МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»
_____,

освітньо-професійної програми «Інформаційні технології проектування»

на тему: Вебдодаток для спільного перегляду фільмів та спілкування в реальному часі

Здобувача (ки) групи __ІТ-02_ ___Оберемко Павла Юрійовича_____
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ ____Павло ОБЕРЕМКО_____
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник доцент кафедри ІТ, к.т.н., доцент Володимир НАГОРНИЙ _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В. о. зав. кафедри ІТ

_____Світлана ВАЩЕНКО
« » 2024 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Оберемко Павлу Юрійовичу

1 Тема роботи *Вебдодаток для спільного перегляду фільмів та спілкування в реальному часі*

керівник роботи *Нагорний Володимир В'ячеславович, к.т.н., доцент*

затверджені наказом по університету від « 07 » травня 2024 р. №0482-VI

2 Строк подання студентом роботи « 26 » травня 2024 р.

3 Вхідні дані до роботи технічне завдання на розробку вебдодатку_____

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, моделювання вебдодатку спільного перегляду, розробка вебдодатку спільного перегляду

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) визначення актуальності розробки вебдодатку, порівняння продуктів-аналогів, функціональні вимоги до вебдодатку, контекстна діаграма IDEF0, діаграма декомпозиції першого рівня, діаграма варіантів використання, архітектура вебдодатку, програмна реалізація вебдодатку, використання вебдодатку, висновки, презентація роботи веб додатку

6. Консультанти розділів роботи:

Розділ	**Консультант**	**Підпис, дата**	
		Завдання видав	**Завдання прийняв**

7.Дата видачі завдання 04.04.2024_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження предметної області	08.04.2024-15.04.2024	Виконано
2	Оформлення планування робіт	17.04.2024-22.04.2024	Виконано
3	Оформлення технічного завдання	23.04.2024-26.04.2024	Виконано
4	Аналіз та вибір засобів реалізації	27.04.2024-01.05.2024	Виконано
5	Проектування вебдодатку	02.05.2024-04.05.2024	Виконано
6	Розробка вебдодатку	03.05.2024-15.05.2024	Виконано
7	Тестування вебдодатку	16.05.2024-17.05.2024	Виконано
8	Оформлення пояснювальної записки	18.05.2024-20.05.2024	Виконано

Студент _____ Павло ОБЕРЕМКО

(підпис)

Керівник роботи _____ к.т.н., доц.

(підпис) Володимир НАГОРНИЙ

АНОТАЦІЯ

Тема кваліфікаційної роботи бакалавра «Вебдодаток для спільного перегляду фільмів та спілкування в реальному часі».

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел, додатків. Загальний обсяг роботи − 96 сторінок, у тому числі 33 сторінки основного тексту, 3 сторінки списку використаних джерел, 53 сторінок додатків.

Актуальність роботи полягає у впливі війни в Україні, підвищений інтерес до онлайн-розваг, вплив пандемії COVID-19, зростаючій потребі в соціальних взаємодіях, зростанні популярності стрімінгових сервісів.

Мета роботи: створення вебдодатку для спільного перегляду фільмів та спілкування в реальному часі. Основним завданням є створення інтерактивного середовища, яке об'єднає користувачів навколо спільних інтересів у фільмах та відкритого обміну враженнями.

У першому розділі проведено аналіз існуючих аналогів та програмних продуктів для реалізації вебдодатків онлайн-кінотеатрів. У другому розділі проведено структурно-функціональне моделювання вебдодатку. У третьому розділі наведена практична реалізація компонентів вебдодатку.

Ключові слова: спільний перегляд, React, TMDB API, Firebase, вебдодаток.

ЗМІСТ

ВСТУП

Сьогодні послуги онлайн-кінотеатрів стають все більш популярними, адже вони пропонують зручний та цікавий спосіб перегляду фільмів та взаємодії в онлайн-середовищі [1]. Інноваційні технології не тільки забезпечують комфорт під час перегляду, але й дають можливість спільно дивитися фільми з друзями та знайомими [2, 3].

Зростання вимог до комфорту та доступності сервісів стимулює появу нових підходів до надання послуг [4-5]. У сфері розваг, зокрема в кінематографії, з'являється потреба в змішаному досвіді, який поєднує онлайн-технології та соціальне спілкування [6-8].

Метою даного проєкту є створення вебдодатку для спільного перегляду фільмів та спілкування в реальному часі.

Задачами кваліфікаційної роботи бакалавра є дослідження актуальності обраної теми та проведення аналізу предметної області, аналіз існуючих аналогічних додатків, складання технічного завдання для створення вебдодатку, структурно-функціональне моделювання, проектування вебдодатку, розробка архітектури вебдодатку та програмна реалізація.

Розроблений вебдодаток може бути використаний для спільного перегляду фільмів з друзями та знайомими, а також для онлайн-кінотеатрів, які прагнуть розширити спектр своїх послуг.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Дослідження останніх років показує швидку еволюцію споживання контенту з використанням стрімінгових сервісів та інших платформ для перегляду фільмів. Спочатку споживачі в основному вибирали фізичні кінотеатри для перегляду фільмів. Однак з появою і популяризацією онлайн-платформ, таких як Netflix, Amazon Prime Video, Disney+, Hulu та інші, тенденція змінилася [9-14].

За даними аналізу, тенденція переходу до споживання контенту через Інтернет активно зростає. Це викликано кількома факторами. По-перше, розвиток технологій, які дозволяють швидко та зручно отримувати доступ до великого обсягу контенту за допомогою мобільних пристроїв, комп'ютерів і сучасних телевізорів. По-друге, споживачі цінують зручність та гнучкість, яку надають вебдодатки, оскільки вони можуть переглядати фільми у будь-який зручний для них час та місце.

Ця зміна має великі наслідки для індустрії кіно. Фізичні кінотеатри стикаються з викликом привернення глядачів, оскільки деякі люди віддають перевагу домашньому перегляду через зручність та економію часу. Проте, це також відкриває нові можливості для кінематографічних студій і творців контенту, які можуть прямо залучати аудиторію через вебдодатки, обходячи традиційні канали дистрибуції.

Щодо перспектив та прогнозів, здається, що вебдодатки для перегляду фільмів і стрімінгові сервіси продовжать займати центральне місце у споживчій поведінці. Проте, конкуренція між ними може посилитися, що призведе до подальшого розвитку функціоналу та збільшення якості контенту. Крім того, можна очікувати зростання інтересу до інтерактивних форматів

перегляду, таких як віртуальна реальність та доповнена реальність, що може змінити спосіб, яким ми сприймаємо фільми та інші види розважального контенту.

У цілому, напрямок розвитку онлайн-перегляду фільмів та спільного перегляду в реальному часі є надзвичайно актуальним, і прогнозується його подальший розвиток та зріст. Тому розробка вебдодатків для спільного перегляду фільмів та спілкування в реальному часі може мати значний потенціал успіху.

1.2 Аналіз існуючих продуктів аналогів

1.2.1 Netflix

Netflix(рис. 1.1) відомий своєю широкою бібліотекою відмінного контенту, включаючи оригінальні фільми та серіали, такі як "Stranger Things" і "The Crown". Його алгоритм рекомендацій відзначається високою точністю та особистою підбіркою контенту для кожного користувача. Однак, зі зростанням конкуренції, Netflix повинен постійно вдосконалювати свій оригінальний контент та збільшувати привабливість для користувачів.

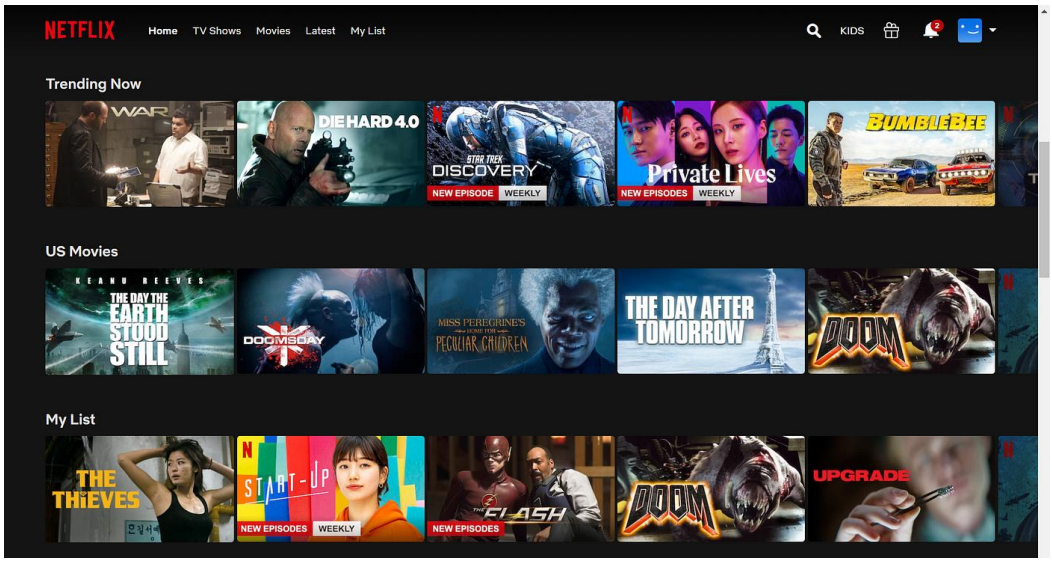


Рисунок 1.1 – «Netflix»

Переваги:

- Зручний інтерфейс та можливість перегляду на будь-якому пристрої.
- Персоналізовані рекомендації контенту.

Недоліки:

- Не підтримує спільний перегляд контенту та спілкування в реальному часі.
- Необхідність підписки.
- Деякі фільми та серіали доступні лише в певних регіонах.

Можливі доповнення:

- Сумісний перегляд контенту для кількох користувачів одночасно.
- Можливість спілкування між користувачами під час перегляду.

1.2.2 Apple TV

Apple TV (рис. 1.2) - це стрімінговий сервіс від компанії Apple, який пропонує широкий вибір високоякісного відеоконтенту, включаючи фільми, серіали, оригінальний контент, трансляції спортивних подій та багато іншого. Цей сервіс відзначається своєю інтеграцією з екосистемою Apple, що означає, що ви можете легко отримати доступ до нього на будь-якому пристрої Apple, такому як iPhone, iPad, Apple TV Box, Mac, а також деяких смарт-телевізорах.

Підвищення персоналізації рекомендацій є ще однією важливою аспектом для Apple TV. Шляхом аналізу даних про користувачів та їхніх переглядів, Apple TV намагається надати користувачам індивідуально підібраний контент, який відповідає їхнім інтересам та вподобанням.

Рисунок 1.2 – «Apple TV»

Переваги:

- Інтеграція з екосистемою Apple.
- Можливість перегляду на різних пристроях.

Недоліки:

- Залежність від пристроїв Apple обмежує доступність для користувачів інших платформ.
- Висока вартість підписки. Не підтримує спільний перегляд контенту та спілкування в реальному часі.

Можливі доповнення:

- Сумісний перегляд контенту для кількох користувачів одночасно.
- Можливість спілкування між користувачами під час перегляду.

1.2.3 Amazon Prime Video

Amazon Prime Video (рис. 1.3) відзначається широким вибором контенту, включаючи оригінальні серіали, які стали хітом. Його алгоритми рекомендацій постійно вдосконалюються, щоб забезпечити користувачам персоналізований контент. Однак, зростаюча конкуренція може вимагати від Amazon Prime Video постійних інновацій та підвищення якості користувацького досвіду.

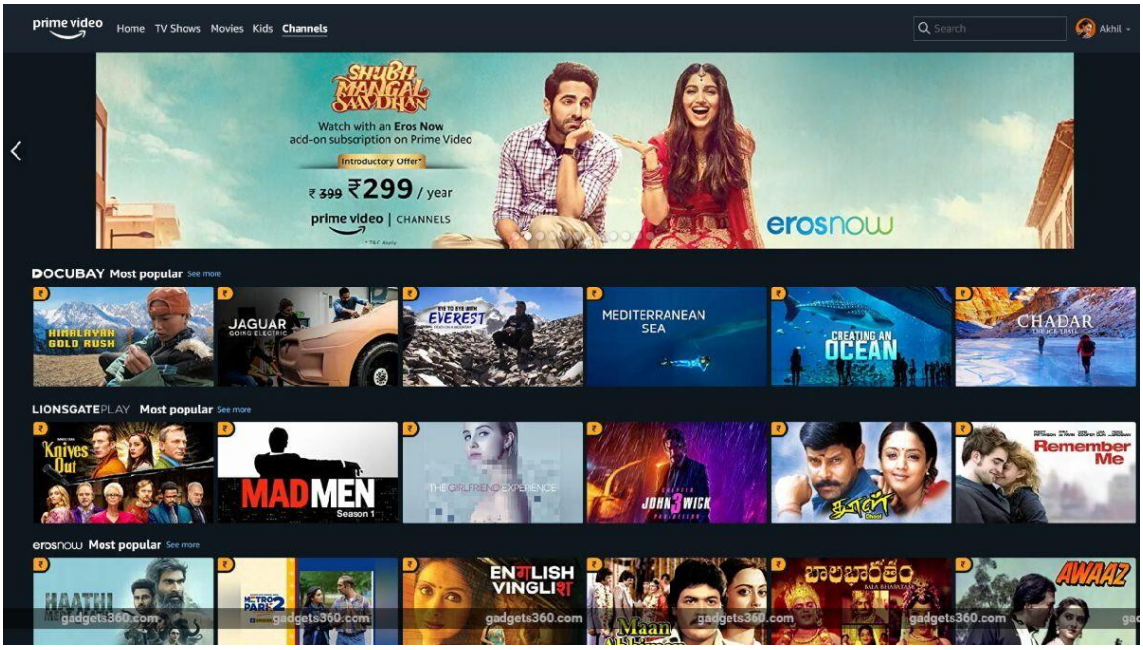


Рисунок 1.3 – «Amazon Prime Video»

Переваги:

- Не підтримує спільний перегляд контенту та спілкування в реальному часі.

Недоліки:

- Складний для навігації інтерфейс може негативно впливати на користувацький досвід.

Можливі доповнення:

- Додати можливість спільного перегляду контенту та спілкування в реальному часі.
- Спростити інтерфейс для більш зручної навігації.

В таблиці характеристик аналогів вебдодатків (табл. 1.1) представлено порівняльний аналіз трьох популярних вебдодатків для спільного перегляду фільмів та серіалів: Netflix, Apple TV та Amazon Prime Video. Оцінювання проводиться за кількома ключовими характеристиками, які є важливими для користувачів, які шукають платформу для спільного перегляду контенту.

Таблиця 1.1 – Порівняльна таблиця характеристик аналогів вебдодатків

Характеристика/вебдодаток	«Netflix»	«Apple TV»	«Amazon Prime Video»	Власна розробка
Широкий вибір контенту (кількість фільмів, серіалів)	+	-	+	+
Підтримка різних пристроїв (мобільні, ПК)	+	+	+	+
Сумісний перегляд контенту	-	-	-	+
Чат для спілкування під час перегляду	-	-	-	+
Можливість спільного створення списків фільмів/серіалів	-	-	-	+

На основі порівняльного аналізу існуючих продуктів-аналогів, а також детального опису технічного завдання (додаток А), визначено наступні функціональні вимоги до майбутньої розробки вебдодатку:

1. Базові функції:
    a. Широкий вибір контенту з акцентом на спільний перегляд.
    b. Підтримка різних пристроїв та адаптивний інтерфейс.
    c. Сумісний перегляд контенту, спілкування в чаті.
2. Додаткові функції:
    a. Інтерактивні елементи (завдання).
    b. Функції спілкування.
3. Технічні вимоги:
    a. Висока якість зображення та звуку (HD, Full HD, 4K, об'ємний звук).
    b. Швидке завантаження контенту без затримок.
    c. Цілодобова технічна підтримка.

1.3 Мета та задачі дослідження

Метою даного проєкту є створення вебдодатку для спільного перегляду фільмів та спілкування в реальному часі. Основним завданням є створення інтерактивного середовища, яке об'єднає користувачів навколо спільних інтересів у фільмах та відкритого обміну враженнями.

Для досягнення цієї мети необхідно виконати наступні задачі:

- Аналіз сучасних тенденцій у веб-розвитку та користувацьких потреб. Передбачається дослідження останніх технологічних та інноваційних тенденцій у веб-розвитку, зокрема у сфері онлайн-кінотеатрів. Очікується виявлення потенційних можливостей для реалізації нових ідей та функціональності вебдодатку для спільного перегляду фільмів та спілкування в реальному часі.

- Проаналізувати аналоги, виділивши їх переваги та недоліки. Передбачається дослідження існуючих вебдодатків, їхнього функціоналу та інтерфейсів[15-17]. Очікується ідентифікація переваг та недоліків конкуруючих додатків, щоб зрозуміти, які можливості можна покращити в новому вебдодатку.
- Сформулювати функціональні вимоги. Передбачається визначення основних функцій та можливостей, які має мати вебдодаток, таких як пошук фільмів, спільний перегляд, чат під час перегляду тощо. Очікується визначення пріоритетів та обсягу реалізації кожної функції для ефективного планування розробки.
- Виконати моделювання та проектування. Передбачається створення детальних схем інтерфейсів користувача а також структурно-функціональне моделювання та IDEF0. Очікується розроблені макети інтерфейсів, які відображатимуть сторінки пошуку фільмів, перегляду фільмів та інші необхідні елементи.
- Розробити back-end та front-end частину. Передбачається реалізація back-end частини для обробки запитів користувачів для реалізації спільного перегляду та спілкування. Очікується розроблення front-end частини для створення зручного та інтуїтивно зрозумілого інтерфейсу користувача.
- Виконати тестування. Передбачається проведення тестування вебдодатку для перевірки правильності реалізації функціональності та виявлення помилок. Очікується виявлення та усунення будь-яких дефектів для забезпечення якості та надійності роботи вебдодатку.

Реалізація вебдодатку буде проведена за допомогою технології React.js, яка забезпечить високий рівень інтерактивності та швидкодії[18]. Для отримання інформації про фільми та їх трейлери буде використано TMDB API.

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

2.1 Структурно-функціональне моделювання

Розробка програмного забезпечення потребує ретельного планування, і тут ключову роль відіграє структурно-функціональне моделювання[19]. Завдяки цьому підходу можна глибоко дослідити систему, оптимізувати її роботу, економлячи час та ресурси.

Для візуалізації функціонального моделювання використовується методологія IDEF0. Її мета – чітко описувати взаємопов'язані системи та процеси. Вона складається з блоків (що представляють процеси та функції) та стрілок (які позначають вхідні, вихідні дані, контролюючі фактори та механізми).

Рисунок 2.1 демонструє контекстну діаграму, що відображає основні процеси вебдодатку для спільного перегляду фільмів та спілкування в реальному часі.

