архітектуру застосунку онлайн-кінотеатру та демонструє розміщення його основних компонентів у мережевому середовищі (рис. 3.5).

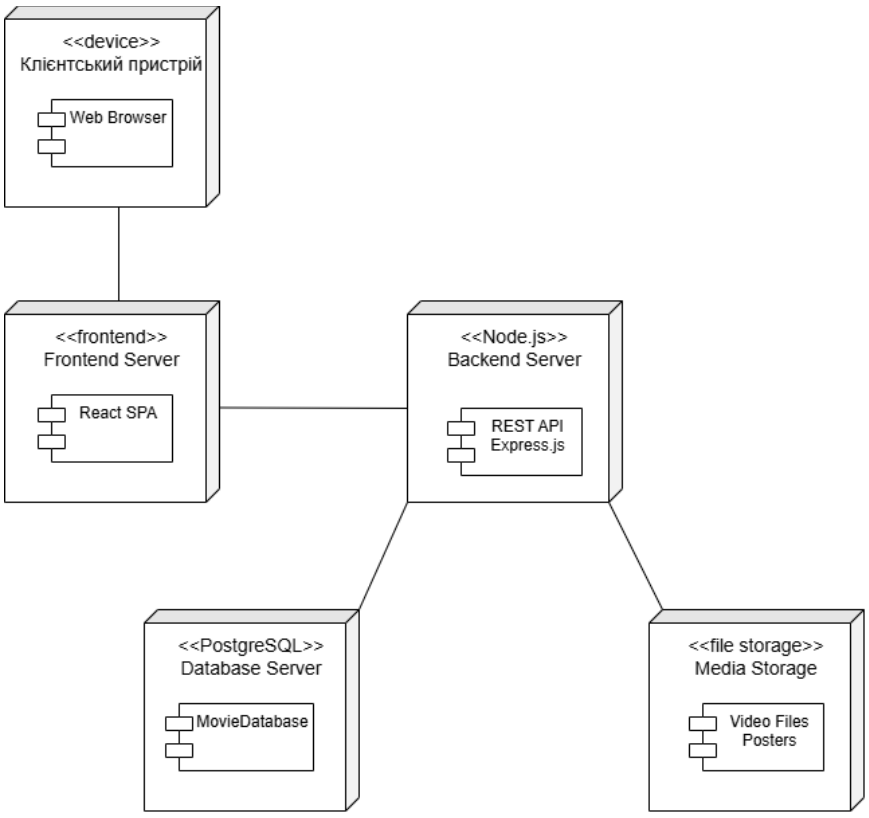

Рисунок 3.5 – Діаграма розгортання застосунку онлайн-кінотеатру

На стороні користувача розташований клієнтський пристрій (ClientDevice), який взаємодіє з системою через веббраузер. Саме браузер відображає користувацький інтерфейс і надсилає запити на серверну частину застосунку. Клієнтська частина системи представлена сервером FrontendServer, на якому розгорнуто односторінковий веб-додаток (React SPA). Даний компонент відповідає за візуалізацію сторінок, навігацію між розділами, обробку дій користувачів та взаємодію з серверним API.

Основна бізнес-логіка реалізована на сервері BackendServer, який працює на платформі Node.js із використанням Express.js. Сервер забезпечує обробку запитів користувачів, виконання операцій авторизації та автентифікації, пошук

фільмів, формування рекомендацій, керування списком обраного, а також взаємодію з базою даних і файловим сховищем через REST API. Для зберігання структурованих даних використовується сервер бази даних DatabaseServer, побудований на основі PostgreSQL. База даних містить інформацію про користувачів, фільми, жанри, історію переглядів, оцінки, коментарі та інші дані, необхідні для функціонування системи.

Окремим компонентом архітектури є сховище файлів MediaStorage, яке містить відеофайли фільмів та графічні матеріали, зокрема постери. Використання окремого сховища зменшує навантаження на базу даних та забезпечує ефективне управління мультимедійним контентом. Взаємодія між усіма компонентами системи здійснюється через мережеві з'єднання. Користувачі отримують доступ до функціоналу онлайн-кінотеатру через браузер, фронтенд взаємодіє із сервером застосунку за допомогою HTTP-запитів, а бекенд забезпечує обмін даними з базою даних та файловим сховищем. Така архітектура відповідає моделі «клієнт-сервер» і гарантує масштабованість, надійність та простоту обслуговування програмного забезпечення.

3.2 Проєктування архітектури системи

Розділ варіантів використання описує взаємодію користувачів із застосунком онлайн-кінотеатру у вигляді прецедентів (Use Cases), які визначають очікувану поведінку системи без деталізації внутрішньої реалізації. Кожен прецедент відображає послідовність дій користувача та відповідь системи, що дозволяє визначити функціональний обсяг програмного забезпечення. Основними акторами системи є гість, зареєстрований користувач та адміністратор. Вони взаємодіють із вебзастосунком через браузер, виконуючи такі дії, як реєстрація, авторизація, пошук фільмів, перегляд відеоконтенту, додавання фільмів до списку обраного, оцінювання та коментування контенту, а також керування каталогом фільмів. Сукупність розглянутих прецедентів є основою для деталізації функціональних вимог, побудови тестових сценаріїв та подальшого проєктування системи.

Таблиця 3.2 – Реєстрація користувача

Usecase section	**Comment**
Use Case Name	Реєстрація користувача
Scope	System
Level	User-goal
Primary Actor	Гість
Stakeholders and interests	Гість бажає створити обліковий запис, щоб отримати доступ до всіх функцій.
Preconditions	Користувач не має зареєстрованого облікового запису в системі.
Success guarantee	Обліковий запис успішно створено, користувач може авторизуватися в системі.
Main Success Scenario	1. Гість переходить на сторінку реєстрації. 2. Вводить ім'я, електронну пошту та пароль. 3. Підтверджує реєстрацію. 4. Система перевіряє коректність введених даних. 5. Створюється новий обліковий запис. 6. Реєстрація успішно завершена.
Extensions	Вказана електронна пошта вже використовується іншим користувачем. Система відображає повідомлення про помилку та пропонує виконати авторизацію.
Special Requirements	Паролі повинні зберігатися у зашифрованому вигляді.
Technology and Data Variations List	Дані користувачів зберігаються в базі даних PostgreSQL.
Frequency of Occurrence	10–20 %
Miscellaneous	У майбутньому можливе впровадження двофакторної аутентифікації.

Після успішного виконання сценарію реєстрації користувач отримує особистий обліковий запис, який надає доступ до всіх функцій онлайн-кінотеатру. Процес реєстрації є початковим етапом взаємодії користувача з системою та забезпечує його ідентифікацію під час подальшого користування сервісом. У ході виконання прецеденту система перевіряє введені дані на відповідність встановленим правилам перевірки. Зокрема, перевіряється коректність формату електронної пошти, складність пароля та унікальність облікового запису. У разі виявлення будь-яких помилок користувач отримує інформаційні повідомлення з рекомендаціями щодо їх виправлення.

Особлива увага приділяється безпеці персональних даних. Паролі користувачів не зберігаються у вигляді звичайного тексту, а обробляються за допомогою сучасних алгоритмів хешування. Це дозволяє мінімізувати ризик несанкціонованого доступу до облікових записів навіть у випадку порушення безпеки бази даних.

Реалізація даного сценарію сприяє формуванню персоналізованого користувацького середовища, у межах якого користувачі можуть створювати власні списки улюблених фільмів, залишати оцінки та коментарі, а також отримувати рекомендації на основі своїх уподобань. У майбутньому систему може бути розширено, щоб забезпечити підтримку двофакторної аутентифікації, підтвердження електронної пошти та інтеграцію із зовнішніми сервісами авторизації.

Таблиця 3.3 – Пошук фільму

Usecase section	**Comment**
Use Case Name	Пошук фільму
Scope	System
Level	User-goal
Primary Actor	Гість, Користувач
Stakeholders and interests	Користувач бажає швидко знайти фільм, який хоче подивитися.

Кінець таблиці 3.3

Preconditions	Система містить каталог фільмів і доступна для користувачів.
Success guarantee	Користувач отримує список фільмів, що відповідають пошуковому запиту.
Main Success Scenario	1. Користувач відкриває головну сторінку застосунку. 2. Користувач вводить назву фільму або ключове слово у поле пошуку. 3. Система виконує пошук у каталозі фільмів. 4. Відображається список знайдених результатів. 5. Користувач відкриває сторінку обраного фільму, щоб переглянути детальну інформацію.
Extensions	За введеним запитом не знайдено жодного фільму. Система відображає відповідне повідомлення та пропонує схожі результати або популярні фільми.
Special Requirements	Час виконання пошуку не повинен тривати довше 2 секунд.
Technology and Data Variations List	Пошук здійснюється через REST API з використанням бази даних PostgreSQL.
Frequency of Occurrence	40–60 %
Miscellaneous	Передбачена можливість додавання фільтрації за жанром, роком випуску та рейтингом.