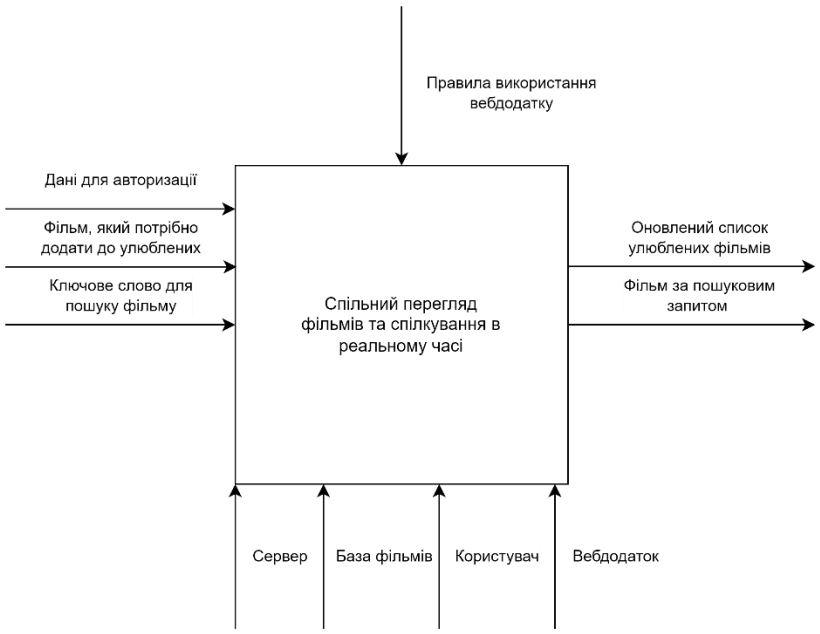


Рисунок 2.1 – Контекстна діаграма IDEF0

Як видно з рисунку 2.1, визначено наступні елементи діаграми IDEF0:

- Вхідні дані: запит на авторизацію, додавання фільму до улюблених, пошук фільму, запит на створення кімнати для спільного перегляду, надсилання повідомлення.
- Керування: інструкція по використанню вебдодатку.
- Механізми: база фільмів, користувачі, вебдодаток.
- Вихідні дані: авторизований користувач, оновлений список улюблених фільмів, список фільмів за запитом, створена кімната для спільного перегляду, чат з автоматично оновленими повідомленнями.

Для детального опису процесів та функціоналу складних систем використовується потужний метод декомпозиції функціонального проектування. Цей метод ґрунтується на принципі розбиття складних функцій на менші, чітко визначені та зрозуміліші елементи.

Декомпозиція може мати декілька рівнів, що дає змогу детально описати всі аспекти роботи системи. Наприклад, на першому рівні декомпозиції система може бути розбита на основні підсистеми. На другому рівні кожна підсистема може бути розбита на менші модулі, а на третьому рівні - на ще менші компоненти.

Для візуалізації декомпозиції функціонального проектування використовуються різні діаграми, такі як діаграми IDEF0, діаграми структурних потоків даних (DFD) та діаграми потоків робіт (workflow diagrams). Ці діаграми допомагають чітко представити структуру системи, зв'язки між її компонентами та потоки даних.

Рисунок 2.2 демонструє декомпозиційну діаграму функціональної моделі вебдодатку.

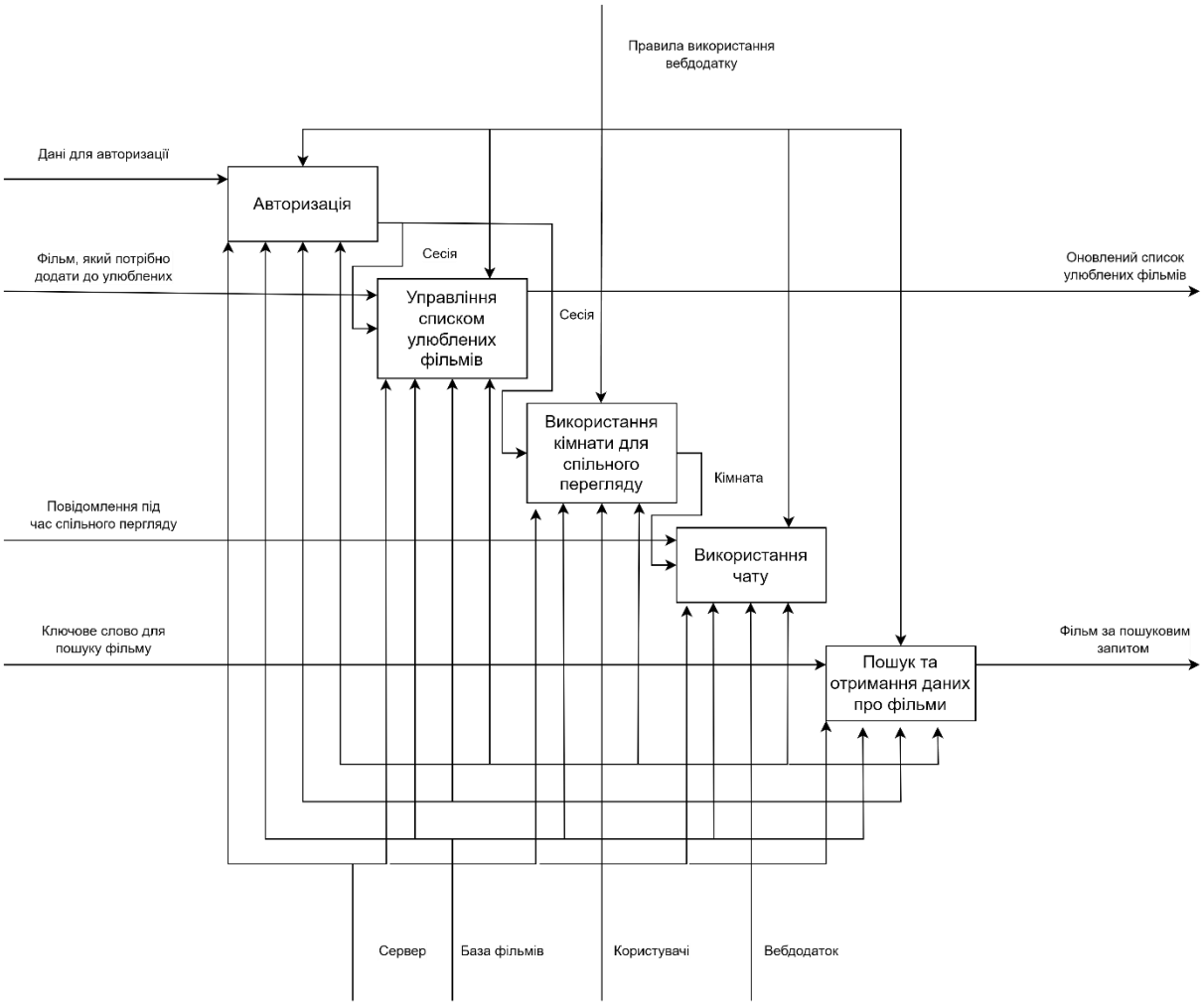


Рисунок 2.2 – Діаграма декомпозиції першого рівня

Ця структурно-функціональна модель допомагає чітко візуалізувати основні компоненти та їх взаємодію у вебдодатку для спільного перегляду фільмів та спілкування в реальному часі. Використання діаграм дозволяє систематизувати процеси, визначити ключові функції та їх взаємозв'язок, що є важливим етапом в розробці ефективного та надійного програмного забезпечення.

Діаграма декомпозиції першого рівня (DFD) для вебдодатку, який включає функції авторизації, управління списком улюблених фільмів, використання кімнати для спільного перегляду, використання чату та пошук і отримання даних про фільми.

Основні компоненти діаграми:

- Авторизація:
  Вхідні дані: Дані для авторизації
  Вихідні дані: Правила використання вебдодатку
- Управління списком улюблених фільмів:
  Вхідні дані: Фільм, який потрібно додати до улюблених
  Вихідні дані: Оновлений список улюблених фільмів
- Використання кімнати для спільного перегляду:
  Вхідні дані: Сесія
  Вихідні дані: Повідомлення під час спільного перегляду
- Використання чату:
  Вхідні дані: Сесія
  Вихідні дані: Повідомлення під час спільного перегляду
- Пошук та отримання даних про фільми:
  Вхідні дані: Ключове слово для пошуку фільму
  Вихідні дані: Фільм за пошуковим запитом

Користувачі взаємодіють з цими компонентами через вебдодаток, який підключений до сервера та бази фільмів для отримання та обробки необхідної інформації.

2.2 Проектування вебдодатку

Діаграма варіантів використання (рис. 2.2)[20] відіграє ключову роль у проектуванні інформаційних систем. Вона візуалізує взаємозв'язки між акторами (користувачами) та системою, чітко відображаючи функціональні вимоги зі сторони користувача.

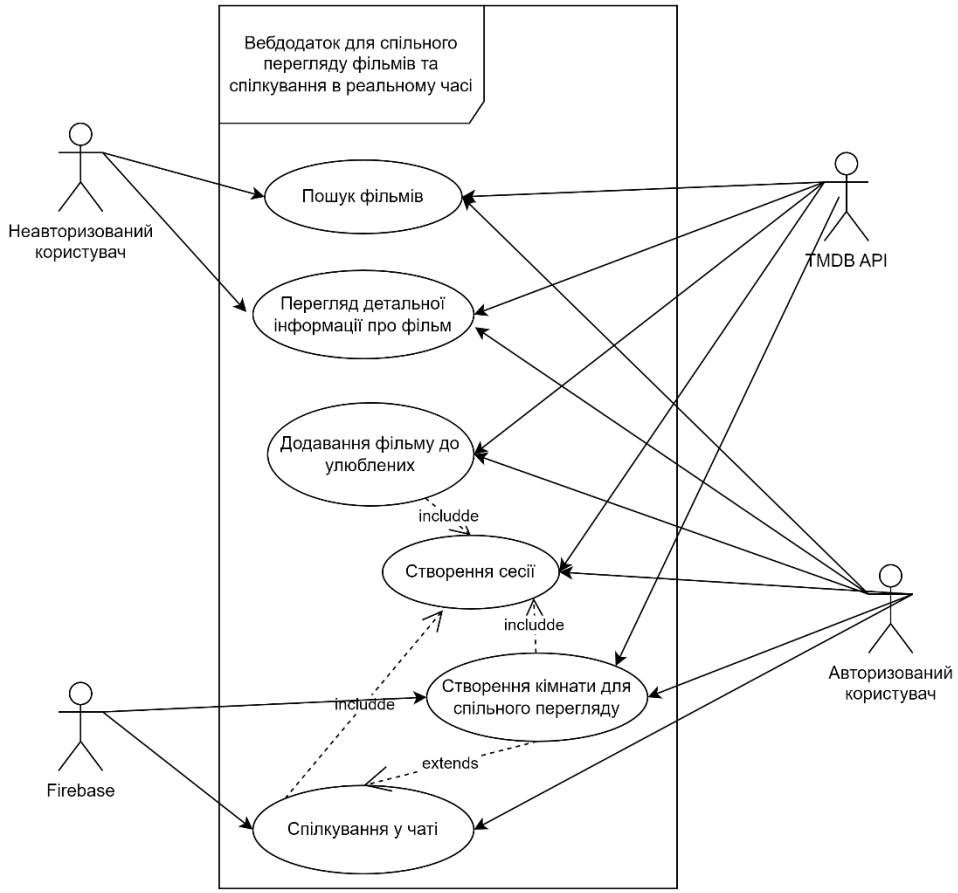


Рисунок 2.2 – Діаграма варіантів використання

Як видно з рисунка, для вебдодатку спільного перегляду фільмів та спілкування в реальному часі було визначено наступні типи користувачів:

- Користувач

Варіанти використання наведено нижче:

- Створення сесії
- Пошук фільмів
- Перегляд детальної інформації про фільм
- Додавання фільму до улюблених
- Створення кімнати для спільного перегляду
- Спілкування у чаті

2.3 Проектування бази даних

Оскільки в проєкті використовується TMDB API яка вже надає готову базу фільмів, функціонал авторизації та додавання фільмів до улюблених, то було прийнято рішення на створювати базу даних для фільмів.

Для реалізації спільного перегляду фільмів та спілкування в реальному часі, було обрано Firebase. Firebase є ідеальним вибором для реалізації проєкту спільного перегляду фільмів з кількох причин:

- Реал-тайм оновлення: Firebase Database забезпечує автоматичне оновлення даних у реальному часі. Це особливо важливо для чату та синхронізованого перегляду фільмів, де всі користувачі повинні отримувати оновлення миттєво.
- Легка інтеграція: Firebase має простий у використанні SDK для JavaScript, що дозволяє легко інтегрувати його з React.js проєктом. Це знижує складність розробки та прискорює процес реалізації.
- Масштабованість: Firebase може обробляти велику кількість запитів і підключень одночасно, що дозволяє вебдодатку залишатися продуктивним навіть при великій кількості користувачів.
- Хмарне зберігання: Firebase пропонує не лише базу даних, але й хмарне зберігання файлів (Firebase Storage).

Сутності та зв'язки в Firebase Database:

Користувачі (Users):

- userId: Унікальний ідентифікатор користувача.
- name: Ім'я користувача.
- roomId: Ідентифікатор кімнати, до якої належить повідомлення.
- text: Текст повідомлення.
- timestamp: Час відправлення повідомлення.

Використання Firebase забезпечує простоту інтеграції, масштабованість та реальну часову синхронізацію, що є критичними для досягнення якісного користувацького досвіду. Описані сутності та їхні зв'язки дозволяють ефективно організувати дані та забезпечити необхідну функціональність вебдодатку.

2.4 Архітектура вебдодатку

Вебдодаток призначений для того, щоб користувачі могли переглядати фільми разом в реальному часі та спілкуватися через чат. Система використовує React.js для фронтенду, TMDB API для отримання даних про фільми, та Firebase для реалізації функцій чату та спільного перегляду.

Клієнтська частина (React.js):

- Компоненти користувацького інтерфейсу: компоненти для відображення списку фільмів, детальної інформації про фільми, відеоплеєра, чату, тощо.
- Стан додатку: управління станом за допомогою Redux для глобального стану.
- Роутинг: використання React Router для навігації між сторінками (наприклад, головна сторінка, сторінка фільму, сторінка спільного перегляду).

Серверна частина:

- TMDB API: взаємодія з API для отримання даних про фільми.
- Firebase:
    - Authentication: аутентифікація користувачів через Firebase Authentication.
    - Realtime Database: для збереження даних чату та інформації про спільний перегляд фільмів.

Схема архітектури вебдодатку для спільного перегляду фільмів та спілкування в реальному часі представлена на рисунку 2.3.

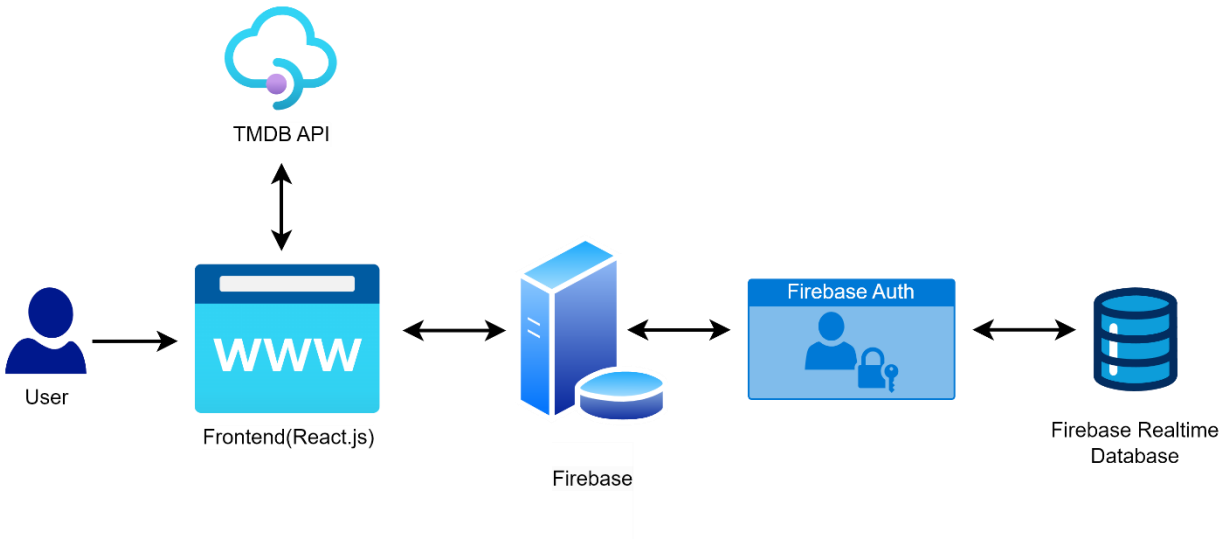


Рисунок 2.3 - Архітектура вебдодатку

Запропонована архітектура вебдодатку забезпечує інтеграцію між сучасними фронтенд і бекенд технологіями для створення платформи спільного перегляду фільмів та спілкування у реальному часі. Використання React.js дозволяє створити динамічний і зручний інтерфейс користувача, тоді як Firebase забезпечує надійний і масштабований бекенд для аутентифікації, зберігання даних і реального часу комунікації. Інтеграція з TMDB API дозволяє легко отримувати актуальну інформацію про фільми, що додає цінності користувачам.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Програмна реалізація

У цьому розділі детально розглядаються кроки, необхідні для створення вебдодатку для спільного перегляду фільмів та спілкування в реальному часі. Описуються основні етапи налаштування середовища розробки, реалізація ключових компонентів додатку та інтеграція з необхідними зовнішніми сервісами, такими як Firebase та TMDB API. Метою є забезпечення надійної, безпечної та масштабованої платформи, яка надає користувачам можливість синхронізованого перегляду відео та чату.

Перший крок – це ініціалізація та налаштування проєкту з усіма залежностями та бібліотеками, які необхідні для розробки.

Список усіх залежностей:

- Fontawesome – відповідає за всі іконки вебдодатку.
- MaterialUI – використовуються готові матеріали базових компонентів, що значно прискорює розробку.
- Axios – використовується для взаємодією з запитами до API.
- React Player – використовується для інтеграції плеєра.
- React Redux – використовується для керування станом програми.
- React Router DOM – використовується для навігації між сторінками.
- Sass – використовується для швидкого написання стилів.
- Swiper – зручна бібліотека слайдерів.

Для більш швидшої та ефективної розробки, використовується інструмент для компіляції – Vite.

Для створення та ініцілізації проєкту було використано команду «npm create vite@latest».

Наступним кроком було розроблено логіку коректної взаємодії з TMDB API. Цей код був винесений в окремий файл, завдяки чому його дуже зручно використовувати на різних етапах розробки. Також детально продумано різні сценарії обробки помилок.

```
const useFetch = (url, lang) => {

    if(url == null){
        console.log('no url');
    }

    const [data, setData] = useState(null);
    const [loading, setLoading] = useState(null);
    const [error, setError] = useState(null);

    useEffect(() => {
        setLoading("loading...");
        setData(null);
        setError(null);

        fetchDataFromApi(url, {}, lang)
            .then((res) => {
                setLoading(false);
                setData(res);
            })
            .catch((err) => {
                setLoading(false);
                setError("Something went wrong!");
            });
    }, [url]);

    return { data, loading, error };
};
```

Такий підхід дозволяє відслідковувати будь-яку відповідь сервісу.

Axios запит, що дозволяє підключитись до TMDB API:

```
import axios from "axios";

const BASE_URL = "https://api.themoviedb.org/3";
const TMDB_TOKEN = import.meta.env.VITE_APP_TMDB_TOKEN;

const headers = {
    accept: 'application/json',
    Authorization: "bearer " + TMDB_TOKEN,
};

export const fetchDataFromApi = async (url, params, lang = 'uk') => {
    const OPTIONS_URL = `?language=${lang}&region=ua`;
```

```
    try {
        const { data } = await axios.get(BASE_URL + url + OPTIONS_URL, {
            headers,
            params,
        });
        return data;
    } catch (err) {
        console.log(err);
        return err;
    }
};
```

```
const { data: popularData, loading: loadingData } =
useFetch("/trending/movie/day");
const [popularMovies, setPopularMovies] = useState([]);

useEffect(() => {
   popularData && setPopularMovies(() => popularData.results);
}, [popularData]);
```

Результат запиту найпопулярніших фільмів зображено на рисунку 3.1.

Рисунок 3.1 – Результат запиту найпопулярніших фільмів

Наступник кроком було налаштовано переміщення між сторінками. Оскільки React.js ідеально підходить для розробки Single Page Application, переміщення було налаштовано таким чином, що сторінка не перезавантажується при переході на інший пункт меню.

```
<Routes>
    <Route element={<Home/>} path='/'/>
    <Route element={<SingleMedia/>} path='/media/:type/:id'/>
    <Route element={<SingelCrew/>} path='/crew/:id'/>
    <Route element={<MediaListPage type='movie'/>} path='/films'/>
    <Route element={<MediaListPage type='tv'/>} path='/series'/>

    <Route element={<h2>404</h2>} path='*'/>
</Routes>
```

Після даного кроку було створено шаблони сторінок, та, за допомогою створеного раніше шаблону для запитів до TMDB API, виведено всю необхідну інформацію на сторінках. Приклад розробленої сторінки зображено на рисунку 3.2 та рисунку 3.3.

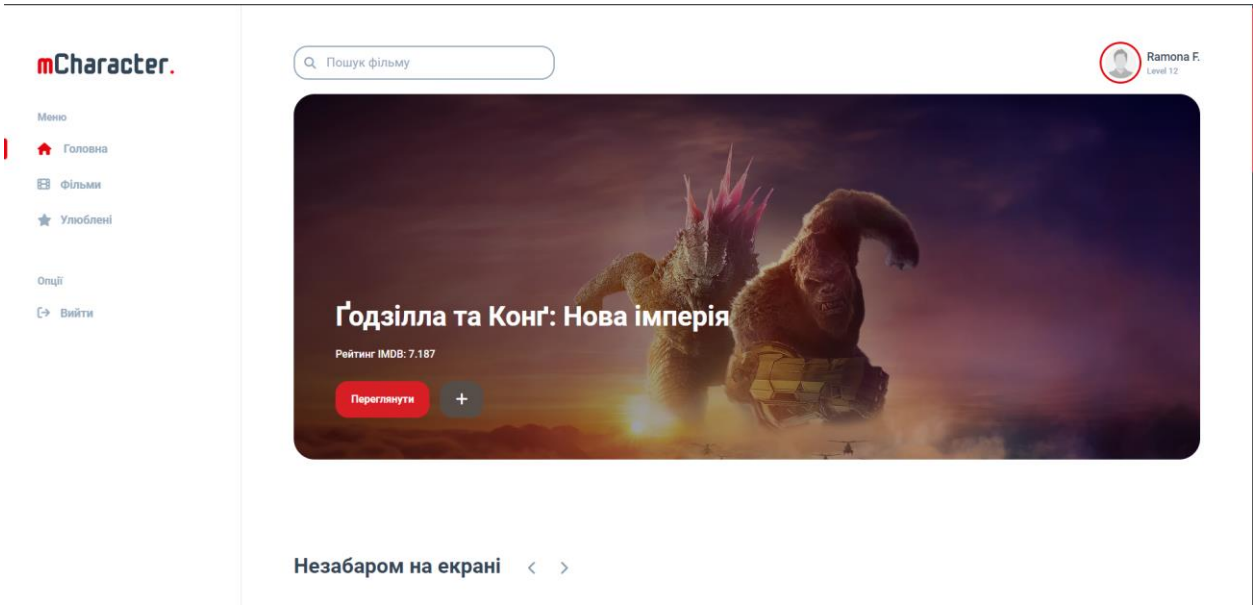


Рисунок 3.2 – Головна сторінка

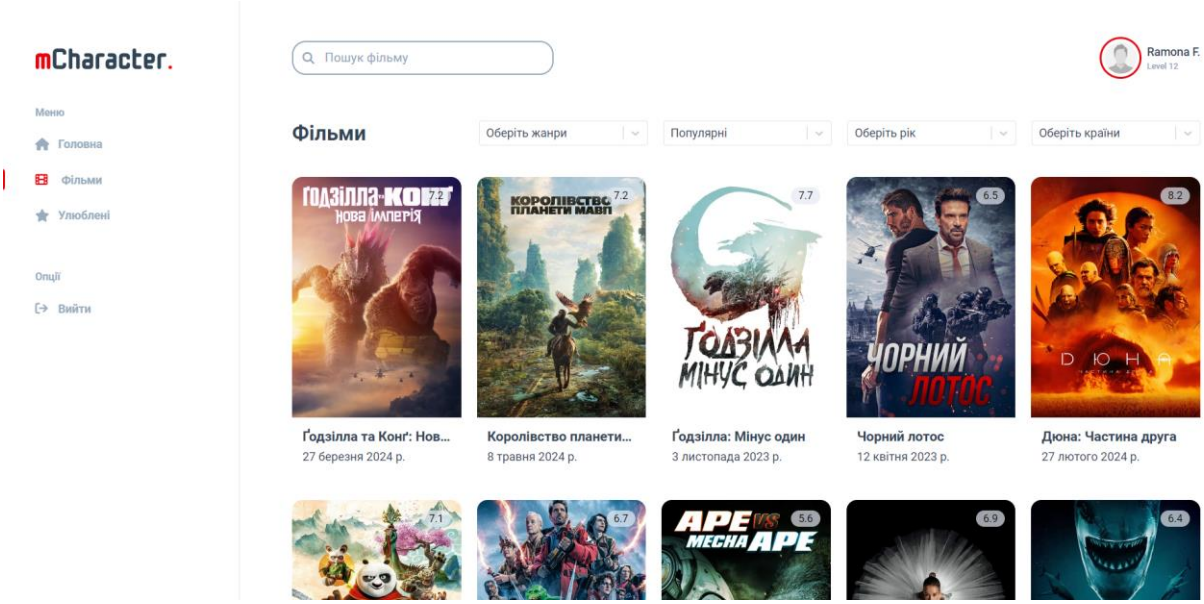


Рисунок 3.3 – Сторінка усіх фільмів

Підключення до сервісу Firebase (рис. 3.4) надало можливість реалізувати спільний перегляд та спілкування в чаті одночасно (рис. 3.5). Для цього потрібно було зареєструватись на сервісі, створити та налаштувати сервер для обміну даними у реальному часі. Після чого сервісом було надано код підключення, за допомогою якого відбувається обмін даними з сервером.

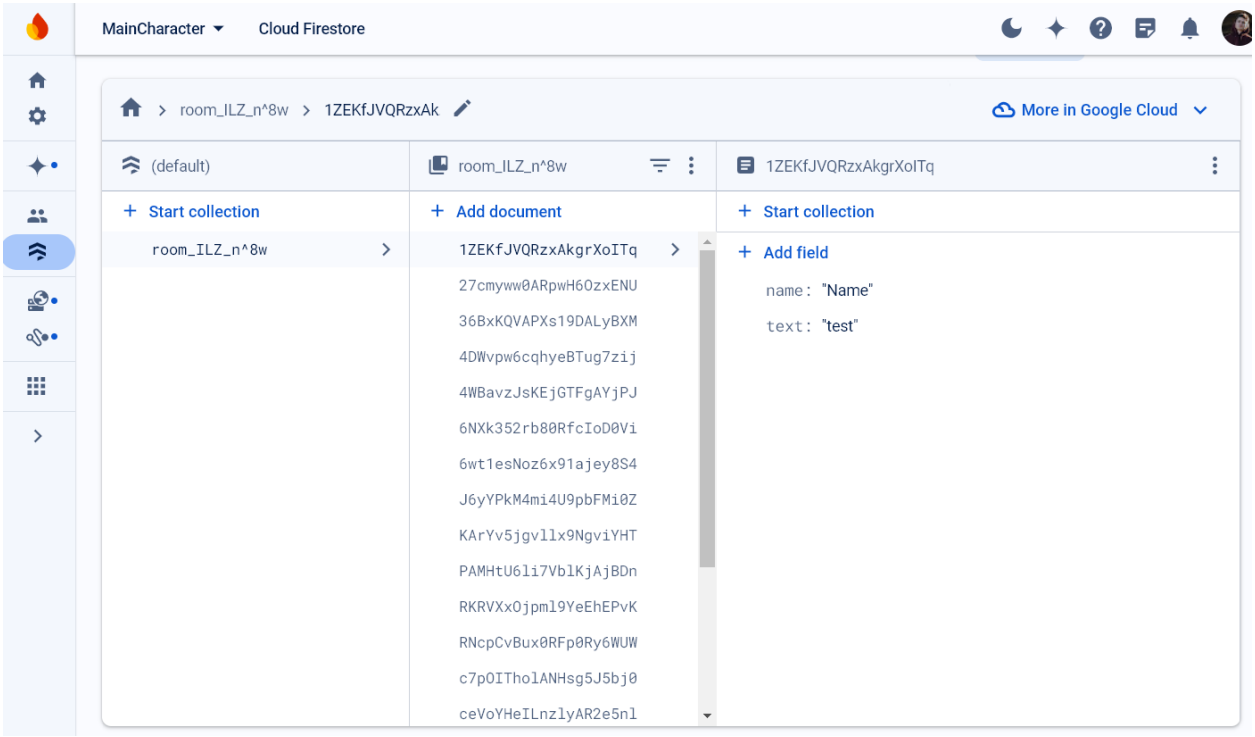


Рисунок 3.4 – Кімната Firebase

Головний функціонал додатку – це спільний перегляд та спілкування у реальному часі(рис. 3.5). Для реалізації цього функціоналу було використано код підключення до Firebase. Оскільки Firebase надає можливість спілкування з базою даних у реальному часі, то це дало можливість оновлювати чат, та маніпулювати відео без перезавантаження сторінки та майже миттєво.

Код підключення до Firebase:

```
const app = initializeApp({
  apiKey: "AIzaSyDZccz9xtY0KH8eASpX4DQDMS_CIVsXBHI",
  authDomain: "maincharacter-e9dd7.firebaseapp.com",
  projectId: "maincharacter-e9dd7",
  storageBucket: "maincharacter-e9dd7.appspot.com",
  messagingSenderId: "725562149817",
  appId: "1:725562149817:web:1ffc31206f0e53391f8468"
});
```

Код отримання даних:

```
useEffect(() => {
    const fetchData = async () => {
      try {
        const querySnapshot = await getDocs(citiesRef);
        const data = querySnapshot.docs.map(doc => ({
          id: doc.id,
          ...doc.data()
        }));
        setMessages(data);
        setLoadingMessages(false);
        console.log(data);
      } catch (error) {
        console.error('Error fetching messages:', error);
      }
    };

    fetchData();
  }, []);
```

Код задавання даних:

```
  setDoc(doc(citiesRef), {
    name: "Name",
    text: 'test'
  });
```

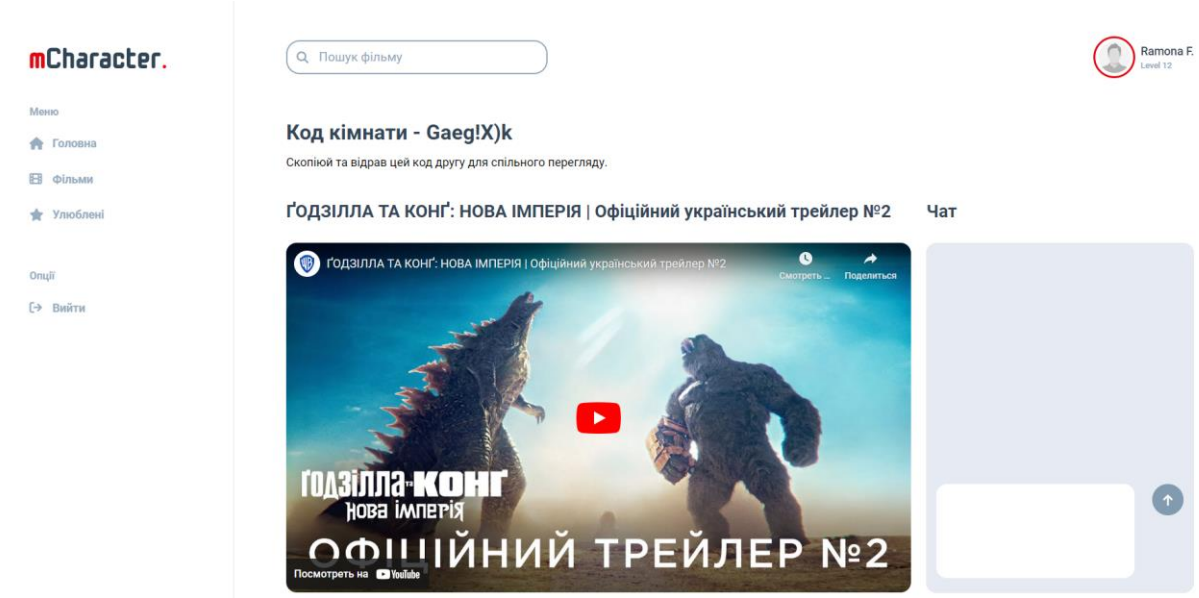


Рисунок 3.5 – Сторінка спільного перегляду

Реалізація пошуку та фільтрації фільмів:

```
const SearchPage = ({ type }) => {
  const { query } = useParams();

  const { data: data, loading: loadingData } =
useFetch(`/search/movie?query=${query}&include_adult=false`);
  const [searchResults, setSearchResults] = useState([]);

  useEffect(() => {
    data && setSearchResults(() => data?.results);
  }, [data]);


  return (
    <div>
      <div className={styles.sectionHead}>
        <h2 className={styles.sectionTitle}>Результати за запитом "{query}"</h2>

      </div>

      {!data ?
        <Grid style={{marginTop: '5rem'}} container columns={10}
columnSpacing='1.5rem' rowSpacing={5}>
          {[...Array(20)].map((_, index) => (
            <Grid key={index} item xs={2} style={{ margin: 0}}>
              <Skeleton width='100%' variant="rectangular" height='22rem'/>
              <Skeleton width='100%' variant="text" sx={{ fontSize: '2rem' }}/>
              <Skeleton width='100%' variant="text" sx={{ fontSize: '1rem' }}/>
            </Grid>
          ))}
```

```
        </Grid>
        :
        <MediaList
          data={searchResults}
          isCollapsed={false}
        />
      }

    </div>
  );
};
```

Для авторизації/реєстрації аккаунту, було обрано функціонал TMDB API. Все що необхідно – це створення сесії, яка буде дійсна максимум 60 хвилин, якщо користувач не користується додатком. Якщо ж користувач активно використовує додаток, то сесія не буде закрита (Додаток В).

```
  const { data, loading: sessionLdng } = useFetch(`/authentication/token/new`);
  const [session, setSession] = useState([]);
```

3.2 Використання вебдодатку

Вебдодаток розроблений таким чином, що неавторизований користувач може користуватись усім функціоналом окрім додавання фільму до улюблених, та використання функції спільного перегляду з чатом для спілкування у реальному часі. На рисунку 3.6 та рисунку 3.7 зображена головна сторінка вебдодатку в залежності від того авторизований користувач чи ні.

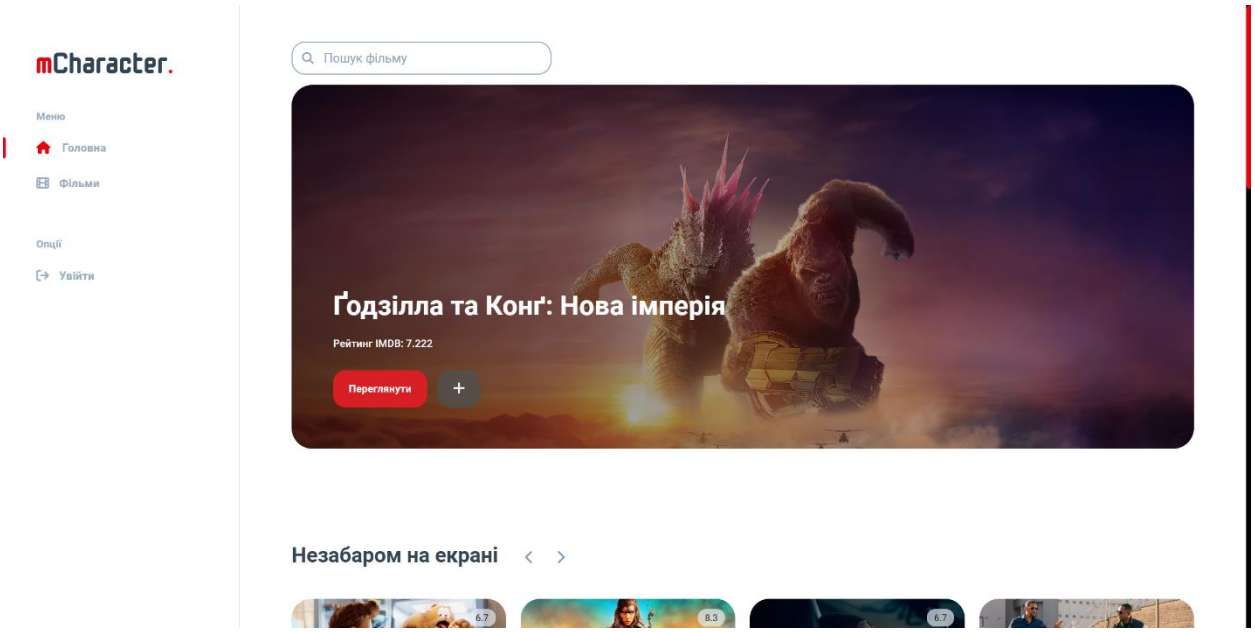


Рисунок 3.6 – Головна сторінка неавторизованого користувача.

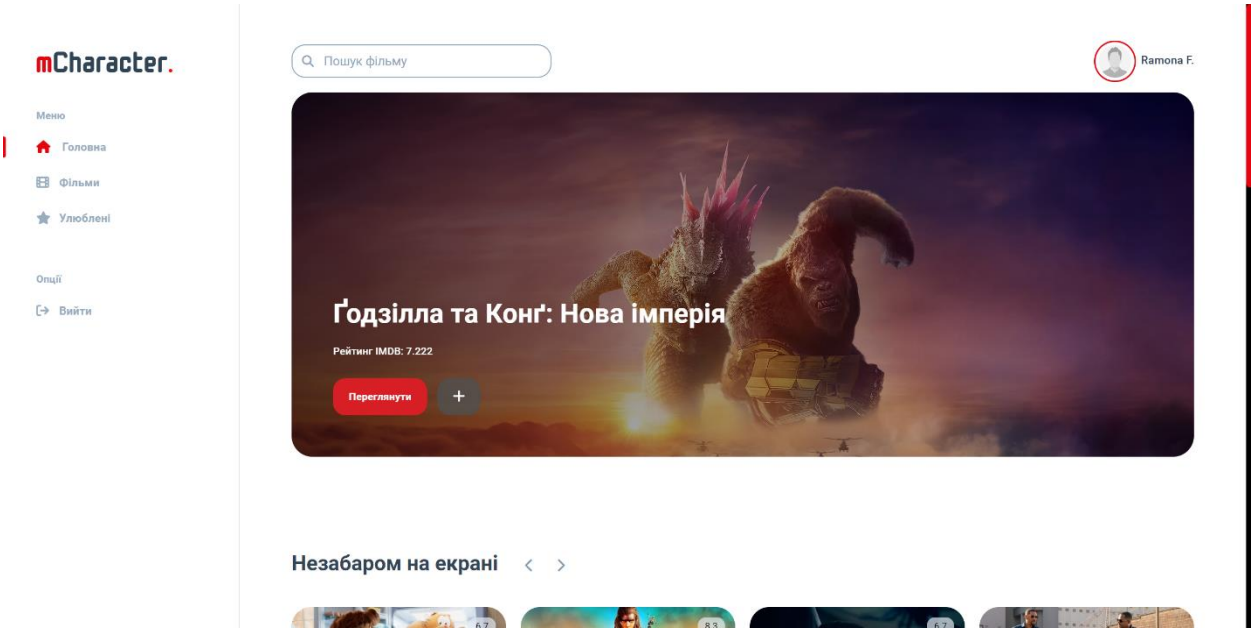


Рисунок 3.7 – Головна сторінка авторизованого користувача.