Пошук фільму є однією з ключових функцій онлайн-кінотеатру, оскільки забезпечує користувачам швидкий доступ до потрібного їм відеоконтенту. Ефективність реалізації даного прецеденту впливає на зручність користування системою та задоволеність користувачів. У процесі пошуку система аналізує запит користувача та шукає відповідні записи в каталозі фільмів. Результати можуть формуватися на основі повної або часткової відповідності назви фільму,

ключових слів, жанру чи інших характеристик контенту. Після завершення обробки запиту користувачеві відображається список знайдених фільмів із короткою інформацією про кожен із них. Для підвищення продуктивності та якості пошуку можуть застосовуватися механізми індексування бази даних, кешування популярних запитів та алгоритми пошуку схожих результатів. Саме такі рішення забезпечують стабільну роботу системи навіть за умови великої кількості користувачів та значного обсягу даних.

У випадку відсутності результатів система система повідомляє про це користувача та пропонує альтернативні варіанти, наприклад популярні фільми, новинки або матеріали на схожу тематику. У подальшому функціональність пошуку може бути розширений шляхом додавання фільтрів за жанром, роком випуску, рейтингом, країною виробництва та іншими параметрами, що значно підвищить точність пошуку та зручність навігації каталогом.

3.3 Вибір технологічного стеку

Під час розробки застосунку онлайн-кінотеатру було обрано сучасний стек вебтехнологій, який забезпечує високу продуктивність, простоту розробки та можливість масштабування системи в майбутньому. Вибір технологій здійснювався з урахуванням вимог до програмного забезпечення, сформованих у другому розділі роботи. Архітектура застосунку побудована на моделі «клієнт-сервер». Такий підхід дозволяє розділити користувацький інтерфейс, серверну логіку та рівень зберігання даних. Завдяки цьому підвищується гнучкість системи, спрощується її обслуговування та забезпечується можливість незалежного розвитку окремих компонентів.

Для реалізації клієнтської частини було обрано бібліотеку React. Використання компонентного підходу дозволяє створювати багаторазові елементи інтерфейсу та забезпечує швидке оновлення даних на сторінках без необхідності їх повного перезавантаження. Для розробки застосунку використовується мова TypeScript, яка забезпечує статичну типізацію та підвищує надійність коду. Серверна частина реалізується платформі Node.js із

використанням фреймворку Express.js. Таке поєднання дозволяє створити REST API для забезпечення взаємодії між клієнтською частиною та базою даних. Сервер відповідає за обробку запитів користувачів, авторизацію, управління каталогом фільмів, збереження історії переглядів та управління контентом. Для зберігання інформації використовується система керування базами даних PostgreSQL. База даних містить інформацію про користувачів, фільми, жанри, оцінки, коментарі та список обраного. PostgreSQL забезпечує високу надійність даних, підтримку транзакцій та ефективну обробку великих обсягів даних.

Таблиця 3.4 – Стек технологій застосунку онлайн-кінотеатру

Категорія	**Обрана технологія**	**Призначення**
Мова програмування	TypeScript	Розробка клієнтської та серверної частини
Frontend	React	Створення користувацького інтерфейсу
Backend	Node.js	Виконання серверної логіки
API Framework	Express.js	Реалізація REST API
База даних	PostgreSQL	Зберігання даних системи
Контроль версій	Git	Керування версіями проєкту
Інструмент розробки	Visual Studio Code	Написання та редагування програмного коду

Обраний технологічний стек дозволяє реалізувати основний функціонал застосунку онлайн-кінотеатру, включаючи авторизацію користувачів, перегляд каталогу фільмів, пошук контенту, збереження списку для перегляду та керування відеофайлами. Використання React, Node.js, Express.js та PostgreSQL забезпечує високу продуктивність системи, простоту обслуговування та можливість подальшого розширення функціональних можливостей. У майбутньому до системи можна буде додати модулі рекомендацій, системи рейтингування та механізми персоналізації контенту без істотних змін у існуючій архітектурі.

3.4 Мокапи інтерфейсу користувача

Для візуалізації основних екранів застосунку онлайн-кінотеатру на етапі проєктування було розроблено набір мокапів інтерфейсу користувача. Мокапи дозволяють провести попередню оцінку структури сторінок, розміщення елементів керування, логіки навігації та зручності користування без необхідності створювати повноцінний програмний інтерфейс. Розроблені макети охоплюють ключові функції системи, зокрема зокрема реєстрацію та авторизацію користувачів, пошук і перегляд відеоконтенту, роботу зі списком обраних фільмів, а також адміністрування контенту. Створені мокапи виступають основою для подальшої розробки користувацького інтерфейсу та допомагають виявити можливі недоліки в структурі застосунку ще до початку етапу програмної реалізації.

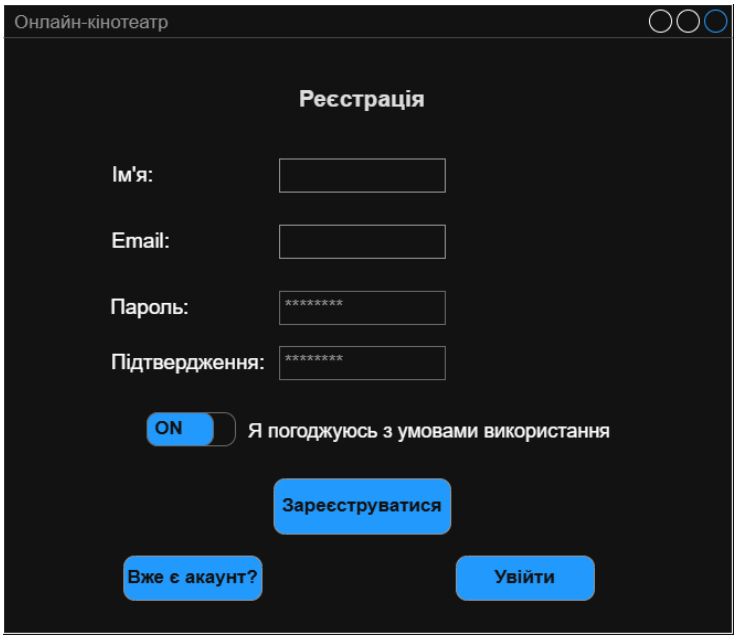

Рисунок 3.6 – Мокап сторінки реєстрації користувача

На сторінці реєстрації користувача реалізовано форму для створення нового облікового запису в системі онлайн-кінотеатру (рис. 3.6). Інтерфейс містить поля для введення імені користувача, адреси електронної пошти, пароля та підтвердження пароля. Також передбачено можливість підтвердження згоди з умовами використання сервісу. У нижній частині сторінки розташовані кнопки

для реєстрації нового користувача та переходу на сторінку входу. Після перевірки введених даних система створює новий обліковий запис.

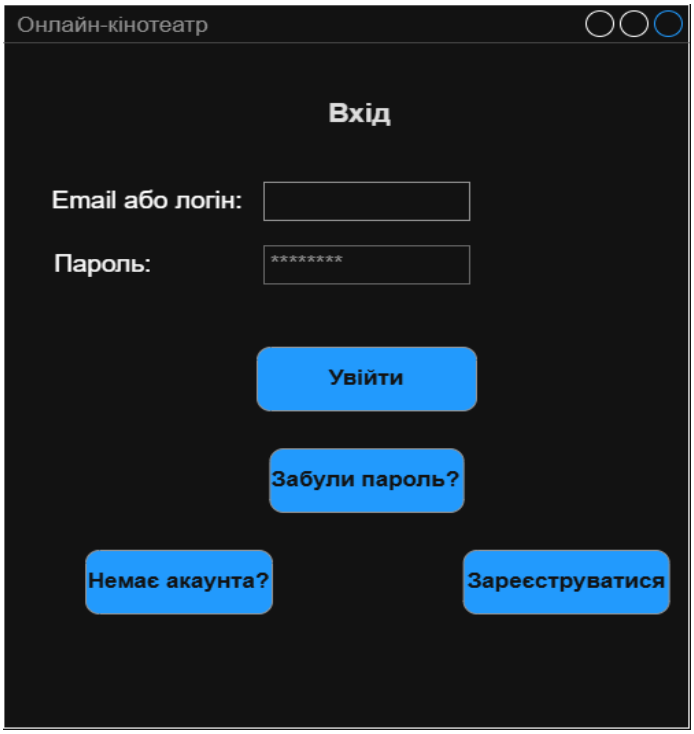

Рисунок 3.7 – Мокап сторінки авторизації

На сторінці авторизації користувач може виконати вхід до системи онлайн-кінотеатру, вказавши своє ім’я користувача або адресу електронної пошти та пароль (рис. 3.7). Для входу передбачена кнопка «Увійти», а також можливість відновлення пароля. Крім того, інтерфейс містить елементи переходу до сторінки реєстрації нового користувача. Після успішної перевірки облікових даних система надає доступ до функціональних можливостей сервісу.

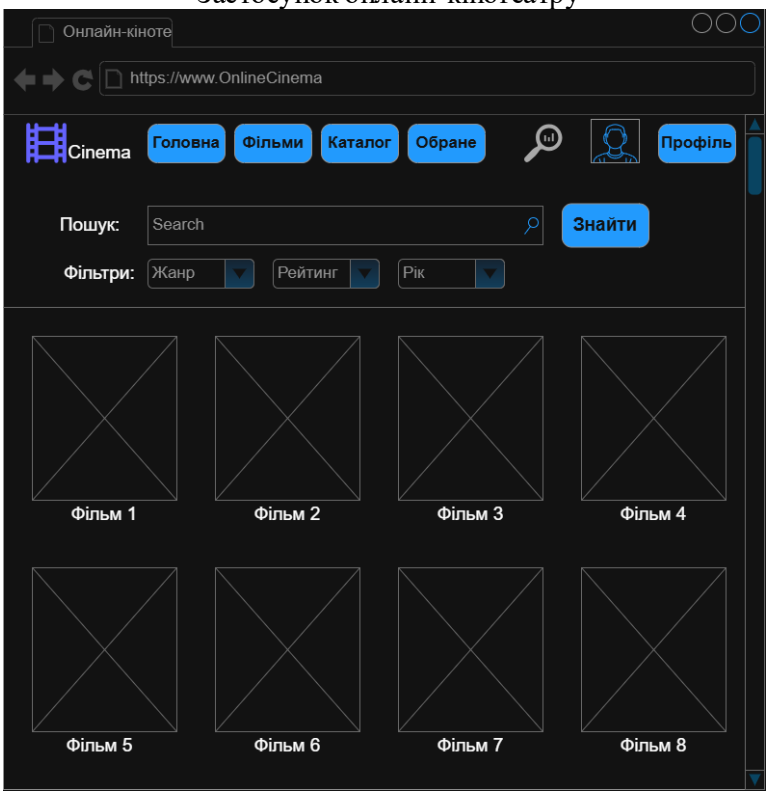

Рисунок 3.8 – Мокап головної сторінки онлайн-кінотеатру

На головній сторінці онлайн-кінотеатру реалізовано засоби пошуку та навігації по каталогу відеоконтенту (рис. 3.8). У верхній частині інтерфейсу розміщено навігаційне меню, рядок пошуку та фільтри для вибору фільмів за жанром, рейтингом і роком випуску. Основну область сторінки займає каталог фільмів, представлений у вигляді карток із постерами та назвами. Така структура забезпечує зручний пошук і перегляд доступного контенту.

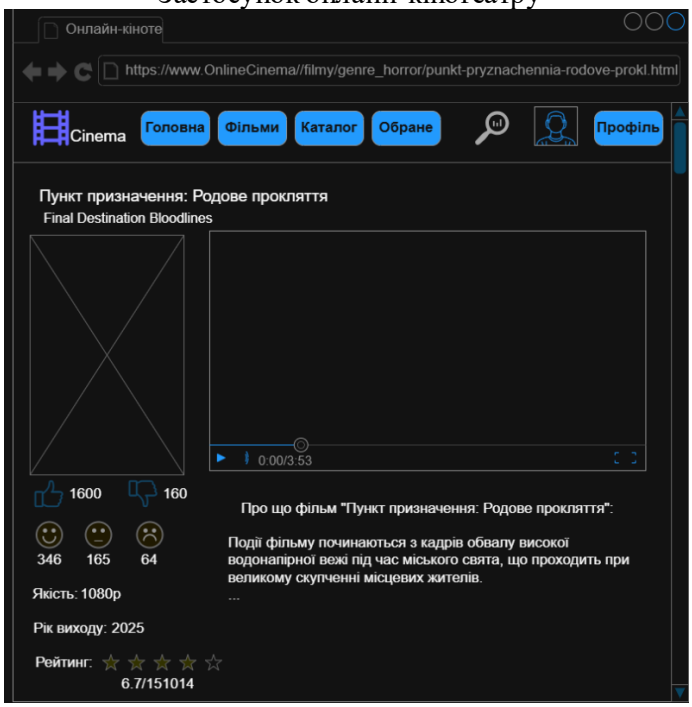

Рисунок 3.9 – Мокап сторінки перегляду фільму

На сторінці перегляду фільму користувач може відтворювати відеоконтент та переглядати детальну інформацію про нього (рис. 3.9). Центральним елементом інтерфейсу є відеоплеєр, під яким розміщено назву фільму, короткий опис, якість відео, рік випуску та рейтинг. Також передбачено можливість оцінювання контент за допомогою реакцій та переглядати статистику популярності фільму.

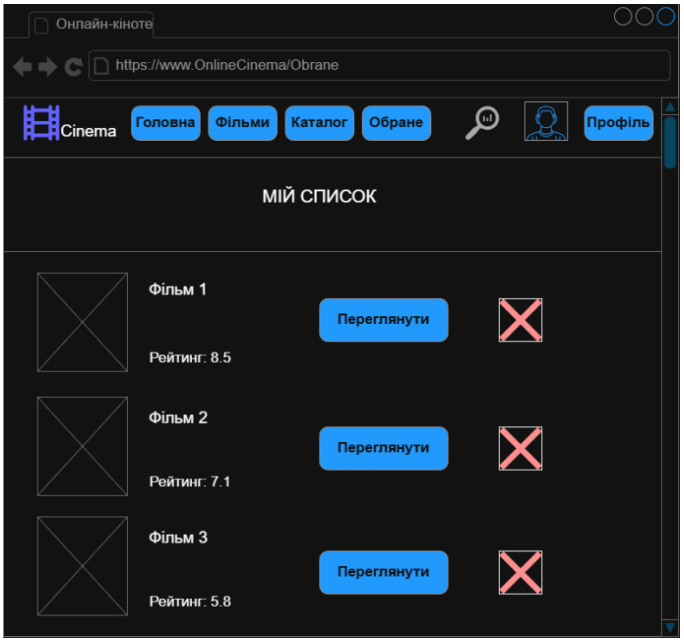

Рисунок 3.10 – Мокап сторінки «Обране»

Сторінка «Обране» відображає список фільмів, який користувач додав для перегляду пізніше (рис. 3.10). Для кожного елемента списку відображаються постер фільму, його назва та рейтинг. Інтерфейс дозволяє користувачам швидко розпочати перегляд обраного контенту або видалити його зі списку. Збережені дані прив'язані до облікового запису користувача, і до них можна отримати доступ після повторного входу в систему.

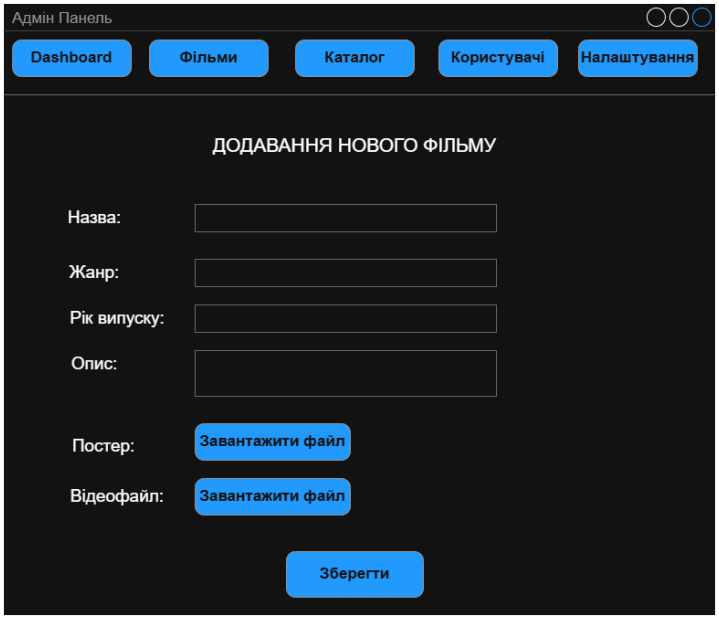

Рисунок 3.11 – Мокап адміністративної панелі

Сторінка адміністративної панелі містить інструменти для управління контентом онлайн-кінотеатру (рис. 3.11). Інтерфейс дозволяє додавати нові фільми до каталогу, редагувати інформацію про наявний контент та виконувати його адміністрування. Форма додавання нового фільму містить поля для введення назви, жанру, року випуску та опису, а також опції для завантаження постера та відеофайлу. Після збереження інформація додається до бази даних і стає доступною для користувачів системи.