На головній сторінці вебдодатку розташовані наступні секції:

- Слайдер з найпопулярнішими фільмами на момент використання вебдодатку.
- Секція «Незабаром на екрані» з фільмами, що незабаром можна буде подивитись в кінотеатрі (рис. 3.8).
- Секція «Популярні фільми» з фільмами, що популярні на момент використання вебдодатку (рис. 3.8).

- Секція «Топ рейтингів» з найпопулярнішими фільмами за весь час (рис. 3.7).
- Секція «Зараз на екранах» з фільмами, що на момент використання додатку, можна подивитись в кінотеатрі (рис. 3.9).

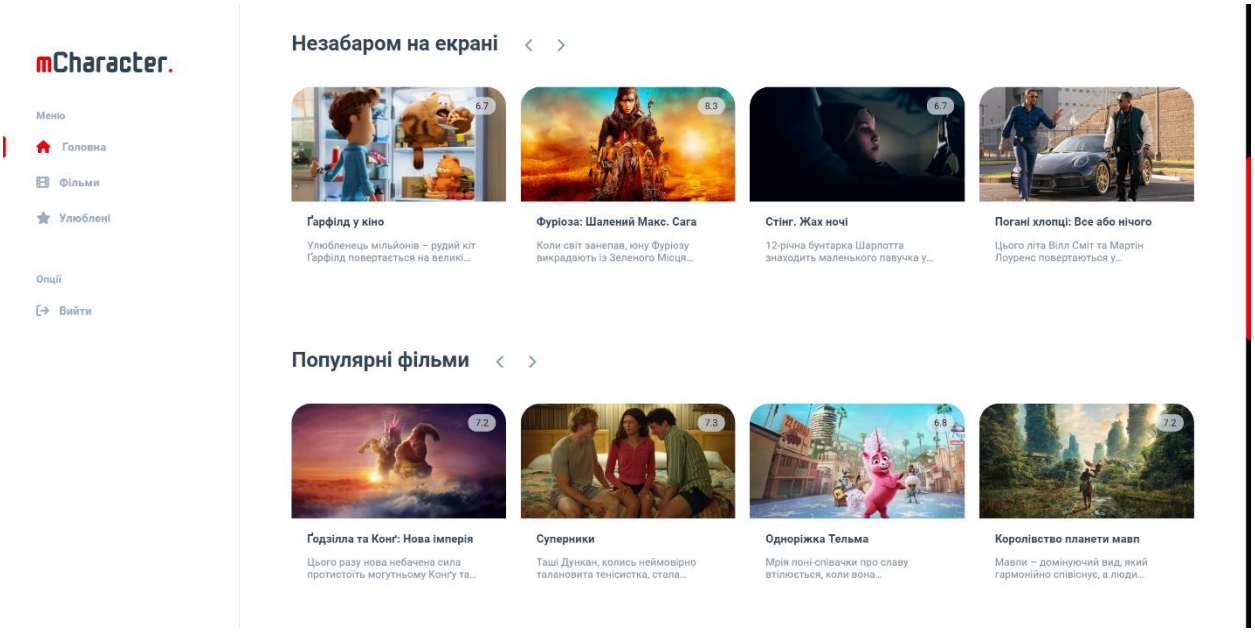


Рисунок 3.8 – Головна сторінка, секції «Незабаром на екрані» та «Популярні фільми»

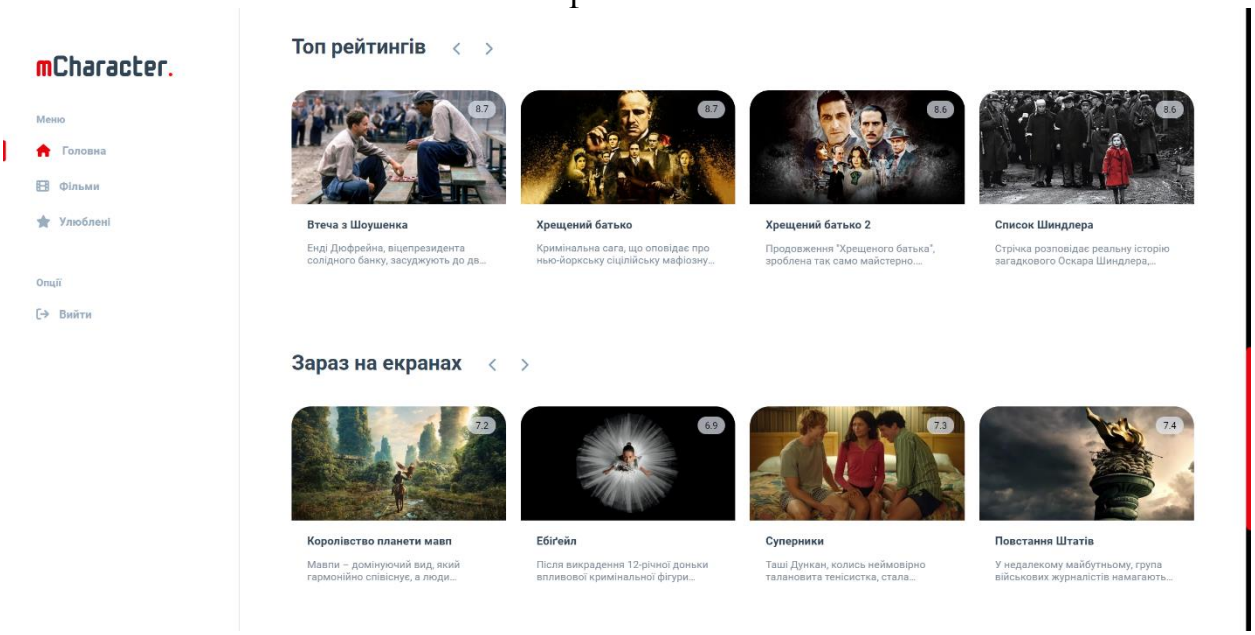


Рисунок 3.9 – Головна сторінка, секції «Топ рейтингів» та «Зараз на екранах»

Сторінка «Фільми», зображена на рисунку 3.10, дає можливість обирати фільми зі списку, фільтрувати фільми за необхідними критеріями та переходити на різні сторінки списку фільмів.

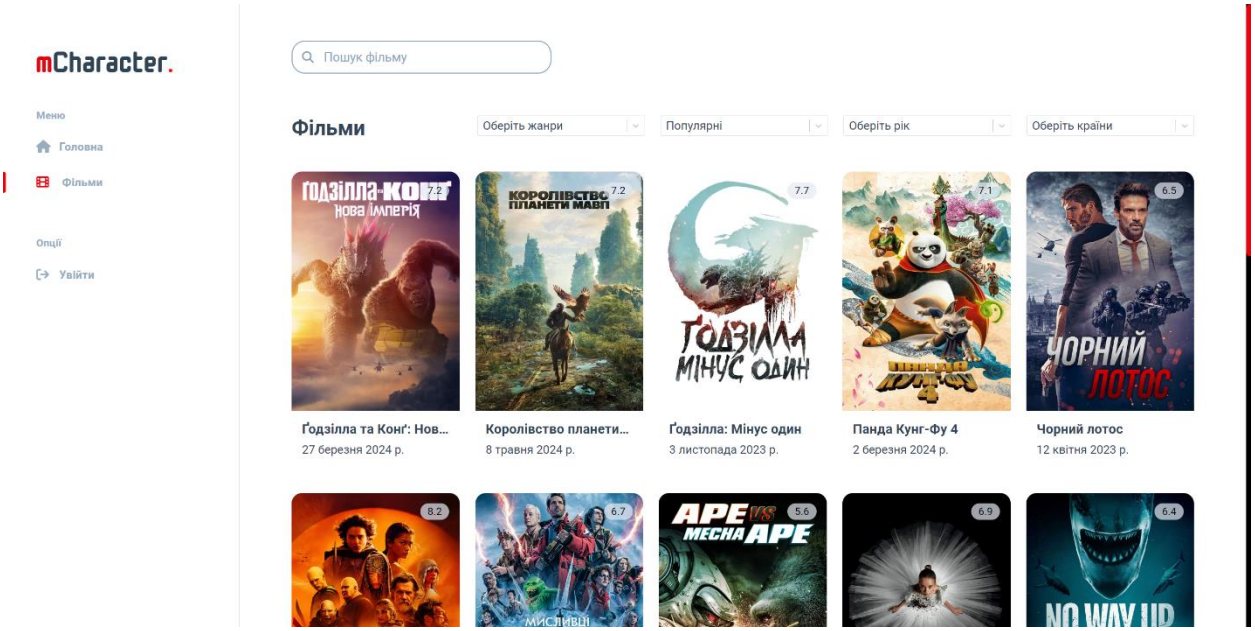


Рисунок 3.10 – Сторінка «Фільми»

Фільтрація відбувається автоматично, після зміни якогось параметру, сторінка не перезавантажується, оновлюється лише список фільмів. Результат простої фільтрації зображено на рисунку 3.11.

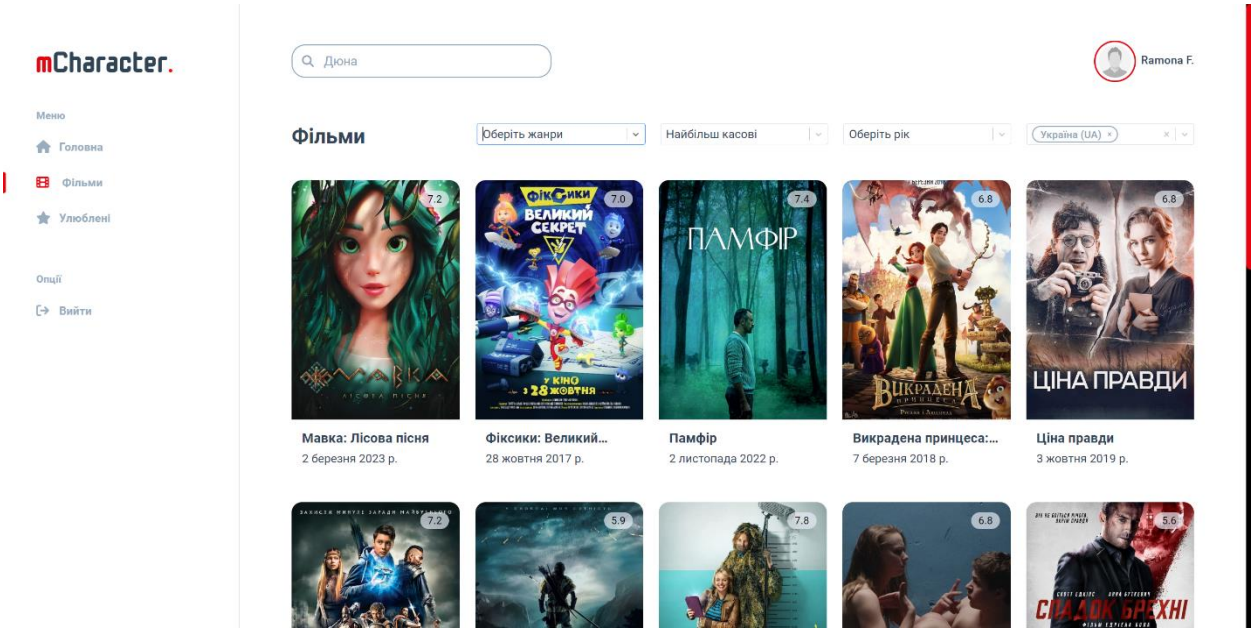


Рисунок 3.11 – Сторінка «Фільми», результат фільтрації.

Відфільтрувавши список фільмів, користувач може обрати один із запропонованих варіантів. Таким чином ми перейдемо на сторінку фільму зображену на рисунку 3.12.

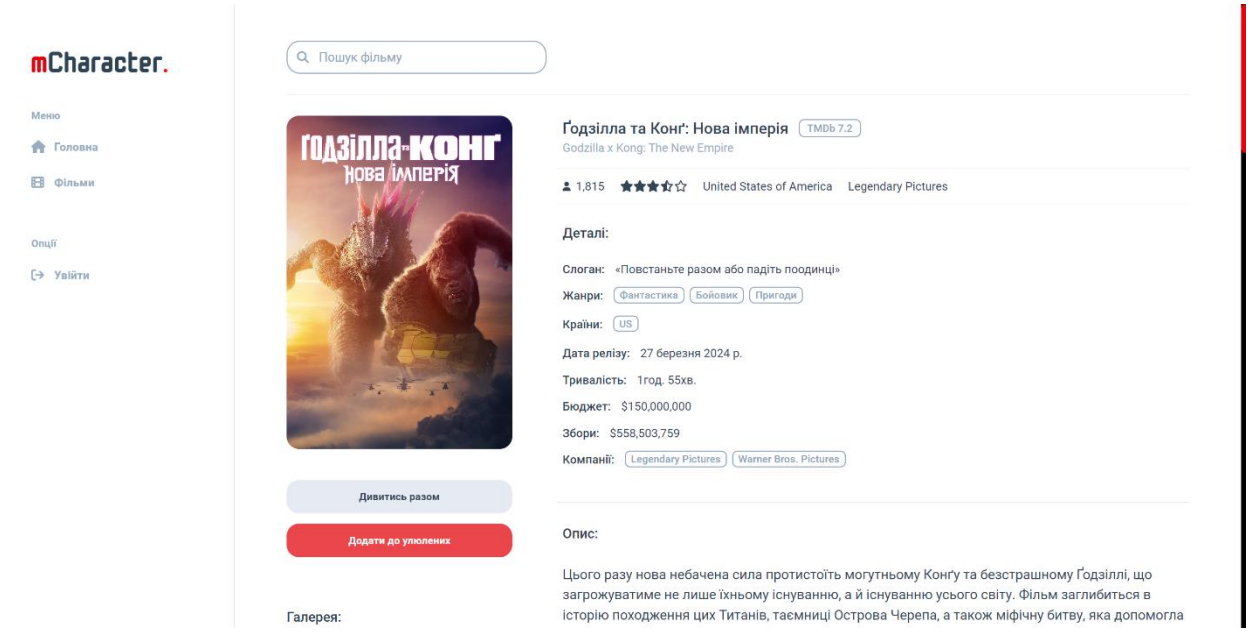


Рисунок 3.12 – Сторінка обраного фільму.

На сторінці конкретного фільму можна побачити детальну інформацію про фільм, та списки людей, які були задіяні для роботи над ним. Також під головним постером фільму можна побачити дві кнопки:

- «Дивитися разом», що створить кімнату для спільного перегляду та чат для спілкування.
- «Додати до улюблених», що додасть фільм до списку улюблених.

Для авторизації потрібно натиснути кнопку «Увійти», яка знаходиться в лівому меню вебдодатку. Відкриється сторінка входу або реєстрації(рис. 3.13).

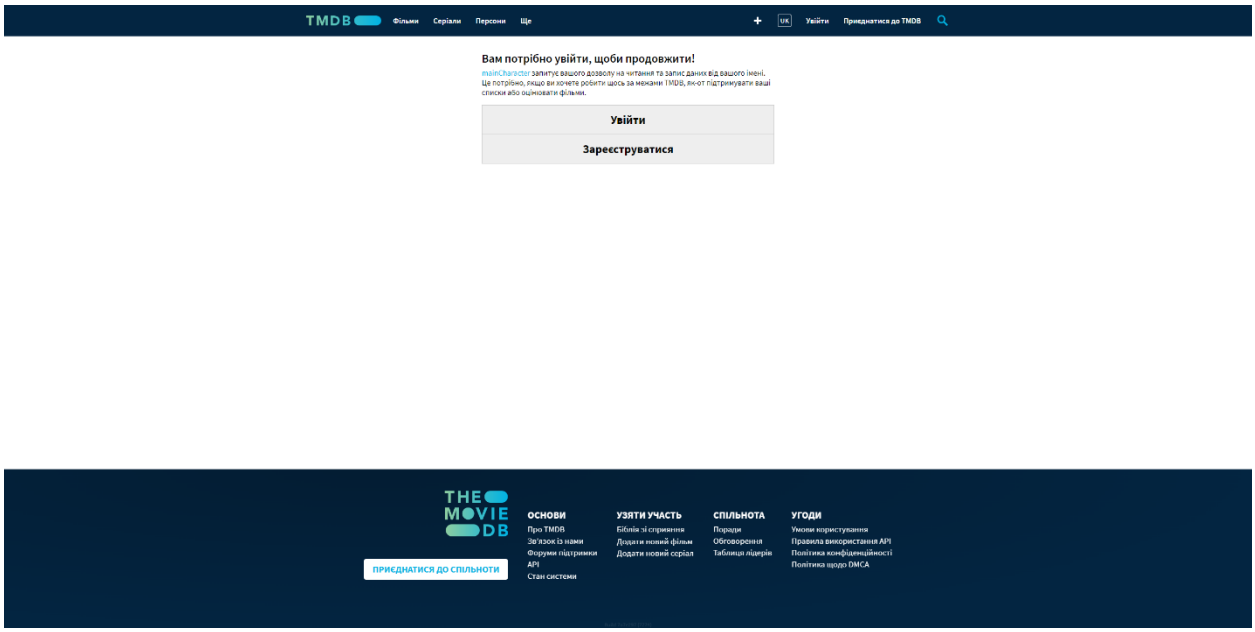


Рисунок 3.13 – Сторінка входу або реєстрації.

Натиснувши кнопку «Увійти», якщо користувач правильно введе логін та пароль, він буде авторизований в системі, що дозволить створити сесію, та користуватись функціоналом додавання до улюблених та спільного перегляду фільмів. Також, можна натиснути кнопку зареєструватись (рис. 3.14), що дозволить створити аккаунт, та користуватись повним доступом до вебдодатку.