Висновки до розділу 3

У третьому розділі було виконано проєктування застосунку онлайн-кінотеатру та визначено основні підходи до його реалізації. Для опису структури та функціонування системи було розроблено UML-діаграми, які зображують

взаємодію користувачів із системою, внутрішню структуру ключових об’єктів, послідовність операцій та логіку бізнес-процесів. Побудовані діаграми дозволили сформувати цілісне уявлення про майбутній програмний продукт ще до початку його розробки.

У процесі проєктування були визначені основні ролі користувачів системи: гість, зареєстрований користувач та адміністратор. Для кожної ролі сформовано набір варіантів використання, що охоплюють пошук фільмів, перегляд відеоконтенту, додавання фільмів до списку обраного, оцінювання контенту та управління каталогом фільмів. Також було розроблено діаграми класів, послідовності, діяльності та розгортання, які деталізують структуру та логіку роботи застосунку. Під час вибору технологічного стеку було визначено використання сучасних вебтехнологій, зокрема React для реалізації клієнтської частини, Node.js та Express.js для серверної логіки, а також PostgreSQL для зберігання даних. Обрані технології забезпечують високу продуктивність, масштабованість та можливість подальшого розширення функціоналу системи.

Крім того, були створені мокапи основних сторінок користувацького інтерфейсу, серед яких сторінки реєстрації та авторизації, головна сторінка каталогу фільмів, сторінка перегляду відеоконтенту, розділ «Обране» та адміністративна панель керування контентом. Розроблені макети дозволили візуалізувати структуру майбутнього застосунку та перевірити зручність його використання. Таким чином, результати проєктування створюють надійну основу для подальшої реалізації застосунку онлайн-кінотеатру, забезпечують узгодженість між функціональними вимогами та технічними рішеннями, а також спрощують процес розробки, тестування та супроводу програмного продукту.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

4.1 Структура та опис програмного коду

Розроблений вебзастосунок онлайн-кінотеатру реалізовано з використанням архітектури «клієнт-сервер». Програмна система складається з трьох основних компонентів: клієнтської частини (Frontend), серверної частини (Backend) та бази даних PostgreSQL. Такий підхід дозволяє розділити логіку візуалізації інтерфейсу, обробку запитів користувачів та зберігання даних, що спрощує подальшу підтримку та розвиток проєкту. Клієнтська частина реалізована з використанням бібліотеки React та мови TypeScript. Вона відповідає за взаємодію користувача з системою, відображення сторінок, пошук фільмів, перегляд інформації про контент, роботу зі списком обраного та авторизацію користувачів.

Усі елементи інтерфейсу побудовані за компонентним підходом, що забезпечує можливість повторного використання коду та спрощує його модифікацію. Серверна частина розроблена на платформі Node.js із використанням фреймворку Express. Backend забезпечує обробку HTTP-запитів, взаємодіє з базою даних і надає дані клієнтській частині через REST API.

Для реалізації основних функцій було створено окремі маршрути для управління фільмами, авторизацією та списком обраного. У якості системи управління базами даних використовується PostgreSQL. База даних зберігає інформацію про користувачів, облікові записи та вибрані фільми. Підключення до бази даних здійснюється за допомогою бібліотеки pg. Структура проєкту організована таким чином, щоб логічно відокремити клієнтську частину від серверної та спростити подальшу розробку. Основні каталоги проєкту наведено в таблиці 4.1.

Таблиця 4.1 – Структура програмного проєкту

Каталог/файл	**Призначення**
frontend/src/pages	Сторінки вебзастосунку
frontend/src/components	Повторно використовувані компоненти інтерфейсу
frontend/src/data	Тимчасові дані про фільми
frontend/src/context	Контекст авторизації користувача
backend/routes	Маршрути REST API
backend/server.js	Основний сервер Express
backend/db.js	Підключення до PostgreSQL
database/cinema.sql	SQL-структура бази даних

Клієнтська частина складається з набору сторінок, що реалізують основні функції онлайн-кінотеатру. На головній сторінці відображається каталог фільмів, а також доступні функції пошуку та фільтрації контенту. Сторінки реєстрації та авторизації відповідають за створення облікових записів та доступ користувачів до системи. На сторінці перегляду фільму відображається детальна інформація про вибраний контент, а сторінка «Обране» дозволяє переглядати список збережених фільмів. Для керування контентом передбачена окрема панель адміністратора. Для наочності основні сторінки застосунку та їх функціональне призначення наведено в таблиці 4.2.

Таблиця 4.2 – Основні сторінки вебзастосунку

Сторінка	**Призначення**
Home.tsx	Головна сторінка та каталог фільмів
Login.tsx	Авторизація користувачів
Register.tsx	Реєстрація користувачів
MovieDetails.tsx	Перегляд інформації про фільм
Favorites.tsx	Список обраних фільмів
AdminPanel.tsx	Панель керування контентом

Для відображення інформації використовуються окремі React-компоненти. Компонент Navbar відповідає за навігацію між сторінками, SearchBar реалізує пошук фільмів, а MovieCard використовується для відображення окремих карток контенту в каталозі. На серверній стороні реалізовано кілька API-маршрутів. Маршрут movies.js забезпечує отримання інформації про фільми, auth.js відповідає за реєстрацію користувачів, а favorites.js реалізує роботу зі списком обраного. Таке розділення дозволяє ізолювати функціональні модулі та спрощує підтримку коду.

Під час розробки використовувався на принципі модульності, згідно з яким кожна функціональна частина системи реалізована у вигляді окремого файлу або компонента. Це дозволяє легко розширювати функціональність застосунку без внесення значних змін у вже реалізовані модулі. У результаті сформовано чітку структуру програмного коду, яка відповідає вимогам сучасних вебзастосунків і забезпечує майбутню масштабованість системи.

4.2 Реалізація основних функцій системи

У цьому розділі наведено детальний опис реалізації основних функціональних можливостей веб-додатку для онлайн-кінотеатру, розробленого з використанням технологій React + TypeScript на клієнтській стороні, Node.js + Express на сервері та PostgreSQL як системи управління базою даних. Архітектура системи базується на моделі «клієнт-сервер», де фронтенд відповідає за користувацький інтерфейс та взаємодію з даними, а бекенд – за обробку запитів, бізнес-логіку та операції з базами даних. Для демонстрації реалізації ключових функцій наведено фрагменти програмного коду, зокрема для основних функціональних можливостей системи.

Реєстрація користувача реалізована за допомогою форми Register.tsx, яка дозволяє користувачу вводити основні дані: ім’я, адресу електронної пошти та пароль. Після заповнення форми дані надсилаються на сервер за допомогою методу POST REST API на адресу /api/auth/register. На серверній частині отримані дані обробляються в маршруті auth.js, після чого виконується SQL-

запит для додавання нового користувача до таблиці users у базі даних PostgreSQL. Для забезпечення безпеки використовується параметризований SQL-запит, що мінімізує ризик SQL-ін'єкції та підвищує рівень безпеки системи.

```
router.post("/register", async (req, res) => {
  const { username, email, password } = req.body;

  try {
    await pool.query(
      `INSERT INTO users(username, email, password)
       VALUES($1, $2, $3)
       RETURNING *`,
      [username, email, password]
    );

    res.json({ message: "Користувача створено" });

  } catch (error) {
    res.status(500).json({ message: "Помилка сервера" });
  }
});
```

Після успішного створення облікового запису сервер повертає відповідь із підтвердженням, а користувач перенаправляється на сторінку авторизації. Авторизація є наступним етапом взаємодії користувача із системою та реалізується за допомогою сторінки Login.tsx і методу POST API /api/auth/login. Під час входу система перевіряє, чи відповідають введені дані записам у таблиці users. У випадку успішної перевірки, сервер повертає об'єкт користувача у форматі JSON. Ці дані використовуються на стороні клієнта для збереження стану авторизації, зокрема за допомогою localStorage або контекст стану застосунку.

```
router.post("/login", async (req, res) => {
  try {
    const result = await pool.query(
      `SELECT * FROM users WHERE email = $1 AND password = $2`,
      [req.body.email, req.body.password]
    );

    if (result.rows.length === 0) {
      return res.status(400).json({
        message: "Невірний email або пароль"
      });
```

```
    }

    return res.json({
      message: "Успішний вхід",
      user: result.rows[0]
    });

  } catch (error) {
    return res.status(500).json({ message: "Помилка сервера" });
  }
});
```

Такий підхід дозволяє забезпечити базовий механізм входу користувача та управління сеансами без необхідності використання складної системи токенів. Після авторизації користувач переходить до основного інтерфейсу системи, де реалізовано перегляд та фільтрацію каталогу фільмів. Функціонал пошуку та фільтрування реалізовано на сторінці Home.tsx, яка відповідає за відображення каталогу фільмів. Користувач може здійснювати пошук за назвою фільму, а також застосовувати додаткові фільтри за жанром, мінімальною оцінкою та роком випуску. Це полегшує пошук потрібного контенту в каталозі. Фільтрація реалізована на клієнтській стороні шляхом послідовного застосування умов до масиву фільмів за допомогою методу filter().

```
function Home({ search, genre, rating, year }: any) {
  const [movies, setMovies] = useState<any[]>([]);

  useEffect(() => {
    fetch("http://localhost:5000/api/movies")
      .then(res => res.json())
      .then(data => setMovies(data));
  }, []);
  const filteredMovies = movies.filter((movie) => {
    return (
      movie.title.toLowerCase().includes(search.toLowerCase()) &&
      (genre ? movie.genre === genre : true) &&
      (rating ? movie.rating >= Number(rating) : true) &&
      (year ? movie.year === Number(year) : true)
    );
  });

  return (
    <>
      {filteredMovies.length === 0 ? (
        <p>Фільми не знайдені</p>
      ) : (
        filteredMovies.map((movie) => (
          <MovieCard key={movie.id} movie={movie} />
        ))
      )}
    </>
  );
}
```

Такий підхід дозволяє миттєво оновлювати список фільмів без додаткових запитів до сервера, що підвищує продуктивність і покращує зручність використання. Наступним етапом взаємодії користувача з системою є перегляд детальної інформації про вибраний фільм. Сторінка MovieDetails.tsx призначена для відображення детальної інформації про вибраний фільм, включаючи постер, назву, жанр, рік випуску, рейтинг та опис. Ідентифікатор фільму отримується з параметрів URL за допомогою хука useParams(), після чого виконується пошук відповідного об’єкта в масиві фільмів. Також на сторінці реалізовано простий відеоплеєр за допомогою HTML-елемента <video controls>, який використовується для демонстрації відтворення відео.

```
import { useParams } from "react-router-dom";
import { useState, useEffect } from "react";
function MovieDetails() {
  const { id } = useParams();
  const [movie, setMovie] = useState<any>(null);

  useEffect(() => {
    fetch("http://localhost:5000/api/movies")
      .then(res => res.json())
      .then(data => {
        const found = data.find((m: any) => m.id === Number(id));
        setMovie(found || null);
      });
  }, [id]);

  if (!movie) return <h1>Фільм не знайдено</h1>;

  return (
    <>
      <h1>{movie.title}</h1>
      <p>{movie.description}</p>
    </>
  );
}
export default MovieDetails;
```

Сторінка детальної інформації фільму реалізована з використанням параметрів маршрутизації useParams().Після отримання ідентифікатора фільму до API надсилається запит, і у списку отриманих даних здійснюється пошук відповідного об’єкта. На основі знайденого запису відображається детальна інформація про фільм. Додатково в системі реалізовано можливість взаємодії користувача з фільмом, зокрема його збереження до списку обраного.

Функціонал «Обране» дозволяє користувачу зберігати вподобані фільми для швидкого доступу. Реалізація виконана на сторінках MovieDetails.tsx та Favorites.tsx. Для збереження обраних фільмів використовується localStorage, що дозволяє зберігати дані навіть після оновлення сторінки або браузера. Додавання фільму здійснюється за допомогою функції, яка перевіряє, чи є цей елемент у списку, і за необхідності оновлює його.

```
const addToFavorites = async () => {
  const user = JSON.parse(localStorage.getItem("currentUser") || "null");

  if (!user) return alert("Увійдіть");

  await fetch("http://localhost:5000/api/favorites", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({
      user_id: user.id,
      movie_id: movie.id,
    }),
  });

  alert("Додано в обране");
};
```

Сторінка Favorites.tsx Під час завантаження отримує дані з localStorage та відображає список збережених фільмів. Також реалізовано можливість видалення елементів зі списку.

```
const removeFavorite = async (id: number) => {
  try {
    await fetch(`http://localhost:5000/api/favorites/${id}`, {
      method: "DELETE",
    });

    setFavorites((prev) =>
      prev.filter((item) => item.id !== id)
    );
  } catch (error) {
    console.error("Ошибка удаления:", error);
  }
};
```

Видалення фільму зі списку обраного реалізовано за допомогою HTTP-запиту DELETE до API, після чого оновлюється локальний стан списку без перезавантаження даних із сервера. Таким чином, користувач отримує можливість повноцінно керувати власним списком обраного контенту, додаючи та видаляючи фільми за потреби. Окрім цього, система містить адміністративний інтерфейс для управління каталогом фільмів. Адміністративна панель

реалізована у файлі AdminPanel.tsx і призначена для керування каталогом фільмів. У межах поточної реалізації адміністратор може виконувати три основні операції: додавання нового фільму, редагування існуючого фільму та видалення фільму з каталогу. Дані вводяться через форму, що оновлює стан списку фільмів у застосунку.

Повний програмний код адміністративної панелі наведено у Додатку А. На даний момент доступ до адміністративної панелі не обмежується ролями користувачів, однак у майбутньому планується впровадження системи авторизації розмежуванням правами доступу (адміністратор/користувач). Також функції блокування користувачів та модерації коментарів були відкладені для подальшого розвитку системи.

4.3 Тестування програмного забезпечення

У процесі розробки вебзастосунку було проведено функціональне тестування основних модулів системи. Основною метою тестування є перевірка правильності роботи реалізованих функцій, відповідності їхньої поведінки вимогам технічної специфікації, а також виявлення можливих помилок на етапі розробки. Тестування проводилося в локальному середовищі розробки з використанням браузера Google Chrome та інструментів для виконання REST API-запитів (Fetch). Основна увага приділялася тестуванню взаємодії між клієнтськими та серверними частинами застосунку, а також коректності обробки даних у базі даних PostgreSQL.

Таблиця 4.3 – Результати тестування функцій системи

Функція	**Очікуваний результат**	**Результат**
Реєстрація користувача	Створення нового акаунта в системі	Успішно
Авторизація користувача	Вхід у систему з доступом до персональних даних	Успішно

Кінець таблиці 4.3

Пошук фільму	Відображення фільмів, що відповідають запиту	Успішно
Фільтрація за жанром	Відображення фільмів відповідного жанру	Успішно
Фільтрація за рейтингом і роком	Коректне звуження списку фільмів	Успішно
Перегляд сторінки фільму	Відображення детальної інформації про фільм	Успішно
Додавання до обраного	Збереження фільму у список обраних	Успішно
Видалення з обраного	Видалення фільму зі списку обраних	Успішно
Додавання фільму адміністратором	Новий фільм з’являється в каталозі та доступний користувачам	Успішно
Редагування фільму адміністратором	Оновлення інформації про фільм у системі	Успішно
Видалення фільму адміністратором	Фільм повністю видаляється з каталогу	Успішно

Результати тестування показали, що всі основні функції вебзастосунку працюють стабільно та відповідають заданим вимогам. Під час перевірки було підтверджено належну взаємодію між клієнтською частиною (React), серверною логікою (Node.js + Express) та базою даних PostgreSQL. Також було перевірено сценарії типових дій користувача, включаючи реєстрацію, авторизацію, роботу з каталогом фільмів і списком обраного. Усі тестові випадки були успішно виконані, що свідчить про готовність системи до використання в реальних умовах.