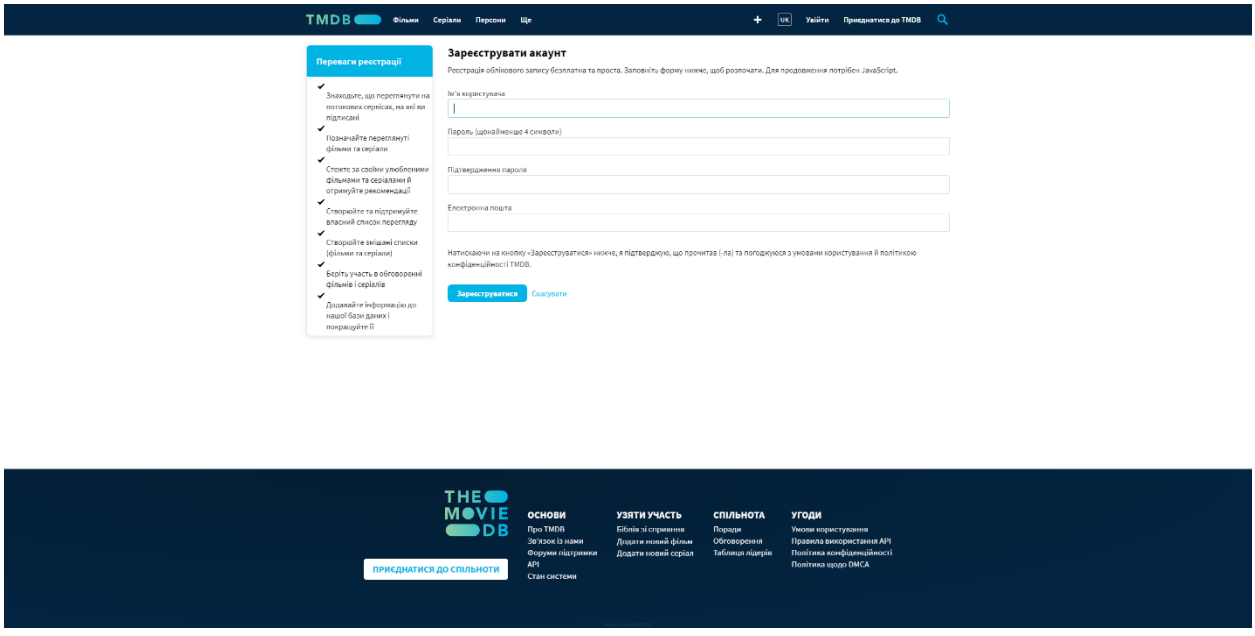


Рисунок 3.14 – Сторінка реєстрації.

Авторизувавшись, користувач може скористатись функцією спільного перегляду та спілкування за допомогою чату в реальному часі (рис. 3.15). Для того щоб інші користувачі змогли приєднатись до кімнати, потрібно скопіювати згенероване посилання, та відправити іншому користувачу.

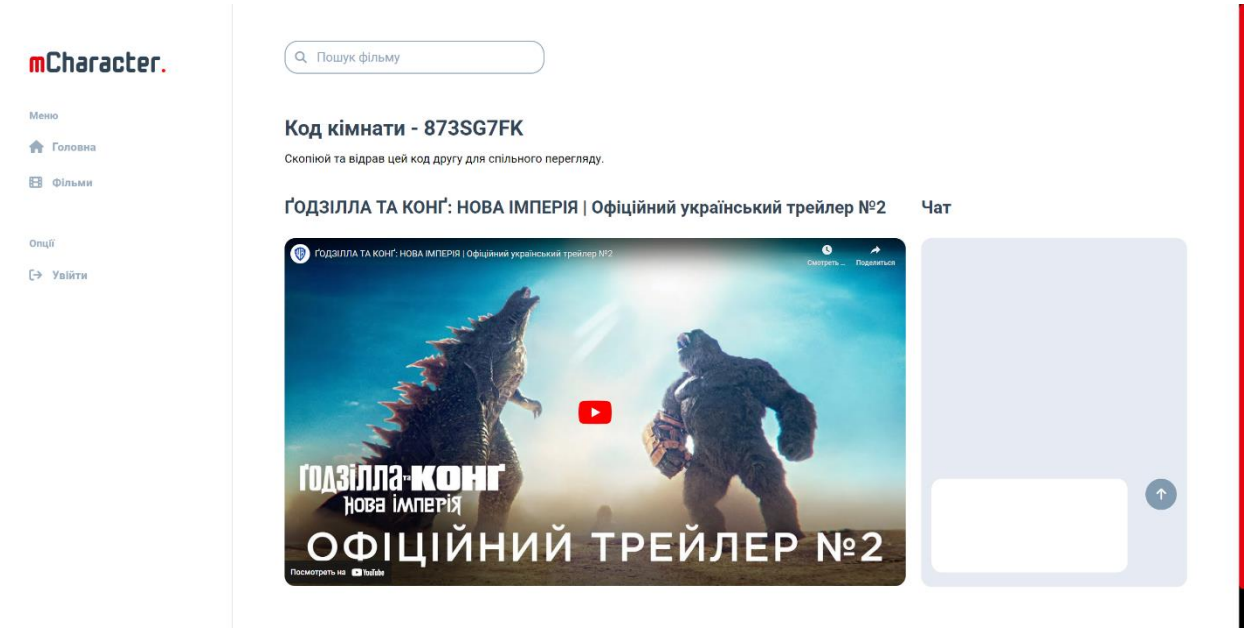


Рисунок 3.15 – Сторінка спільного перегляду.

Відправивши повідомлення інший користувач побачить це повідомлення в чаті, на рисунку 3.16 зображено початок перегляду, та обмін повідомленнями під час перегляду. Коли користувач 1 ставить на паузу фільм, то він автоматично ставиться на паузу у всіх користувачів. Такий функціонал доступний всім авторизованим користувачам, що знаходяться в кімнаті спільного перегляду.

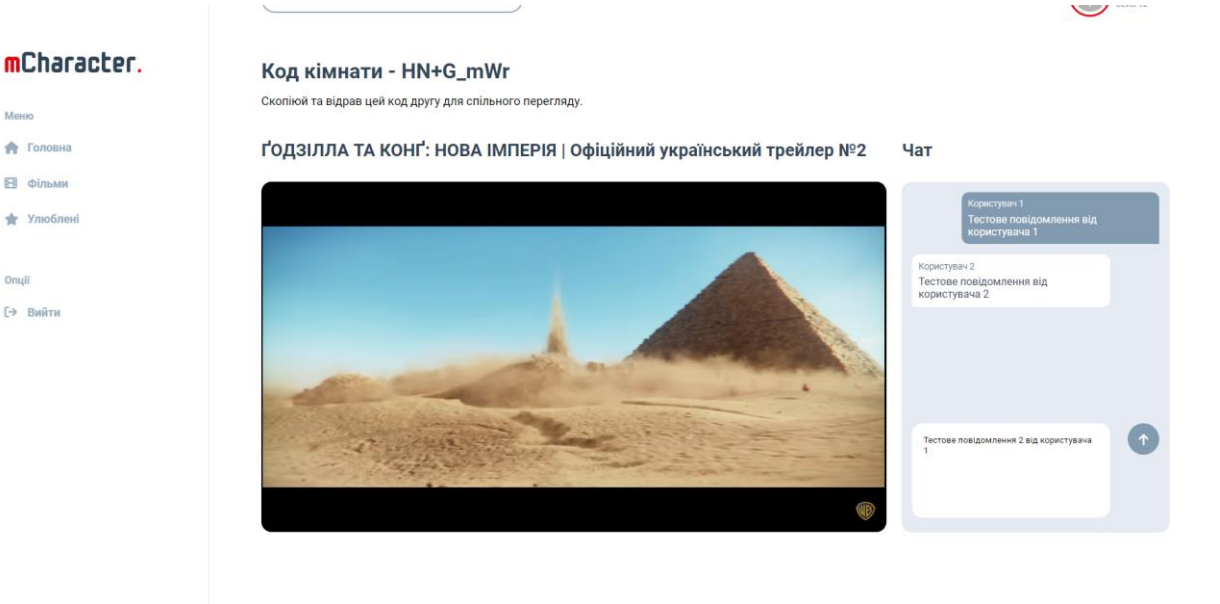


Рисунок 3.16 – Сторінка спільного перегляду.

Сторінка пошуку (рис. 3.17) реалізована таким чином, що результати генеруються за ключовим словом незалежно від мови.

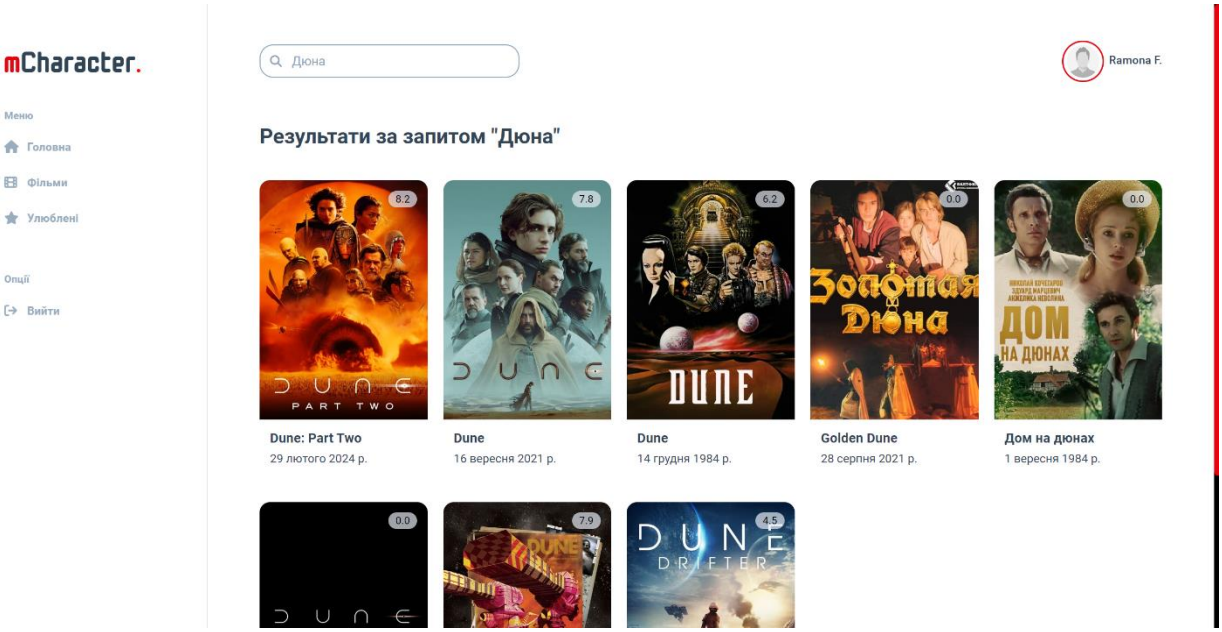


Рисунок 3.17 – Сторінка пошуку.

Сторінка улюблених фільмів зображена на рисунку 3.18. Для додавання фільмів до списку улюблених потрібно перейти на сторінку конкретного фільму(рис. 3.12), та натиснути кнопку «Додати до улюблених».

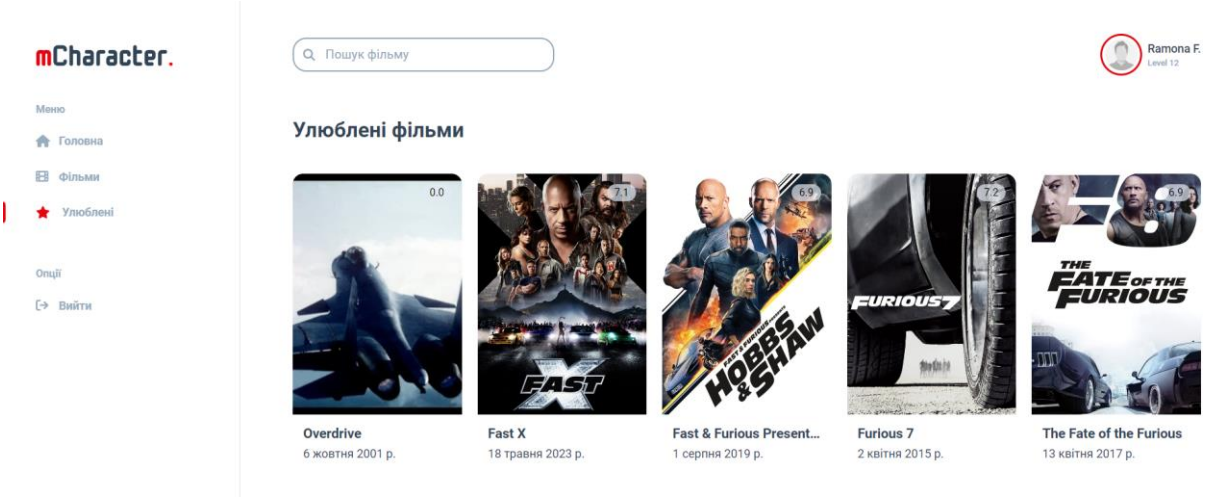


Рисунок 3.18 – Сторінка улюблених фільмів.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи бакалавра було розроблено вебдодаток для спільного перегляду фільмів та спілкування в реальному часі. Для реалізації цього проєкту був здійснений детальний аналіз предметної області та проведено огляд сучасних досліджень і публікацій. Це підтвердило актуальність теми та високу потребу у рішеннях, що підтримують синхронізований перегляд відео та інтерактивне спілкування. Проаналізовано існуючі аналоги, виявлені їхні сильні та слабкі сторони. На основі цього були сформульовані вимоги, завдання та цілі проєкту. Вибрано відповідні технології для розробки.

Було виконано моделювання та проектування:

- Створено діаграму в нотації IDEF0 та її декомпозицію для детального розуміння процесів та їх взаємозв’язків.
- Розроблено діаграму варіантів використання, яка демонструє, як користувачі взаємодіють з вебдодатком.
- Розроблено модель бази даних Firebase, що взаємодіє з вебдодатком.
- Представлено архітектуру вебдодатку, програмну реалізацію основних модулів та інструкції щодо використання продукту.

Підсумком роботи став функціональний вебдодаток для спільного перегляду фільмів та реального часу чату, який сприятиме зміцненню соціальних зв'язків та забезпечить користувачам приємний досвід спільного перегляду фільмів, незалежно від їхнього місцезнаходження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gaustad T. Nordic Journal of Media Studies. *www.researchgate.net*. URL: https://www.researchgate.net/publication/334070058_How_streaming_services_make_cinema_more_important (дата звернення: 05.05.2024).

2. IMDb | Help. IMDb | Help. URL: https://help.imdb.com/article/imdb/general-information/what-is-imdb/G836CY29Z4SGNMK5?ref_=helpsect_cons_1_1 (дата звернення: 05.05.2024).

3. The Movie Database. The Movie Database (TMDB). URL: https://www.themoviedb.org/about (дата звернення: 05.05.2024).

4. Academic Journals and Conferences |. URL: https://science.lpnu.ua/sites/default/files/journal-paper/2019/apr/16490/semi12019-55-61.pdf (дата звернення: 04.05.2024).

5. Global Online Video Platforms Market Opportunity Analysis | 2023-2030. Top Market Research Intelligence and Consulting Firm | SkyQuest Technology Consulting Pvt. Ltd. URL: https://www.skyquestt.com/report/online-video-platforms-market#:~:text=Global%20Online%20Video%20Platforms%20Market%20Insights,period%20(2024-2031) (date of access: 07.05.2024).

6. Online Video Platform Market Size, Trends, Share Analysis And Forecast To 2033. Global Market Research Reports & Consulting | The Business Research Company. URL: https://www.thebusinessresearchcompany.com/report/online-video-platform-global-market-report (date of access: 04.05.2024).

7. The State of Video Streaming in 2023: A Special Report. Variety. URL: https://variety.com/vip-special-reports/the-state-of-video-streaming-in-2023-a-special-report-1235607613/ (date of access: 04.05.2024).

8. Video Streaming Market Size To Reach $416.84Bn By 2030. Market Research Reports & Consulting | Grand View Research, Inc. URL: https://www.grandviewresearch.com/press-release/global-video-streaming-market (date of access: 04.05.2024).

9. Which Consumer Attitudes Will Shape the Streaming Wars?. Nielsen. URL: https://www.nielsen.com/insights/2020/playback-time-which-consumer-attitudes-will-shape-the-streaming-wars/ (date of access: 04.05.2024).

10. Corporate - Hulu. Hulu. URL: https://press.hulu.com/corporate/ (дата звернення: 05.05.2024).

11. Disney+ - Stream Unlimited Movies and TV Series. Disney+. URL: https://www.disneyplus.com/ (дата звернення: 05.05.2024).

12. Netflix. Netflix. URL: https://www.netflix.com/ua/ (date of access: 04.05.2024).

13. Amazon.de Sign up for Prime Video. Amazon.de Sign up for Prime Video.
URL: https://www.primevideo.com/offers/nonprimehomepage/ref=dv_web_force_root (date of access: 05.05.2024).

14. Apple TV+ (UA). Apple TV. URL: https://tv.apple.com/ua?l=uk-UA (date of access: 04.05.2024).

15. Проектування інтерфейсу користувача веб-сайту з використанням функціонально-орієнтованого підходу. DSpace :: ELAKPI :: Репозитарій КПІ ім. Ігоря Сікорського. URL: https://ela.kpi.ua/items/35b48209-367e-4c1b-b4a7-573bd7dd223b (дата звернення: 05.05.2024).

16. Figma Tutorial. Figma. URL: https://help.figma.com/hc/en-us/sections/4405269443991-Figma-for-Beginners-tutorial-4-parts (date of access: 05.05.2024).

17. Rapid prototyping UX/UI продуктів. YellowArrow.Design. URL: https://yellowarrow.design/index.php/ua/blog-article/84-rapid-prototyping-ux-ui (дата звернення: 05.05.2024).

18. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. URL: https://uk.legacy.reactjs.org/ (дата звернення: 07.05.2024).

19. Structural-functional modeling for the determination of the company's equilibrium conditions in the dynamic business environment | Academic Journals and Conferences. *Academic Journals and Conferences* |. URL: https://science.lpnu.ua/mmc/all-volumes-and-issues/volume-7-number-1-2020/structural-functional-modeling-determination (date of access: 18.05.2024).

20. Use Case Diagrams | Unified Modeling Language (UML) - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/use-case-diagram/ (date of access: 26.05.2024).

21. Hanna K. T., Rosencrance L. What is Google Firebase?. *Mobile Computing*.
URL: https://www.techtarget.com/searchmobilecomputing/definition/Google-Firebase#:~:text=Firebase%20provides%20a%20secure%20and,Realtime%20Database. (date of access: 19.05.2024).

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку вебдодатку для спільного перегляду фільмів та спілкування в реальному часі

ПОГОДЖЕНО:

Доцент кафедри
інформаційних технологій

_____ доц. Нагорний В.В.

Студент групи ІТ-02

_____ Оберемко П.Ю.

Суми 2024

1. Призначення й мета вебдодатку

1.1 Призначення вебдодатку

Вебдодаток спрямований на створення інтерактивного онлайн кінотеатру, де користувачі можуть спільно насолоджуватися переглядом фільмів в реальному часі.

Вебдодаток надає можливість синхронного перегляду, спілкування та співпраці з іншими учасниками, створюючи новий рівень взаємодії під час кінопоказу.

Вебдодаток прагне забезпечити захист особистих даних користувачів та забезпечувати неповторний кінематографічний досвід у віртуальному середовищі.

1.2 Мета створення вебдодатку

Мета проєкту - розробка вебдодатку для спільного перегляду фільмів у реальному часі з можливістю спілкування та взаємодії між користувачами.

Головний акцент - створення синхронного середовища для спільного кінопоказу та взаємодії учасників для покращення соціального кінематографічного досвіду.

1.3 Цільова аудиторія

Цільовою аудиторією даного проєкту є особи будь-якої статі та віку.

2. Вимоги до проєкту

2.1 Вимоги до проєкту в цілому

2.1.1 Вимоги до структури й функціонування

Вебдодаток має забезпечувати можливість спільного перегляду фільмів в реальному часі для користувачів будь-якого віку. Користувачі повинні мати доступ до синхронного перегляду фільмів та можливості спілкування під час перегляду. Вебдодаток повинен мати зручний та інтуїтивно зрозумілий інтерфейс для вибору та відтворення фільмів, а також для взаємодії з іншими учасниками.

Основний акцент - створення системи стимулювання активності та участі у спільному перегляді. Додаток повинен бути доступним для використання на різних пристроях, зокрема на комп'ютерах, планшетах та смартфонах.

2.1.2 Вимоги до розробників

Команда розробників повинна мати доступ до стабільного Інтернет-з’єднання та власних комп’ютерів з необхідними ресурсами для розробки вебдодатків. Вимагається досвід роботи з веб-технологіями, такими як JavaScript, React.js, робота з базами даних та системами безпеки.

2.1.3 Вимоги до збереження інформації

Додаток повинен мати систему автоматизованого збереження даних користувачів, що забезпечить надійність та безпеку інформації. Система збереження повинна автоматично фіксувати користувацьку активність, щоб уникнути можливої втрати даних в разі втрати з’єднання чи інших технічних проблем.

2.1.4 Вимоги до розмежування доступу

Вебдодаток повинен бути загальнодоступним для користувачів у мережі Інтернет, з урахуванням відповідних обмежень доступу до контенту для користувачів з територій Росії та Білорусії. Доступ до функціоналу додатку буде розмежований на дві групи: авторизовані та неавторизовані користувачі.

Авторизовані користувачі повинні мати доступ до всіх функціональних можливостей вебдодатку, проте вони не повинні мати права на його редагування або модифікацію. Їхні можливості мають обмежуватися виключно переглядом контенту та взаємодією з функціями, доступними для глядачів у межах вебдодатку.

Неавторизовані користувачі повинні мати обмежений доступ до всіх функціональних можливостей вебдодатку. Їхні можливості мають обмежуватися виключно переглядом контенту.

2.2 Структура вебдодатку

2.2.1 Загальна інформація про структуру вебдодатку

Структура вебдодатку включає наступні ключові елементи, що становлять основні функціональні блоки системи:

- Головна сторінка, де користувачі отримують доступ до загальної інформації про додаток та рекомендаційних фільмів;
- Каталог фільмів та серіалів, що включає доступ до контенту за різними жанрами, топами додатку, рекомендаціями, топами акторів, режисерів та інших категорій;
- Персональна сторінка фільму призначена для отримання користувачами детальної інформації про конкретний фільм або серіал;
- Улюблені фільми, де користувачі можуть зберігати свої улюблені твори для майбутнього перегляду та отримувати рекомендації на їх основі;
- Сторінка спільного перегляду з чатом, де користувачі можуть

разом з іншими користувачами синхронно переглядати фільм та обмінюватись повідомленнями;

- Вихід, що дозволяє користувачам безпечно вийти з облікового запису в додатку.

2.2.2 Навігаційне меню

Навігаційне меню вебдодатку містить основні розділи, що включають головну сторінку, каталог фільмів для перегляду та вибору контенту, сторінку улюблених фільмів, а також функцію пошуку для зручного пошуку конкретного контенту за різними критеріями.

2.2.3 Управління контентом

Управління контентом вебдодатку має здійснюватися розробником за допомогою коду. Усе інформаційне наповнення вебдодатку має міститися у базі даних TMDb API.

2.2.4 Дизайн вебдодатку

Дизайн вебдодатку має бути сучасним та зручним для користувача. Кольорова палітра буде базуватися на темних і світлих відтінках, з можливим використанням акцентних кольорів для важливої інформації.

Важливо, щоб розміщення елементів було логічним для зручної навігації та комфортного використання додатку. Тестовий варіант інтерфейсу користувача зображено на рисунку А.1

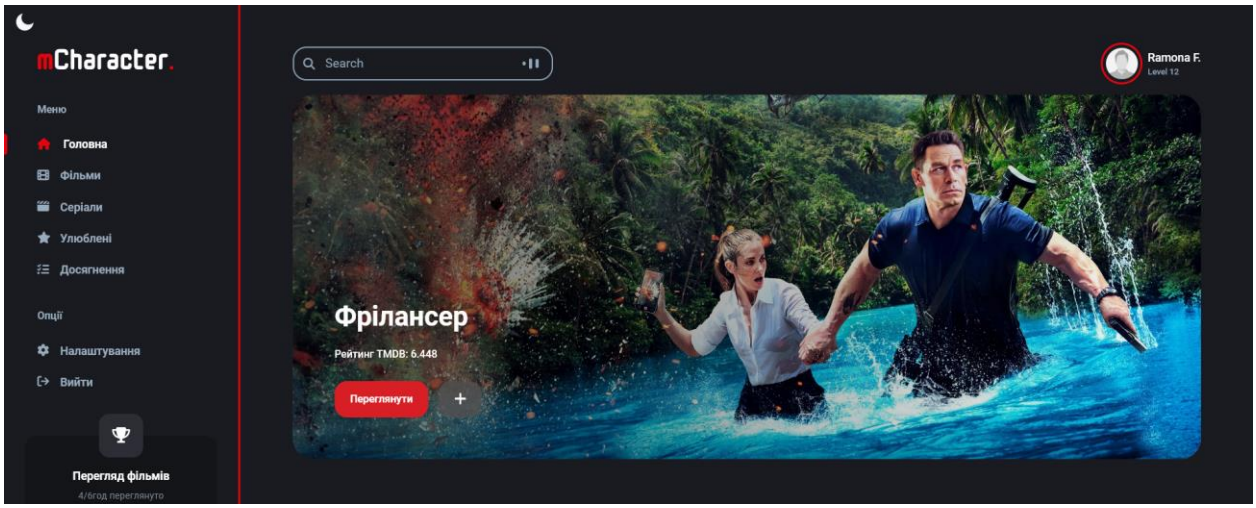


Рисунок А.1 – Частина головної сторінки додатку

2.2.5 Система навігації (карта вебдодатку)

Карта вебдодатку зображена на рисунку А.2.

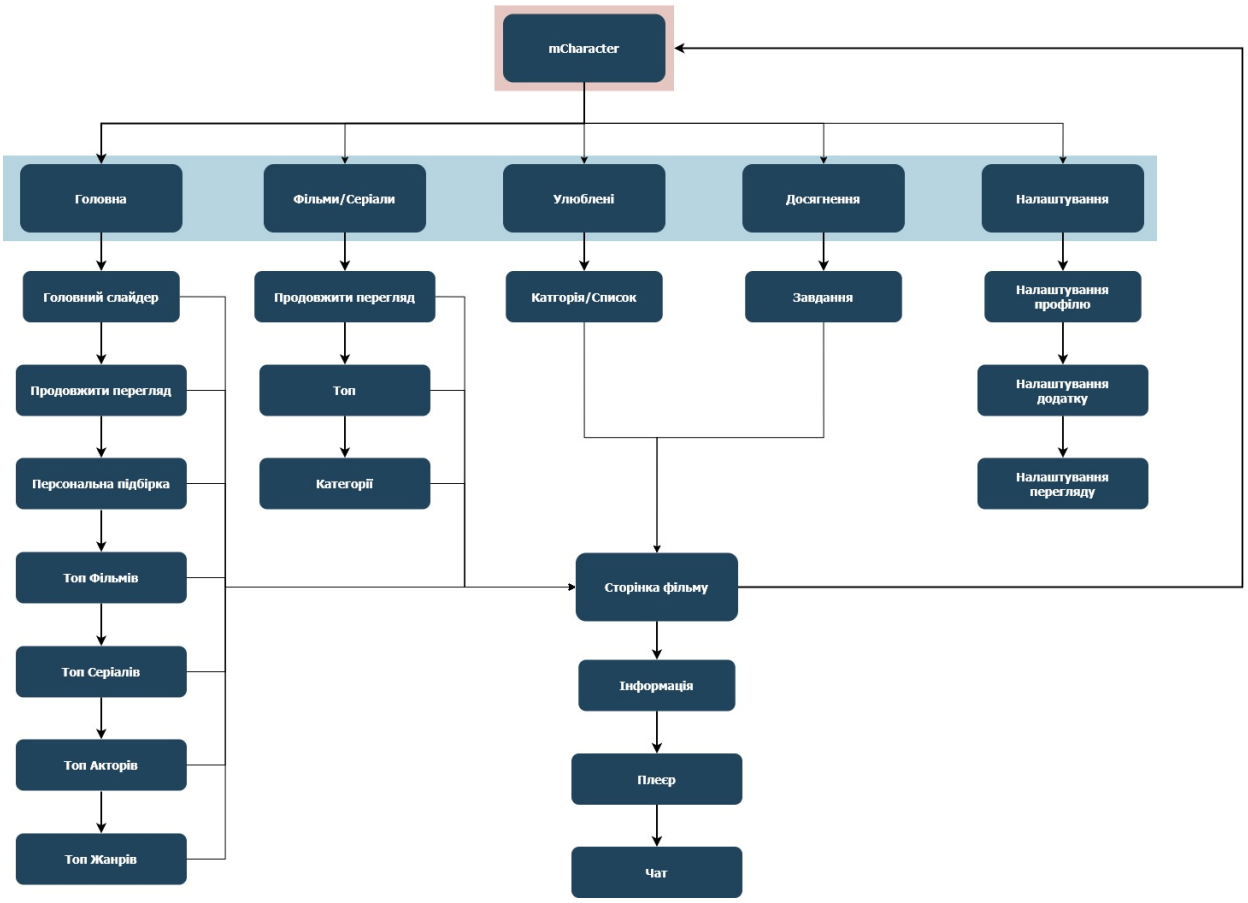


Рисунок А.2 – Система навігації

2.3 Вимоги до видів забезпечення

2.3.1 Вимоги до лінгвістичного забезпечення

Все текстове та звукове наповнення додатку має бути виконане українською або англійською мовами.

2.3.2 Вимоги до програмного забезпечення

Для забезпечення стабільної роботи додатку версія браузера повинна підтримувати JavaScript, а також веб-браузер має бути Internet Explorer 7.0 і вище, або Firefox 3.5 і вище, або Opera 9.5 і вище, або Safari 3.2.1 і вище, або Chrome 2 і вище.

2.4 Вимоги до функціонування системи

2.4.1 Потреби користувача

Потреби користувача, визначені та представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

ID	Потреби користувача	Джерело
UN-01	Перегляд списку фільмів/серіалів для вибору контенту	Клієнт
UN-02	Можливість пошуку фільмів/серіалів за жанром, рейтингом, акторами	Клієнт
UN-03	Перегляд детальної інформації про фільм/серіал: опис, трейлери, рейтинги	Клієнт
UN-04	Зберігання фільмів у списку улюблених для подальшого доступу	Клієнт

Продовження таблиці А.1

ID	**Потреби користувача**	**Джерело**
UN-05	Спільний перегляд фільмів у режимі онлайн з іншими користувачами	Клієнт
UN-06	Можливість виставлення рейтингу фільмів та залишення відгуків	Клієнт
UN-07	Перегляд особистого профілю з історією перегляду та досягнень	Клієнт
UN-08	Редагування контенту	Розробник
UN-09	Моніторинг та аналіз даних про використання додатку(ведення статистики за допомогою Google Аналітики)	Розробник

У таблиці потреб користувача А.1 наведені усі можливі категорії користувачів:

- Клієнт – особа, яка використовує додаток;
- Розробник – особа, яка змінює основний контент.

2.4.2 Системні вимоги

Проаналізувавши потреби користувачів було визначено наступні вимоги:

- Авторизація та аутентифікація користувачів;
- Система рекомендацій, що базується на історії переглядів користувача, жанрах, акторах, рейтингах фільмів тощо;
- Функціонал пошуку фільмів та серіалів за різними критеріями: жанр, рік випуску, режисер, актор тощо;
- Можливість додавання фільмів у список улюблених, створення власних списків та рейтингування контенту;

- Реалізація можливості синхронного перегляду фільмів користувачами, обмін повідомленнями та реакціями під час перегляду;
- Механізми збереження даних користувачів, включаючи інформацію про переглянуті фільми, вибрані налаштування та список улюблених контенту;
- Вебдодаток для взаємодії користувачів під час перегляду, з можливістю обміну повідомленнями та реакціями в реальному часі;

3 Склад і зміст робіт зі створення вебдодатку

Детальний опис етапів створення додатку наведено в таблиці А.2.

Таблиця А.2 – Етапи створення додатку

№	Склад і зміст робіт	Строк розробки
1	Аналіз вимог та планування	5 днів
2	Розробка технічного завдання	7 днів
3	Проектування	7 дні
4	Дизайн та UX/UI розробка	7 дні
5	Фронтенд розробка	20 днів
6	Бекенд розробка	23 днів
7	Інтеграція та тестування	12 днів
8	Оптимізація та виправлення помилок	10 дні
9	Alpha-тестування	8 дні
10	Beta-тестування	4 дні

11	Написання супровідної документації	6 дні
12	Реліз та запуск додатку	3 дні
	Загальна тривалість робіт	117 днів

4 Вимоги до складу й змісту робіт із введення додатку в експлуатацію

Вебдодаток має бути затверджено та розміщено на вебхостингу.

ДОДАТОК Б

Планування робіт

Розробка й впровадження вебдодатків на сьогоднішній день виявляються важливими етапами для бізнесу в будь-якій сфері. Послуги онлайн-кінотеатру, зокрема, стають все більш популярними та затребуваними серед користувачів, шукаючи зручний та цікавий спосіб перегляду фільмів та взаємодії в онлайн-середовищі. Інноваційні технології не тільки забезпечують зручність у виборі та перегляді вмісту, а й створюють можливість для спільного перегляду фільмів зі знайомими, друзями та онлайн спільнотою.

Із зростанням вимог до комфорту та доступності сервісів сьогоднішній світ вітає новаторські підходи до надання послуг. У сфері розваг, особливо в кінематографії, з'являється потреба в змішаному досвіді, що поєднує онлайн-технології та соціальне спілкування.

Ідея онлайн-кінотеатру, де користувачі можуть не лише переглядати фільми, а й ділитися цим досвідом з друзями чи навіть незнайомими, стає ключовою інновацією у світі кінорозваг. Тут сходяться дві важливі сфери: технології та соціальність, що розкриває нові можливості для заохочення користувачів.

Деталізація мети проєкту методом SMART. Для студента формат постановки SMART-мети такий : «Розробка та запуск інтернет-кінотеатру на React.js із можливістю синхронного перегляду фільмів та виконання завдань для користувачів, розширення функціоналу та впровадження механізмів взаємодії та нагород за участь у перегляді фільмів до кінця другого кварталу 2024 року(6 червня 2024).»

Результати деталізації мети методом SMART розміщені у таблиці Б.1.

Таблиця Б.1 – Деталізація мети проєкту методом SMART

Specific	Розробка інтернет-кінотеатру на React.js з можливостями синхронного перегляду фільмів для користувачів та з реалізованою системою завдань, за які користувачі отримують призи під час перегляду фільмів
Measurable	Встановлення конкретних показників успішності, а саме кількісті користувачів, які можуть одночасно переглядати фільм, а також завдань та призів для користувачів
Achievable	Враховуючи обсяг проєкту та доступні ресурси, ціль реалізована в межах можливого
Relevant	Проєкт відповідає стратегії розвитку бізнесу, спрямованої на розширення аудиторії та поліпшення користувацького досвіду, а також вирішує проблеми доступу до сучасних розважальних можливостей через інтернет та створює спільний досвід для користувачів
Time-framed	Цільна дата завершення - червень 2024 року.

Планування змісту робіт. WBS (Work Breakdown Structure – ієрархічна структура робіт) – це графічний вигляд елементів проєкту, які згруповані ієрархією у єдине ціле з продуктом проєкту. Структура декомпозиції робіт орієнтована на досконале виконання робіт по частинам і сама є ключовою частиною проєкту, яка спрямована на організацію командної роботи. Елементами декомпозиції можуть бути продукти, дані та послуги. Більше того, WBS забезпечує необхідним каркасом для ретельної оцінки термінів та контролю та графіків роботи.

На найвищому (першому) рівні розміщений продукт проєкту. Основні

дії та заходи, що забезпечують досягнення мети проєкту, зафіксовані на другому рівні декомпозиції. Декомпозиція робіт виконується до тих пір, поки вони не стануть елементарними (простими).

Елементарні роботи – це дії, які мають однозначний чіткий результат, на які призначена відповідальному одна конкретна особа, для якої можна обчислити витрати праці і тривалість виконання. На рисунку Б.1 представлено WBS з вебдодатку.

Планування структури виконавців. Наступним етапом після декомпозиції процесів є розробка організаційної структури виконавців або OBS, яка визначається як графічна структура відображення учасників або відповідальних осіб, які беруть участь у реалізації проєкту.

У ролі відповідальних осіб виступають співробітники, що відповідають за організацію і виконання елементарної роботи, що зазначена у WBS. Кожну елементарну роботи можна розглядати як окремий проєкт.

На рисунку Б.2 представлено організаційну структуру планування проєкту. Список виконавців, що функціонують в проєкті описано в таблиці Б.2.