4.4 Керівництво користувача

Цей розділ містить інструкцію користувача для роботи з вебзастосунком онлайн-кінотеатру. У ньому описано основні сценарії використання системи, починаючи з моменту входу на головну сторінку до взаємодії з додатковими функціями, такими як перегляд детальної інформації про фільми, створення списку для перегляду та користування панеллю адміністратора.

Запуск застосунку

Для початку роботи користувач відкриває головну сторінку вебзастосунку у сучасному браузері (Google Chrome, Microsoft Edge або інший сумісний браузер). Після завантаження сторінки користувач отримує доступ до каталогу фільмів та основних функцій системи.

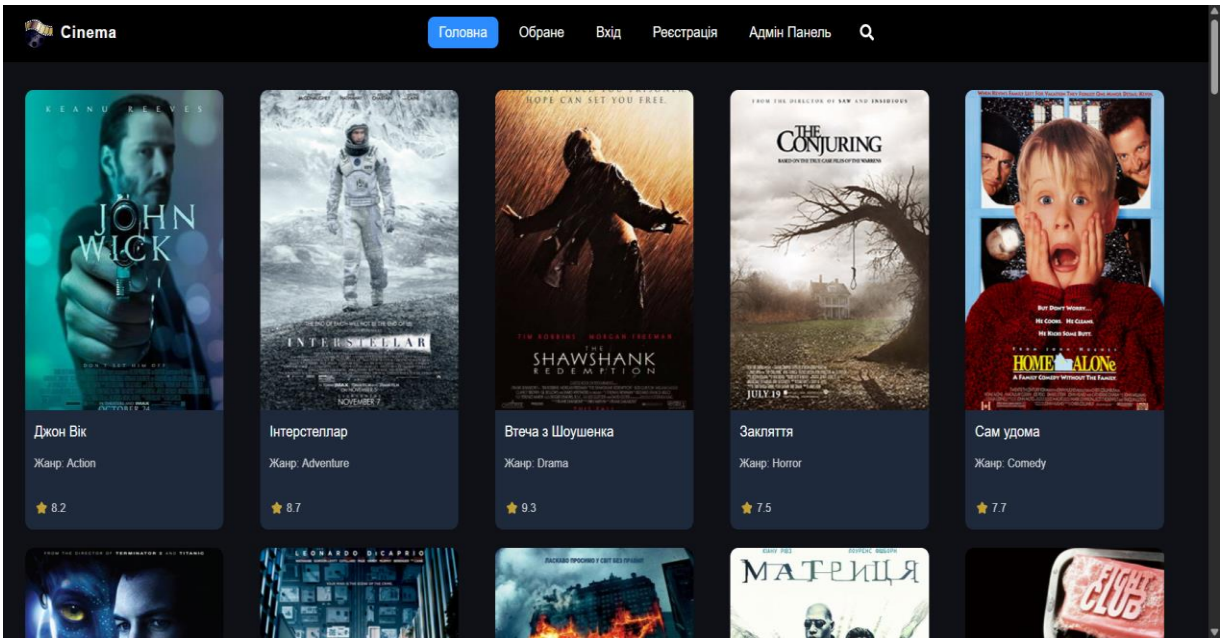

Рисунок 4.1 – Головна сторінка

Сценарій користувача

Для використання повного функціоналу системи користувач повинен створити обліковий запис. На сторінці реєстрації користувач повинен ввести своє ім’я, адресу електронної пошти та пароль. Після підтвердження форма надсилає дані на сервер, де в базі даних PostgreSQL створюється новий запис.

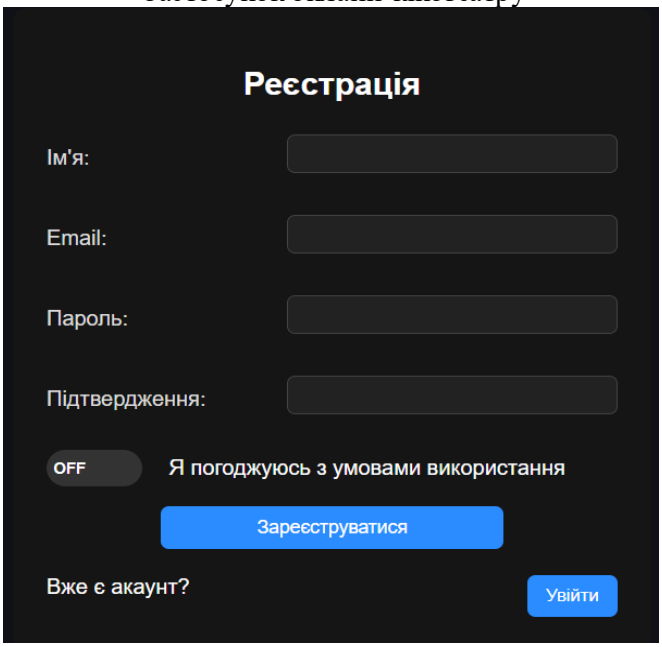

Рисунок 4.2 – Сторінка реєстрації

Після успішної реєстрації користувач може виконати вхід у систему. Під час авторизації введені дані перевіряються, а після успішного входу інформація про користувача зберігається в localStorage, що дозволяє підтримувати сесію без необхідності повторного входу.

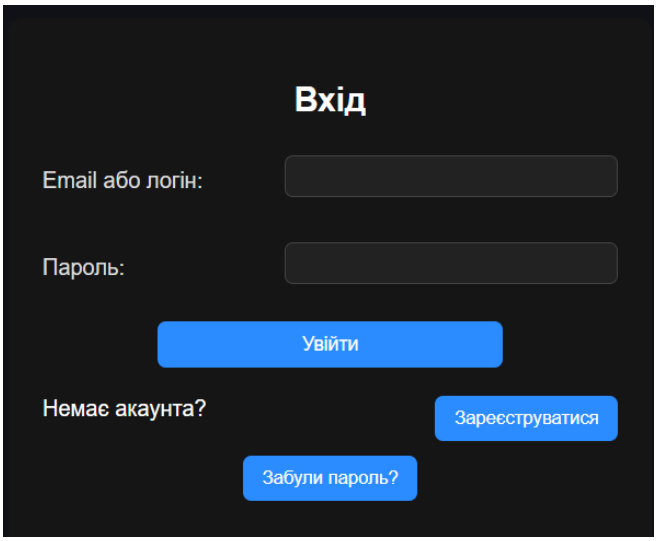

Рисунок 4.3 – Сторінка авторизації

Після входу користувач отримує доступ до каталогу фільмів. На головній сторінці доступна функція пошуку за назвою та фільтри за жанром, рейтингом і роком випуску. Фільтрація виконується динамічно на стороні клієнта без

перезавантаження сторінки, що забезпечує швидку роботу сервісу та зручність використання інтерфейсу.

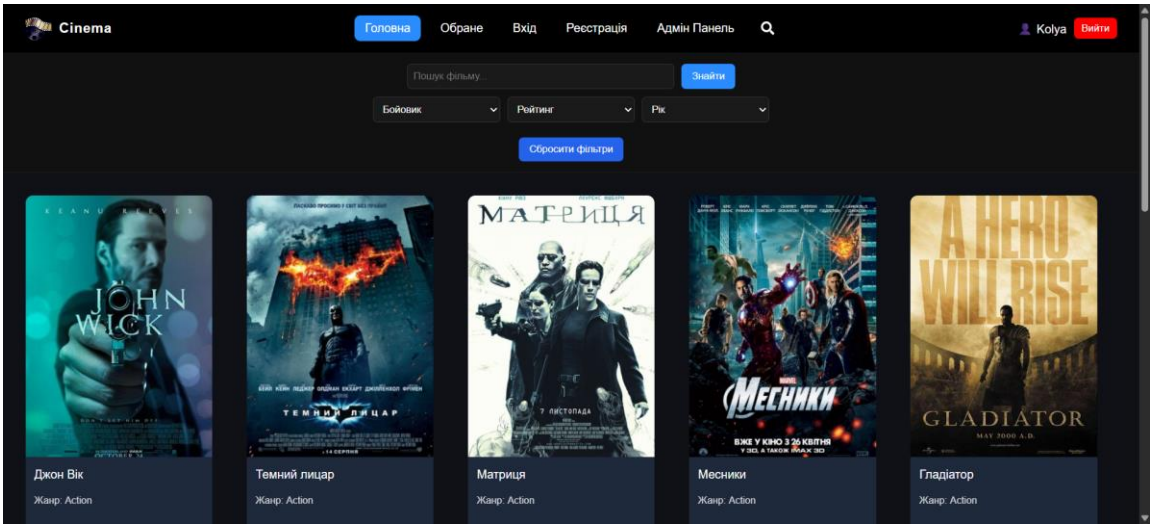

Рисунок 4.4 – Каталог фільмів із фільтрацією

При виборі конкретного фільму користувач переходить на сторінку з детальною інформацією. На цій сторінці відображаються постер, опис, жанр, рік виходу, рейтинг, відеоплеєр для перегляду контенту та коментарі. Сторінка реалізована з використанням параметрів маршрутизації useParams(), після чого надсилається запит до API і відповідний фільм витягується з бази даних.

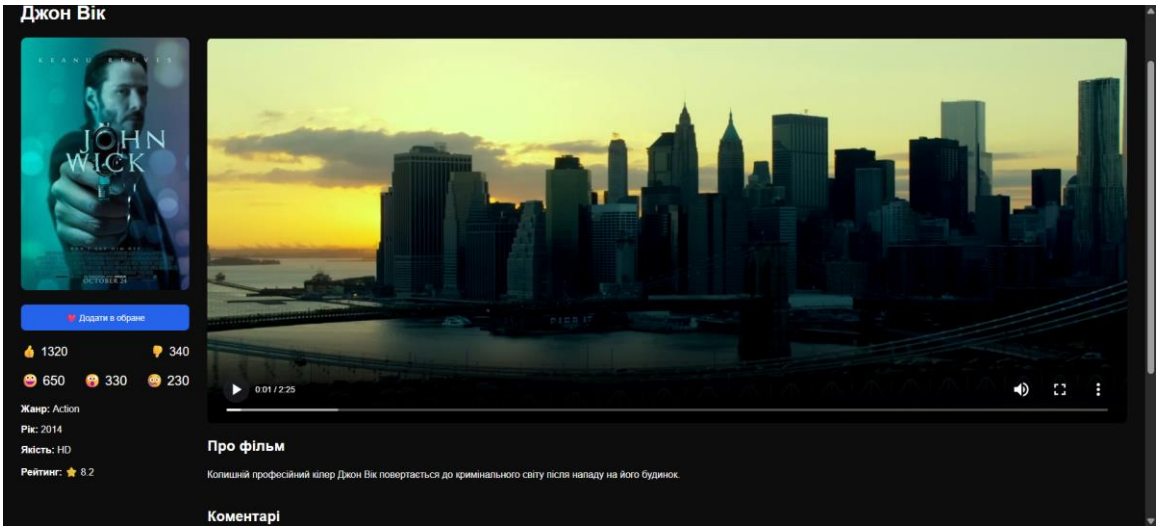

Рисунок 4.5 – Сторінка перегляду фільму

Також користувач має можливість додавати фільми до списку обраного для швидкого доступу. Додані фільми зберігаються та відображаються на окремій сторінці. Також передбачено можливість видалення фільмів зі списку.

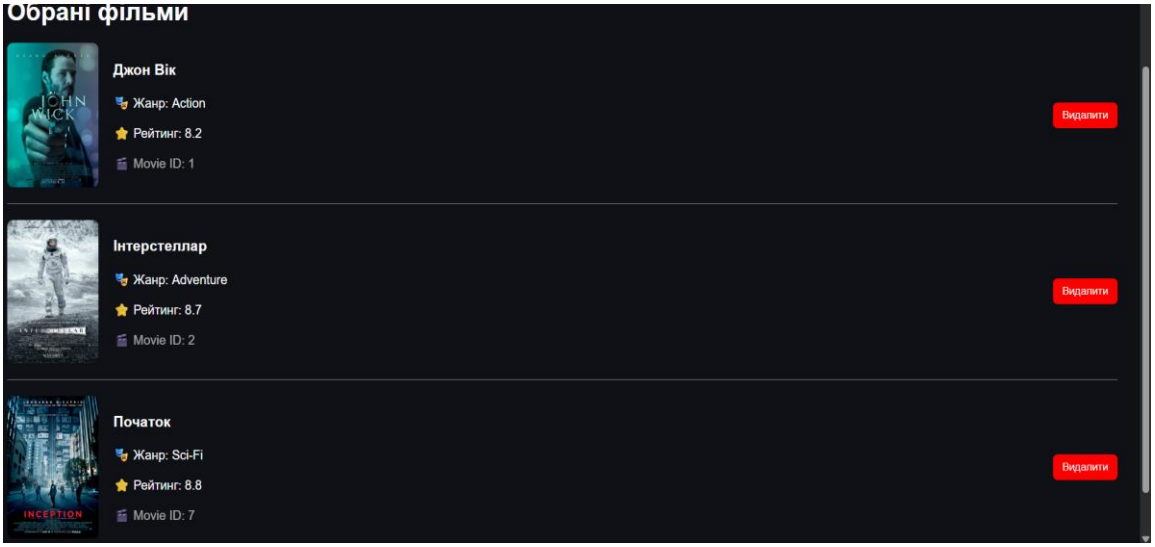

Рисунок 4.6 – Список обраних фільмів

Сценарій адміністратора

Адміністративна панель реалізована у файлі AdminPanel.tsx і призначена для управління каталогом фільмів. У поточній реалізації доступ до панелі адміністратора не обмежується ролями користувачів, тому вона доступна для всіх користувачів системи. Функціонал панелі дозволяє додавати нові фільми, редагувати існуючі записи та видаляти їх із каталогу.

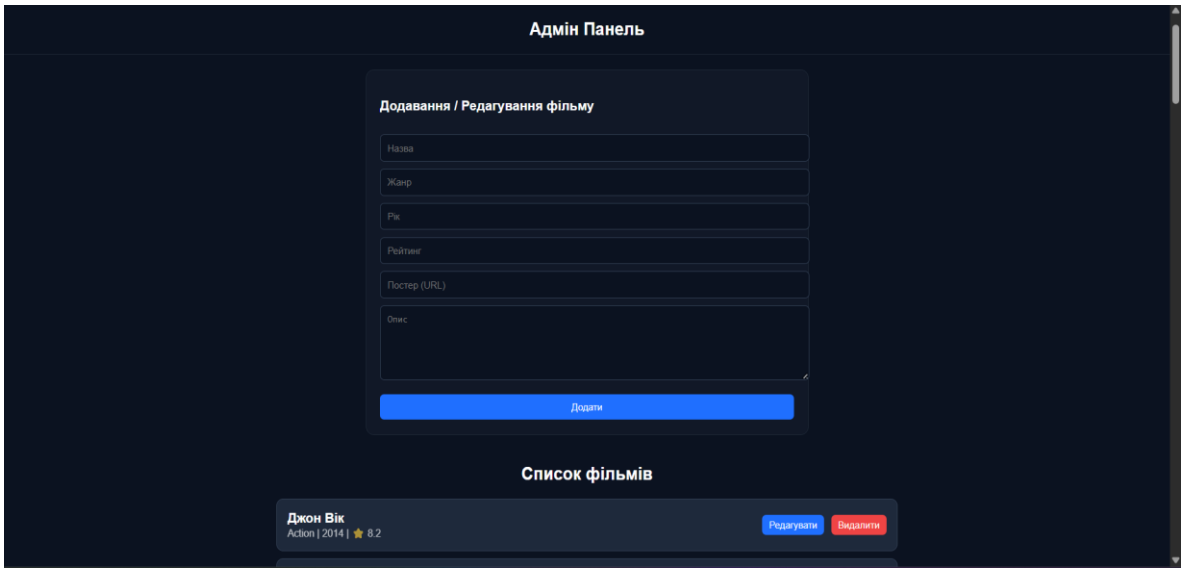

Рисунок 4.7 – Адміністративна панель

У даному розділі наведено керівництво користувача для роботи з вебзастосунком онлайн-кінотеатру. Описані сценарії дозволяють швидко освоїти основні функції системи та ефективно використовувати її можливості. Інтерфейс застосунку є інтуїтивно зрозумілим, що забезпечує зручну навігацію та мінімальну кількість кроків, необхідних для виконання основних операцій.

Висновки до розділу 4

У четвертому розділі детально розглянуто програмну реалізацію вебзастосунку онлайн-кінотеатру. Описано основні функціональні модулі системи, зокрема реєстрації та авторизації користувачів, управління каталогом фільмів, систему пошуку та фільтрації, перегляд детальної інформації про фільми, а також функціонал додавання та видалення фільмів зі списку обраного. Окрему увагу приділено адміністративній панелі, яка дозволяє користувачам керувати каталогом фільмів шляхом додавання, редагування та видалення записів.