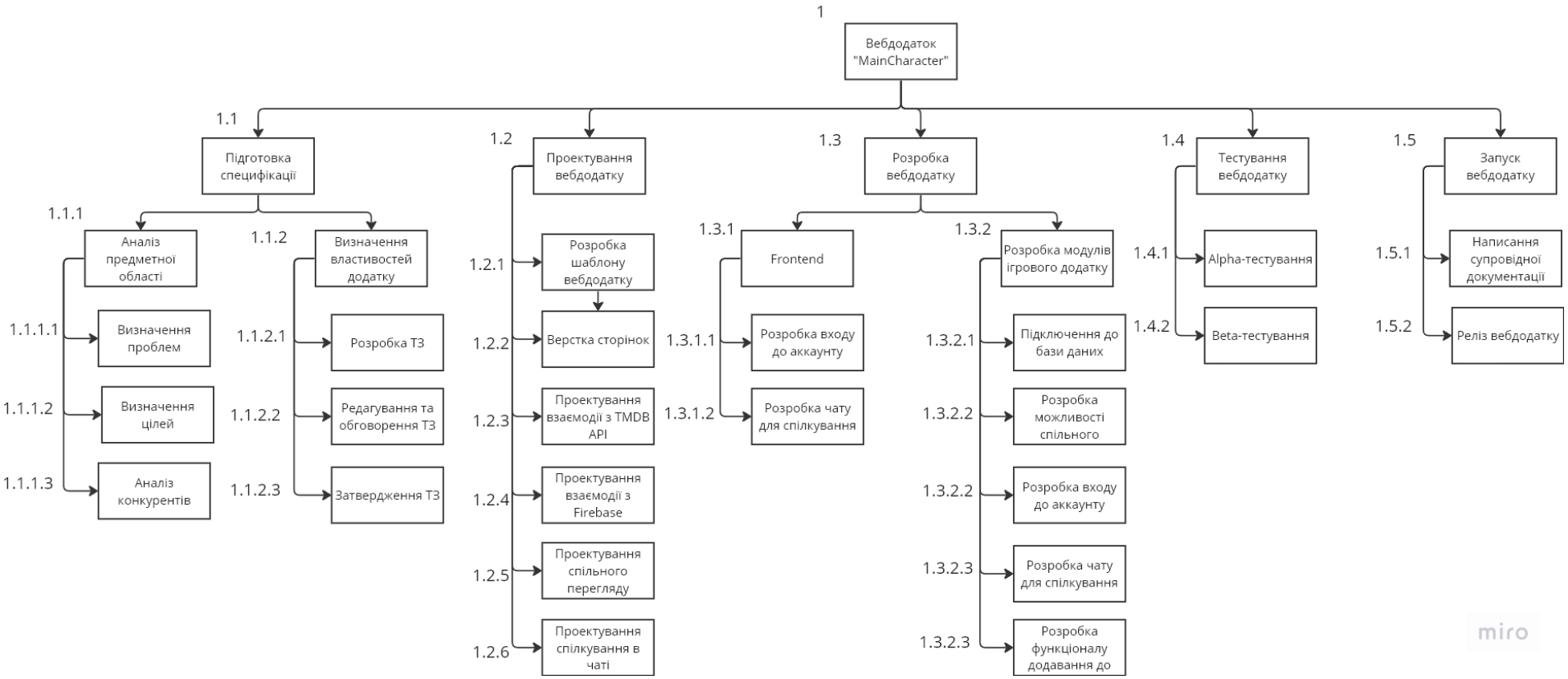


Рисунок Б.1 – WBS-структура робіт проєкту

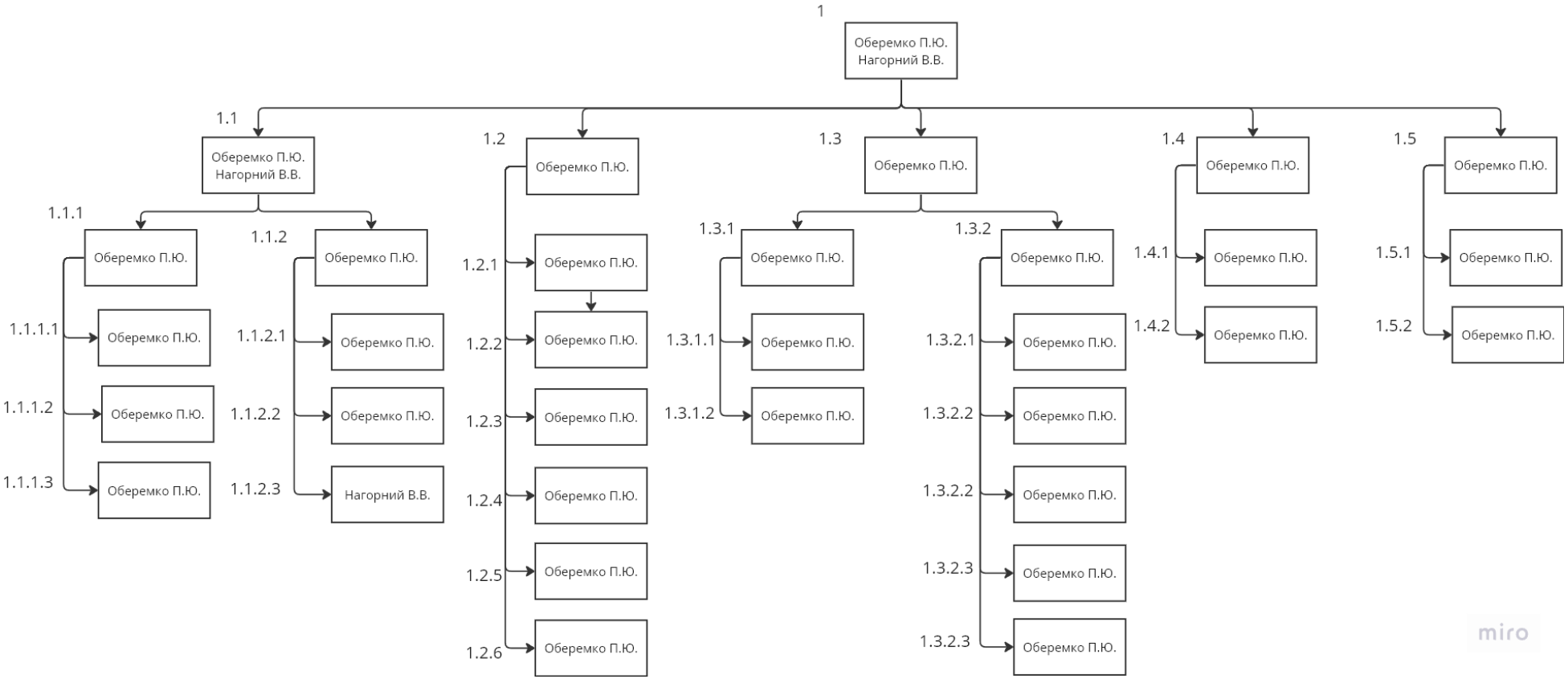


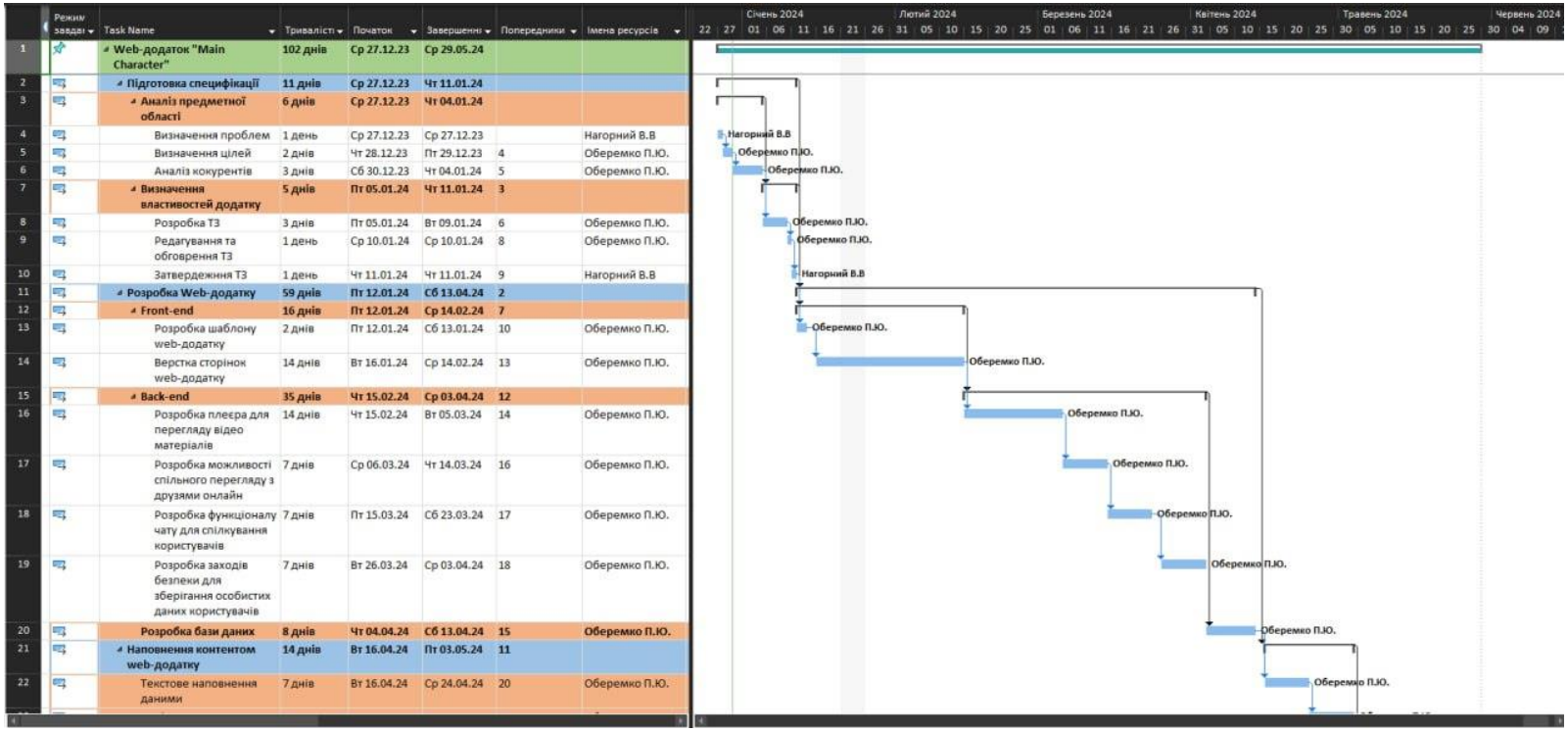
Рисунок Б.2 – OBS-структура робіт проєкту

Таблиця Б.2 – Виконавці проєкту

Роль	ПІБ	Проєктна роль
Розробник	Оберемко П.Ю.	Виконує розробку основних механік проєкту
Дизайнер	Оберемко П.Ю.	Виконує розробку дизайну проєкту
Проектувальник	Оберемко П.Ю.	Відповідає за структуру вебдодатку
Тестувальник	Оберемко П.Ю.	Відповідає за тестування функціоналу вебдодатку
Менеджер проєкту	Оберемко П.Ю.	Відповідає за виконання термінів, розподіл ресурсів та завдань між учасниками
Керівник проєкту	Нагорний В.В.	Формує завдання на розробку проєкту

Діаграма Ганта. Побудова календарного графіку (діаграми Ганта) є одним з важливих етапів планування проєкту, що виглядає як розклад виконання робіт з реальним розподілом дат. Завдяки йому можна отримати достовірне уявлення про тривалість процесів з обмеженнями у ресурсах, урахуванням вихідних днів та свят.

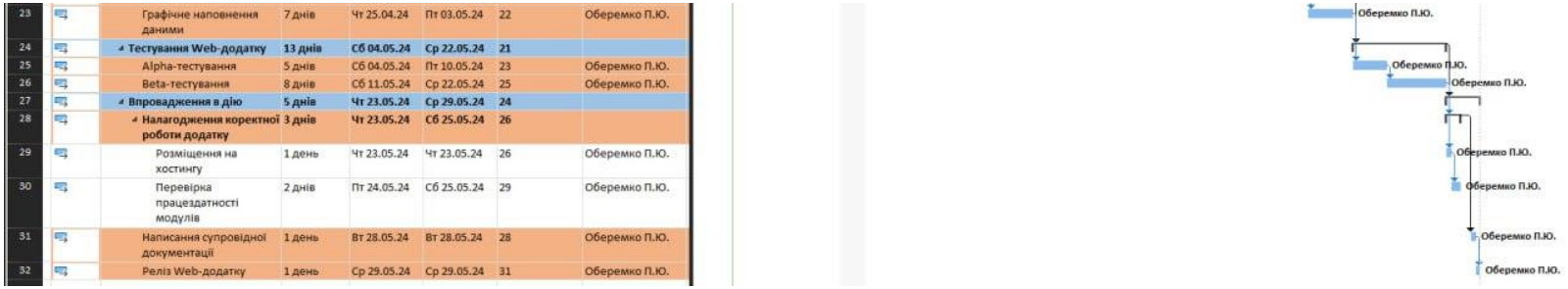
Календарний графік проєкту представлено на рисунках Б.3-Б.4.



	Режим завдан	Task Name	Тривалість	Початок	Завершення	Попередники	Імена ресурсів
1		**Web-додаток "Main Character"**	**102 днів**	**Ср 27.12.23**	**Ср 29.05.24**		
2		**Підготовка специфікації**	**11 днів**	**Ср 27.12.23**	**Чт 11.01.24**		
3		**Аналіз предметної області**	**6 днів**	**Ср 27.12.23**	**Чт 04.01.24**		
4		Визначення проблем	1 день	Ср 27.12.23	Ср 27.12.23		Нагорний В.В
5		Визначення цілей	2 днів	Чт 28.12.23	Пт 29.12.23	4	Оберемко П.Ю.
6		Аналіз кокурентів	3 днів	Сб 30.12.23	Чт 04.01.24	5	Оберемко П.Ю.
7		**Визначення властивостей додатку**	**5 днів**	**Пт 05.01.24**	**Чт 11.01.24**	**3**	
8		Розробка ТЗ	3 днів	Пт 05.01.24	Вт 09.01.24	6	Оберемко П.Ю.
9		Редагування та обговрення ТЗ	1 день	Ср 10.01.24	Ср 10.01.24	8	Оберемко П.Ю.
10		Затвердежння ТЗ	1 день	Чт 11.01.24	Чт 11.01.24	9	Нагорний В.В
11		**Розробка Web-додатку**	**59 днів**	**Пт 12.01.24**	**Сб 13.04.24**	**2**	
12		**Front-end**	**16 днів**	**Пт 12.01.24**	**Ср 14.02.24**	**7**	
13		Розробка шаблону web-додатку	2 днів	Пт 12.01.24	Сб 13.01.24	10	Оберемко П.Ю.
14		Верстка сторінок web-додатку	14 днів	Вт 16.01.24	Ср 14.02.24	13	Оберемко П.Ю.
15		**Back-end**	**35 днів**	**Чт 15.02.24**	**Ср 03.04.24**	**12**	
16		Розробка плеєра для перегляду відео матеріалів	14 днів	Чт 15.02.24	Вт 05.03.24	14	Оберемко П.Ю.
17		Розробка можливості спільного перегляду з друзями онлайн	7 днів	Ср 06.03.24	Чт 14.03.24	16	Оберемко П.Ю.
18		Розробка функціоналу чату для спілкування користувачів	7 днів	Пт 15.03.24	Сб 23.03.24	17	Оберемко П.Ю.
19		Розробка заходів безпеки для зберігання особистих даних користувачів	7 днів	Вт 26.03.24	Ср 03.04.24	18	Оберемко П.Ю.
20		**Розробка бази даних**	**8 днів**	**Чт 04.04.24**	**Сб 13.04.24**	**15**	**Оберемко П.Ю.**
21		**Наповнення контентом web-додатку**	**14 днів**	**Вт 16.04.24**	**Пт 03.05.24**	**11**	
22		Текстове наповнення даними	7 днів	Вт 16.04.24	Ср 24.04.24	20	Оберемко П.Ю.

Січень 2024 | Лютий 2024 | Березень 2024 | Квітень 2024 | Травень 2024 | Червень 2024

Рисунок Б.3 – Календарний графік проєкту



23		Графічне наповнення даними	7 днів	Чт 25.04.24	Пт 03.05.24	22	Оберемко П.Ю.
24		**Тестування Web-додатку**	**13 днів**	**Сб 04.05.24**	**Ср 22.05.24**	**21**	
25		Alpha-тестування	5 днів	Сб 04.05.24	Пт 10.05.24	23	Оберемко П.Ю.
26		Beta-тестування	8 днів	Сб 11.05.24	Ср 22.05.24	25	Оберемко П.Ю.
27		**Впровадження в дію**	**5 днів**	**Чт 23.05.24**	**Ср 29.05.24**	**24**	
28		**Налагодження коректної роботи додатку**	**3 днів**	**Чт 23.05.24**	**Сб 25.05.24**	**26**	
29		Розміщення на хостингу	1 день	Чт 23.05.24	Чт 23.05.24	26	Оберемко П.Ю.
30		Перевірка працездатності модулів	2 днів	Пт 24.05.24	Сб 25.05.24	29	Оберемко П.Ю.
31		Написання супровідної документації	1 день	Вт 28.05.24	Вт 28.05.24	28	Оберемко П.Ю.
32		Реліз Web-додатку	1 день	Ср 29.05.24	Ср 29.05.24	31	Оберемко П.Ю.

Рисунок Б.4 – Продовження календарного графіку проєкту

Управління ризиками проєкту. Під час виконання якісної оцінки ризиків треба визначити ризики, які мають бути усунені якнайшвидше. В залежності від ступеня важливості ризику – реагування буде відповідне. Наступним етапом є виконання кількісного оцінювання ризиків. Кількісне та якісне оцінювання можуть виконувати одночасно або окремо, що залежить від ступеня забезпечення проєкту. У таблиці Б.3 надано перелік ризиків даного проєкту. Результати оцінки ризиків надано у таблиці Б.4. Таблиця Б.5 представляє шкалу для класифікації ризиків за величиною впливу на проєкт та ймовірністю їх виникнення.

Таблиця Б.3 – Ризики проєкту

№ ризику	Назва (опис) ризику
R1	Масові відключення електроживлення
R2	Повітряні тривоги
R3	Проблеми зі здоров’ям співробітника
R4	Раптова поломка техніки
R5	Нестабільний Інтернет
R6	Затримка у виконанні
R7	Недостатня тестування та відлагодження
R8	Непередбачені ризики безпеки
R9	Недостатня впровадження бекапів та захисту даних
R10	Відсутність стратегії впровадження змін

Таблиця Б.4 – Результати визначення ймовірності, впливу та рангу ризиків проєкту

№ ризику	Назва (опис) ризику	Ймовірність (0,1-0,9)	Вплив (0,05-0,8)	Ранг
R1	Масові відключення електроживлення	0,7	0,2	0,14
R2	Повітряні тривоги	0,6	0,2	0,12
R3	Проблеми зі здоров’ям співробітника	0,1	0,3	0,03
R4	Раптова поломка техніки	0,1	0,4	0,04
R5	Нестабільний Інтернет	0,5	0,2	0,1
R6	Затримка у виконанні	0,2	0,4	0,08
R7	Недостатня тестування та відлагодження	0,3	0,3	0,09
R8	Непередбачені ризики безпеки	0,2	0,5	0,1
R9	Недостатня впровадження бекапів та захисту даних	0,2	0,3	0,06
R10	Відсутність стратегії впровадження змін	0,3	0,4	0,12

Таблиця Б.5 – Шкала оцінювання ризиків за ймовірністю виникнення та величиною впливу

Оцінка	**Ймовірність виникнення**	**Вплив ризику**	**Тип ризику**
1	Низька	Низький	Прийнятні
2	Середня	Середній	Виправдані
3	Висока	Високий	Недопустимі

Для того, щоб знизити негативний вплив ризиків на проєкт треба виконати планування реагування на них. До нього входить оцінка наслідків впливу на проєкт і розробка відповідних заходів. Аналіз виконується за показниками, які описані в таблиці Б.4. У результаті планування заходів реагування на ризики проєкту було отримано матрицю ймовірності виникнення та впливу ризиків (табл. Б.6). Зеленим кольором на матриці позначають прийнятні ризики, жовтим – виправдані, а червоним – недопустимі.

Таблиця Б.6 – Матриця ймовірності та впливу згідно проєкту

Ймовірність ризику (Й)	Вплив загрози (ризику)				
	Дуже малий	Малий	Середній	Великий	Дуже великий
	0,2	0,3	0,4	0,5	0,7
0,7	0,14(R1)	0,21	0,28	0,35	0,49
0,6	0,12(R2)	0,18	0,24	0,3	0,42
0,5	0,1(R5)	0,15	0,2	0,25	0,35
0,3	0,06	0,09 (R7)	0,12 (R10)	0,15	0,21
0,2	0,04	0,06(R9)	0,08(R6)	0,1(R8)	0,14
0,1	0,02	0,03 (R3)	0,04 (R4)	0,05	0,07

Класифікація ризиків проєкту за рівнем, відповідно до отриманого значення індексу, представлена у таблиці Б.7.

Таблиця Б.7 – Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризики, які входять (номера)
1	Прийняті	$0,005 \leq R \leq 0,05$	R3, R4
2	Виправдані	$0,05 < R \leq 0,14$	R1, R2, R5, R6, R7, R8, R9, R10
3	Недопустимі	$0,14 < R \leq 0,72$	

У таблиці Б.8 описано ризики та стратегії реагування на кожен із них.

Таблиця Б.8 – Ризики та стратегії реагування на них

ID ризику	Статус ризику	Опис ризику	Ймовір-ність вини-кнення	Вплив ризику	Ранг ризику (кольором зазначити категорію ризику)	Тип стратегії реагування (який може бути застосовано до ризику)	План А (заходи запобігання виникненню ризику)	План Б (заходи усунення наслідків ризику)
R1	Виправданий	Масові відключення електроживлення	0,7	0,3	0,14	Ухилення	Використання генератора електроструму	Швидкий перехід на резервне джерело живлення.