Незважаючи на відсутність розмежування ролей у поточній версії системи, функціонал адміністратора доступний користувачам і забезпечує повну підтримку операцій з управління контентом. Також проведено тестування основних функцій системи, яке підтвердило належну роботу всіх ключових модулів та стабільну взаємодію між клієнтською частиною (React), серверною логікою (Node.js + Express) та базою даних PostgreSQL.

Керівництво користувача, сформоване для системи, описує основні сценарії використання та забезпечує легке освоєння функціоналу для кінцевого користувача. Таким чином, програмна частина проєкту пройшла етапи реалізації, тестування та документації, що підтверджує готовність системи до подальшого використання та можливого розширення її функціональних можливостей.

ВИСНОВКИ

У кваліфікаційній роботі було розроблено вебзастосунок онлайн-кінотеатру, призначений для перегляду та управління мультимедійним контентом. У ході виконання роботи було проведено дослідження предметної області, пов'язаної з розробкою вебзастосунків для перегляду мультимедійного контенту. У ході дослідження було визначено актуальність створення онлайн-кінотеатрів у сучасних умовах розвитку цифрових технологій та зростаючої популярності стрімінгових сервісів. Під час виконання роботи було проаналізовано існуючі аналоги онлайн-кінотеатрів, їхні функціональні можливості, переваги та недоліки.

Також досліджено сучасний стан інструментів, моделей, методів та архітектур розробки вебзастосунків. Було розглянуто можливості використання React, TypeScript, Node.js, PostgreSQL та REST API для створення масштабованого та високопродуктивного програмного забезпечення. Було визначено доцільність використання архітектури «клієнт-сервер» та підходу SPA під час реалізації вебзастосунку для онлайн-кінотеатру.

Крім того, у роботі було визначено основні функціональні та нефункціональні вимоги до програмного забезпечення. До основних функцій системи належать аутентифікація користувачів, перегляд та пошук відеоконтенту, формування списку обраного, а також адміністративне управління каталогом фільмів. Також сформовано вимоги до продуктивності, безпеки, надійності та масштабованості системи.

Отримані результати дослідження та аналізу предметної області, сучасних технологій і підходів до розробки вебзастосунків дозволили обґрунтувати вибір архітектурних рішень і інструментів реалізації. Це створює основу для подальшого розвитку та вдосконалення вебзастосунку онлайн-кінотеатру як зручного та функціонального сервісу для користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) React Documentation. URL: https://react.dev/ (дата звернення: 03.05.2026).

2) PostgreSQL Documentation. URL: https://www.postgresql.org/docs/ (дата звернення: 03.05.2026).

3) ISO/IEC/IEEE 29148:2021. Systems and software engineering – Life cycle processes – Requirements engineering. Geneva : ISO, 2021. 100 p.

4) ISO/IEC 25010:2023. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. Geneva : ISO, 2023. 34 p.

5) PostgreSQL Documentation. Version 17. https://www.postgresql.org/docs/17/ (Accessed: 03.05.2026).

6) Цибульник С. О., Барандич К. С. Технології розроблення програмного забезпечення. Частина 1. Життєвий цикл програмного забезпечення : підручник. Київ : КПІ ім. Ігоря Сікорського, 2022. 270 с.

7) Sarthak Verma: Express.Js and its Usage in Web Development.2024. International Journal of Research Publication and Reviews. Vol. 5 P. 2582-7421. DOI: https://www.ijrpr.com/archive.php?volume=5&issue=4 (Accessed: 03.05.2026).

8) WEZOM. REST API: принципи роботи вебсервісів. URL: https://wezom.com.ua/ua/blog/rest-api-printsipi-roboti-veb-servisiv (дата звернення: 03.05.2026).

9) JetBrains. JavaScript and TypeScript Trends in 2024. URL: https://blog.jetbrains.com/webstorm/2024/02/js-and-ts-trends-2024/ (Accessed: 03.05.2026).

10) Системна архітектура "клієнт-сервер" URL: https://studfile.net/preview/9250282/ (дата звернення: 03.05.2026).

11) Git Documentation. URL: https://git-scm.com/doc (дата звернення: 03.05.2026).

12) Docker Documentation. URL: https://docs.docker.com/ (дата звернення: 03.05.2026).

13) JWT Introduction. URL: https://jwt.io/introduction (дата звернення: 03.05.2026).

14) Node.js Documentation. URL: https://nodejs.org/docs/latest/api/ (дата звернення: 03.05.2026).

15) Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. O'Reilly Media, 2021. 617 p.

16) Mozilla Developer Network. MVC Architecture. URL: https://developer.mozilla.org/ (дата звернення: 03.05.2026).

ДОДАТОК А

Програмний код адміністративної панелі

Лістинг А.1 – Адміністративна панель (AdminPanel.tsx):

```
import { useEffect, useState } from "react";

interface Movie {
  id: number;
  title: string;
  genre: string;
  year: number;
  rating: number;
  image: string;
  description: string;
}

function AdminPanel() {
  const [movies, setMovies] = useState<Movie[]>([]);

  const [title, setTitle] = useState("");
  const [genre, setGenre] = useState("");
  const [year, setYear] = useState("");
  const [rating, setRating] = useState("");
  const [image, setImage] = useState("");
  const [description, setDescription] = useState("");

  const [editingId, setEditingId] = useState<number | null>(null);

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

  const loadMovies = async () => {
    const res = await fetch("http://localhost:5000/api/movies");
    const data = await res.json();
    setMovies(data);
  };

  const saveMovie = async () => {
    if (!title || !genre || !year || !rating || !image) {
      alert("Заповніть обов'язкові поля");
      return;
    }

    const payload = {
      title,
      genre,
      year: Number(year),
      rating: Number(rating),
      image,
      description,
    };

    if (editingId !== null) {
      await fetch(`http://localhost:5000/api/movies/${editingId}`, {
        method: "PUT",
        headers: { "Content-Type": "application/json" },
```

```
        body: JSON.stringify(payload),
      });
    } else {
      await fetch("http://localhost:5000/api/movies", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(payload),
      });
    }

    clearForm();
    setEditingId(null);
    loadMovies();
  };

  const deleteMovie = async (id: number) => {
    await fetch(`http://localhost:5000/api/movies/${id}`, {
      method: "DELETE",
    });

    loadMovies();
  };

  const editMovie = (movie: Movie) => {
    setTitle(movie.title);
    setGenre(movie.genre);
    setYear(movie.year.toString());
    setRating(movie.rating.toString());
    setImage(movie.image);
    setDescription(movie.description);
    setEditingId(movie.id);
  };

  const clearForm = () => {
    setTitle("");
    setGenre("");
    setYear("");
    setRating("");
    setImage("");
    setDescription("");
  };

  return null;
}

export default AdminPanel;
```

ДОДАТОК Б

Програмний код контексту автентифікації користувачів

Лістинг Б.1 – Контекст автентифікації (AuthContext.tsx):

```
import { createContext, useContext, useEffect, useState } from "react";

type User = {
  name: string;
  email?: string;
  role?: string;
};

type AuthContextType = {
  user: User | null;
  login: (user: User) => void;
  logout: () => void;
};

const AuthContext = createContext<AuthContextType | null>(null);

export function AuthProvider({ children }: any) {
  const [user, setUser] = useState<User | null>(null);

  useEffect(() => {
    const savedUser = localStorage.getItem("user");
    if (savedUser) {
      setUser(JSON.parse(savedUser));
    }
  }, []);

  const login = (userData: User) => {
    setUser(userData);
    localStorage.setItem("user", JSON.stringify(userData));
  };

  const logout = () => {
    setUser(null);
    localStorage.removeItem("user");
  };

  return (
    <AuthContext.Provider value={{ user, login, logout }}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  const context = useContext(AuthContext);
  if (!context) throw new Error("useAuth must be used inside AuthProvider");
  return context;
}
```

ДОДАТОК В

Структура бази даних онлайн-кінотеатру

Лістинг В.1 – SQL-скрипт створення бази даних онлайн-кінотеатру (cinema.sql):

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(100) NOT NULL,
    email VARCHAR(150) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    role VARCHAR(20) DEFAULT 'user'
);

CREATE TABLE movies (
    id SERIAL PRIMARY KEY,
    title VARCHAR(255),
    genre VARCHAR(100),
    year INTEGER,
    rating NUMERIC(2,1),
    image TEXT,
    description TEXT
);

CREATE TABLE favorites (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(id) ON DELETE CASCADE,
    movie_id INTEGER NOT NULL
);

CREATE TABLE reviews (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(id) ON DELETE CASCADE,
    movie_id INTEGER NOT NULL,
    comment TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```