Продовження таблиці Б.8

ID ризику	Статус ризику	Опис ризику	Ймовір-ність вини-кнення	Вплив ризику	Ранг ризику (кольором зазначити категорію ризику)	Тип стратегії реагування (який може бути застосовано до ризику)	План А (заходи запобігання виникненню ризику)	План Б (заходи усунення наслідків ризику)
R2	Виправданий	Повітряні тривоги	0,6	0,2	0,12	Зменшення	Проведення тренувань з персоналом щодо реагування на тривоги.	Швидка евакуація персоналу

Продовження таблиці Б.8

ID ризику	Статус ризику	Опис ризику	Ймовір-ність вини-кнення	Вплив ризику	Ранг ризику (кольором зазначити категорію ризику)	Тип стратегії реагування (який може бути застосовано до ризику)	План А (заходи запобігання виникненню ризику)	План Б (заходи усунення наслідків ризику)
R3	Прийнятний	Проблеми зі здоров'ям співробітника	0,1	0,3	0,03	Зменшення	Організація регулярних медичних оглядів	Надання медичної допомоги

Продовження таблиці Б.8

ID ризику	Статус ризику	Опис ризику	Ймовір-ність вини-кнення	Вплив ризику	Ранг ризику (кольором зазначити категорію ризику)	Тип стратегії реагування (який може бути застосовано до ризику)	План А (заходи запобігання виникненню ризику)	План Б (заходи усунення наслідків ризику)
R4	Прийнятний	Раптова поломка техніки	0,1	0,4	0,04	Зменшення	Регулярне технічне обслуговування	Перехід на резервне обладнання або швидкий ремонт

Продовження таблиці Б.8

ID ризику	Статус ризику	Опис ризику	Ймовір-ність вини-кнення	Вплив ризику	Ранг ризику (кольором зазначити категорію ризику)	Тип стратегії реагування (який може бути застосовано до ризику)	План А (заходи запобігання виникненню ризику)	План Б (заходи усунення наслідків ризику)
R5	Виправданий	Нестабільний Інтернет	0,5	0,2	0,1	Зменшення	Використання декількох постачальників Інтернету	Швидка зміна джерела Інтернету
R6	Виправданий	Затримка у виконанні	0,2	0,4	0,08	Ухилення	Регулярний моніторинг прогресу	Перепланування завдань

Продовження таблиці Б.8

ID ризику	Статус ризику	Опис ризику	Ймовір-ність вини-кнення	Вплив ризику	Ранг ризику (кольором зазначити категорію ризику)	Тип стратегії реагування (який може бути застосовано до ризику)	План А (заходи запобігання виникненню ризику)	План Б (заходи усунення наслідків ризику)
R7	Виправданий	Недостатня тестування та відлагодження	0,3	0,3	0,09	Ухилення	Ретельне планування тестування	Швидке виділення та використання ресурсів для виправлення помилок

Продовження таблиці Б.8

ID ризику	Статус ризику	Опис ризику	Ймовір-ність вини-кнення	Вплив ризику	Ранг ризику (кольором зазначити категорію ризику)	Тип стратегії реагування (який може бути застосовано до ризику)	План А (заходи запобігання виникненню ризику)	План Б (заходи усунення наслідків ризику)
R8	Виправ даний	Непередб ачені ризики безпеки	0,2	0,5	0,1	Зменшення	Постійне оновлення заходів безпеки	Негайна відповідь на виявлення уразливостей шляхом їхнього усунення або обмеження можливих наслідків.

Продовження таблиці Б.8

ID ризику	Статус ризику	Опис ризику	Ймовір-ність вини-кнення	Вплив ризику	Ранг ризику (кольором зазначити категорію ризику)	Тип стратегії реагування (який може бути застосовано до ризику)	План А (заходи запобігання виникненню ризику)	План Б (заходи усунення наслідків ризику)
R9	Виправ даний	Недостат ня впровадж ення бекапів та захисту даних	0,2	0,3	0,06	Зменшення	Автоматизоване регулярне створення бекапів	Відновлення даних з резервних копій у разі втрати чи пошкодження основних даних

Продовження таблиці Б.8

ID ризику	Статус ризику	Опис ризику	Ймовір-ність вини-кнення	Вплив ризику	Ранг ризику (кольором зазначити категорію ризику)	Тип стратегії реагування (який може бути застосовано до ризику)	План А (заходи запобігання виникненню ризику)	План Б (заходи усунення наслідків ризику)
R10	Виправданий	Відсутність стратегії впровадження змін	0,3	0,4	0,12	Ухилення	Розроблення стратегії впровадження змін	Перегляд стратегії впровадження змін

ДОДАТОК В (Лістинг програмного коду основних модулів вебдодатку)

Home.jsx:

```
import { useEffect, useState } from 'react';
// import { useSelector } from "react-redux";
import useFetch from '../../../hooks/useFetch';
import SmallMediaSlider from '../../ui/SmallMediaSlider/SmallMediaSlider';
import HeroSlider from './HeroSlider/HeroSlider';

function Home() {

  // Movies

  // const { data: popularData, loading: loadingData } = useFetch("/movie/popular");
  const { data: popularData, loading: loadingData } = useFetch("/trending/movie/day");
  const [popularMovies, setPopularMovies] = useState([]);

  useEffect(() => {
    popularData && setPopularMovies(() => popularData.results);
  }, [popularData]);

  const { data: upcomingData, loading: upcomingLoading } = useFetch("/movie/upcoming");
  const [upcomingMovies, setUpcomingMovies] = useState([]);
```

```
useEffect(() => {
   upcomingData && setUpcomingMovies(() => upcomingData.results);
}, [upcomingData]);

const { data: topRatedData, loading: topRatedLoading } = useFetch("/movie/top_rated");
const [topRatedMovies, setTopRatedMovies] = useState([]);

useEffect(() => {
   topRatedData && setTopRatedMovies(() => topRatedData.results);
}, [topRatedData]);



const { data: nowPlayingData, loading: nowPlayingLoading } = useFetch("/movie/now_playing");
const [nowPlayingMovies, setNowPlayingMovies] = useState([]);

useEffect(() => {
   nowPlayingData && setNowPlayingMovies(() => nowPlayingData.results);
}, [nowPlayingData]);

// TV Series
const { data: airingTodayDataSeries, loading: airingTodayLoadingSeries } = useFetch("/tv/airing_today");
const [airingTodaySeries, setAiringTodaySeries] = useState([]);

useEffect(() => {
```

```
    airingTodayDataSeries && setAiringTodaySeries(() => airingTodayDataSeries.results);
  }, [airingTodayDataSeries]);

 return (
  <>
    <HeroSlider data={popularMovies} isLoading={loadingData}/>
    <SmallMediaSlider mediaType='movie' title='Незабаром на екрані' mediaData={upcomingMovies} isLoading={upcomingLoading}/>
    <SmallMediaSlider mediaType='movie' title='Популярні фільми' mediaData={popularMovies} isLoading={loadingData}/>
    <SmallMediaSlider mediaType='movie' title='Топ рейтингів' mediaData={topRatedMovies} isLoading={topRatedLoading}/>
    <SmallMediaSlider mediaType='movie' title='Зараз на екранах' mediaData={nowPlayingMovies} isLoading={nowPlayingLoading}/>
    <SmallMediaSlider mediaType='tv' title='Сьогодні в ефірі' mediaData={airingTodaySeries} isLoading={airingTodayLoadingSeries}/>
  </>
 );
}

export default Home;
```

SingleMedia.jsx:

```
import React, { useEffect, useState } from 'react';
import { useParams } from 'react-router-dom';
import useFetch from '../../../hooks/useFetch';
import MediaPlayer from '../../ui/MediaPlayer/MediaPlayer';
import SmallMediaSlider from '../../ui/SmallMediaSlider/SmallMediaSlider';
import styles from './SingleMedia.module.scss';
import SingleMediaCollection from './SingleMediaCollection/SingleMediaCollection';
```

```
import SingleMediaInfo from './SingleMediaInfo/SingleMediaInfo';
import SingleMediaPoster from './SingleMediaPoster/SingleMediaPoster';

export default function SingleMedia() {
  const { type, id } = useParams();
  const { data: mediaData, loading: mediaLoading } = useFetch(`/${type}/${id}`);
  const [media, setMedia] = useState([]);
  const [collectionID, setCollectionID] = useState('');

  useEffect(() => {
    mediaData ? setMedia(mediaData) : setMedia([]);
    mediaData && mediaData.belongs_to_collection ? setCollectionID(mediaData.belongs_to_collection.id) : setCollectionID('');
  }, [mediaData]);

  const { data: movieData, loading: movieLoading } = useFetch(`/${type}/${id}/videos`);
  const [movie, setMovie] = useState([]);

  useEffect(() => {
    movieData && movieData?.results?.length ? setMovie(movieData.results) : setMovie([]);
  }, [movieData]);

  const { data: relatedData, loading: relatedLoading } = useFetch(`/${type}/${id}/similar`);
  const [relatedMovies, setRelatedMovies] = useState([]);

  useEffect(() => {
```

```
  relatedData ? setRelatedMovies(relatedData.results) : setRelatedMovies([]);
 }, [relatedData]);

 const { data: recommendationsData, loading: recommendationsLoading } = useFetch(`/${type}/${id}/recommendations`);
 const [recommendationsMovies, setRecommendationsMovies] = useState([]);

 useEffect(() => {
  recommendationsData ? setRecommendationsMovies(recommendationsData.results) : setRecommendationsMovies([]);
 }, [recommendationsData]);


 return (
  <>
   <div className={styles.singleMedia}>
    <SingleMediaPoster id={id} type={type} imageLink={media.poster_path} alt={media.title}/>
    <SingleMediaInfo data={media} isLoading={mediaLoading}/>
   </div>

   <MediaPlayer movie={movie} rate={media}/>

   {collectionID && <SingleMediaCollection collectionID={collectionID}/>}

   {recommendationsMovies && <SmallMediaSlider mediaType={type} mediaData={recommendationsMovies} title='Вам також може сподобатись' isLoading={recommendationsLoading}/>}
   {relatedMovies && <SmallMediaSlider mediaType={type} mediaData={relatedMovies} title={`Схоже на «${media.title}»`} isLoading={relatedLoading}/>}
```

```
    </>
  )
}
```

MediaListPage.jsx:

```
import { Grid, Pagination, Skeleton } from '@mui/material';
import { useEffect, useState } from 'react';
import { useLocation, useNavigate } from 'react-router-dom';
import Select from "react-select";
import useFetch from '../../../hooks/useFetch';
import { fetchDataFromApi } from "../../../utils/api";
import MediaList from '../../ui/MediaList/MediaList';
import styles from './MediaListPage.module.scss';

const today = new Date();
const year = today.getFullYear();
const month = String(today.getMonth() + 1).padStart(2, '0');
const day = String(today.getDate()).padStart(2, '0');
const formattedDate = `${year}-${month}-${day}`;

const sortbyData = [
  { value: "popularity.desc", label: "Популярні" },
  { value: "primary_release_date.desc", label: "Найновіші" },
  { value: "revenue.desc", label: "Найбільш касові" },
];
```

```
const _MIN_YEAR = 1980;
const yearFilter = [
  { value: 'all', label: 'Всі роки' },
  ...Array.from({ length: year - _MIN_YEAR + 1 }, (_, index) => ({
    value: year - index,
    label: year - index
  }))
];


const customStyles = {
  control: (base, state) => ({
    ...base,
    borderColor: 'var(--accentColor)',
    background: 'var(--accentColor)',
    color: 'red'
  }),
};

const SelectComponent = ({ defaultValue, name, value, options, onChange, isMulti, getOptionLabel, getOptionValue, placeholder }) => (
  <Select
    classNames={{
      control: () => styles.select,
      placeholder: () => styles.selectPlaceholder,
```

```
      option: () => styles.selectOption,
      multiValueLabel: () => styles.selectInput,
      multiValue: () => styles.selectMultiValue,
      multiValueRemove: () => styles.selectMultiValueRemove,
      singleValue: () => styles.selectSingleValue,
      menu: () => styles.menu,
    }}
    styles={customStyles}
    isMulti={isMulti}
    name={name}
    value={value}
    options={options}
    onChange={onChange}
    getOptionLabel={getOptionLabel}
    getOptionValue={getOptionValue}
    placeholder={placeholder}
    defaultValue={defaultValue}
  />
);

const MediaListPage = ({ type }) => {
  const navigate = useNavigate();
  const { search } = useLocation();
  const params = new URLSearchParams(search);
```

```
const [filters, setFilters] = useState(() => ({
  'vote_count.gte': 50,
  'primary_release_date.lte': formattedDate,
  'primary_release_year': parseInt(params.get('primary_release_year')) || null,
  'with_genres': params.get('with_genres'),
  'with_origin_country': params.get('with_origin_country'),
  'sort_by': params.get('sort_by'),
  'page': parseInt(params.get('page')) || 1
}));

const [data, setData] = useState(null);
const [genres, setGenres] = useState(null);
const [countries, setCountries] = useState(null);
const [page, setPage] = useState(filters.page);
const [selectedGenres, setSelectedGenres] = useState(filters.with_genres ? filters.with_genres.split(",").map(id => parseInt(id)) : []);
const [selectedCountries, setSelectedCountries] = useState(filters.with_origin_country ? filters.with_origin_country.split(",") : []);
const [selectedSortBy, setSelectedSortBy] = useState(sortbyData.find(option => option.value === filters.sort_by));
const [selectedYear, setSelectedYear] = useState(filters.primary_release_year);

const { data: genresData } = useFetch(`/genre/${type}/list`);
useEffect(() => {
  if (genresData && genresData.genres) {
    setGenres(genresData.genres);
  }
}, [genresData]);
```

```
const { data: countriesData } = useFetch(`/configuration/countries`);
useEffect(() => {
  if (countriesData && countriesData) {
    setCountries(countriesData);
  }
}, [countriesData]);

useEffect(() => {
  fetchData();
}, [filters]);

const fetchData = () => {
  fetchDataFromApi(`/discover/${type}`, filters)
    .then((res) => {
      setData(res);
    })
    .catch((err) => {
      console.error(err);
    });
};

const handleSelectChange = (value, action) => {
  const updatedFilters = { ...filters };
  const params = new URLSearchParams(search);
```

```
switch (action.name) {
  case "sort_by":
  case "primary_release_year":
    params.set(action.name, value?.value || '');
    if (action.name === "sort_by") {
      setSelectedSortBy(value);
    } else {
      setSelectedYear(value?.value || null);
    }
    updatedFilters[action.name] = value?.value || null;
    break;
  case "with_genres":
  case "with_origin_country":
    const selectedItems = value.map(item => item.id || item.iso_3166_1).join(",");
    setSelectedGenres(action.name === "with_genres" ? value.map(item => item.id) : selectedGenres);
    setSelectedCountries(action.name === "with_origin_country" ? value.map(item => item.iso_3166_1) : selectedCountries);
    updatedFilters[action.name] = selectedItems;
    params.set(action.name, selectedItems);
    break;
  default:
    break;
}

params.set('page', 1);
```

```
  updatedFilters.page = 1;
  setFilters(updatedFilters);
  setPage(1);
  navigate(`?${params.toString()}`);
};

const handlePageChange = (event, page) => {
  const params = new URLSearchParams(search);
  params.set('page', page);
  navigate(`?${params.toString()}`);
  setPage(page);
  setFilters(prevFilters => ({...prevFilters, 'page': page}));
};

return (
  <div>
    <div className={styles.sectionHead}>
      <h2 className={styles.sectionTitle}>Фільми</h2>
      <div className={styles.selectsWrapper}>
        <SelectComponent
          name="with_genres"
          value={genres?.filter(genre => selectedGenres.includes(genre.id))}
          options={genres}
          getOptionLabel={(option) => option.name}
          getOptionValue={(option) => option.id}
```

```
  onChange={handleSelectChange}
  isMulti
  placeholder="Оберіть жанри"
/>

<SelectComponent
  name="sort_by"
  value={selectedSortBy}
  options={sortbyData}
  onChange={handleSelectChange}
  placeholder="Сортування за"
  defaultValue={sortbyData[0]}
/>

<SelectComponent
  name="primary_release_year"
  value={yearFilter.find(option => option.value === selectedYear)}
  options={yearFilter}
  onChange={handleSelectChange}
  placeholder="Оберіть рік"
/>

<SelectComponent
  name="with_origin_country"
  value={countries?.filter(country => selectedCountries.includes(country.iso_3166_1))}
```

```
      options={countries}
      onChange={handleSelectChange}
      isMulti
      getOptionLabel={(option) => `${option.native_name} (${option.iso_3166_1})`}
      getOptionValue={(option) => option.iso_3166_1}
      placeholder="Оберіть країни"
    />
  </div>
</div>

{!data ?
  <Grid style={{marginTop: '5rem'}} container columns={10} columnSpacing='1.5rem' rowSpacing={5}>
    {[...Array(20)].map((_, index) => (
      <Grid key={index} item xs={2} style={{ margin: 0}}>
        <Skeleton width='100%' variant="rectangular" height='22rem'/>
        <Skeleton width='100%' variant="text" sx={{ fontSize: '2rem' }}/>
        <Skeleton width='100%' variant="text" sx={{ fontSize: '1rem' }}/>
      </Grid>
    ))}
  </Grid>
  :
  <MediaList
    data={data?.results}
    isCollapsed={false}
  />
```

```
    }

    {
      data?.total_pages > 1 &&
      <Pagination
        size="large"
        color="primary"
        onChange={handlePageChange}
        style={{
          marginTop: '6rem',
          display: 'flex',
          justifyContent: 'center',
          color: 'red'
        }}
        page={page}
        count={data?.total_pages <= 500 ? data?.total_pages : 500}
        showFirstButton
        showLastButton
      />
    }

  </div>
 );
};
```

```
export default MediaListPage;
```

WatchTogether.jsx:

```
import { initializeApp } from "firebase/app";
import { collection, doc, getDocs, getFirestore, query, setDoc } from "firebase/firestore";
import { useEffect, useState } from 'react';
import { useParams } from 'react-router-dom';
import useFetch from '../../../hooks/useFetch';
import MediaPlayer from '../../ui/MediaPlayer/MediaPlayer';
import styles from './WatchTogether.module.scss';

const app = initializeApp({
  apiKey: "AIzaSyDZccz9xtY0KH8eASpX4DQDMS_CIVsXBHI",
  authDomain: "maincharacter-e9dd7.firebaseapp.com",
  projectId: "maincharacter-e9dd7",
  storageBucket: "maincharacter-e9dd7.appspot.com",
  messagingSenderId: "725562149817",
  appId: "1:725562149817:web:1ffc31206f0e53391f8468"
});



const WatchTogether = ({ type }) => {

  const { mediaID, roomID } = useParams();
```

```
const citiesRef = collection(getFirestore(app), `room_${roomID}`);
const [messages, setMessages] = useState([]);
const [loadingMessages, setLoadingMessages] = useState(true);

useEffect(() => {
  const fetchData = async () => {
    try {
      const querySnapshot = await getDocs(citiesRef);
      const data = querySnapshot.docs.map(doc => ({
        id: doc.id,
        ...doc.data()
      }));
      setMessages(data);
      setLoadingMessages(false);
      console.log(data);
    } catch (error) {
      console.error('Error fetching messages:', error);
    }
  };

  fetchData();
}, []);

// setDoc(doc(citiesRef), {
//   name: "Name",
```

```
//   text: 'test'
// });
const { data: movieData, loading: movieLoading } = useFetch(`/movie/${mediaID}/videos`);
const [movie, setMovie] = useState([]);

useEffect(() => {
  movieData && movieData?.results?.length ? setMovie(movieData.results) : setMovie([]);
}, [movieData]);

// const { data: data, loading: loadingData } = useFetch(`/search/movie?query=${query}&include_adult=false`);
// const [searchResults, setSearchResults] = useState([]);

// useEffect(() => {
//   data && setSearchResults(() => data?.results);
// }, [data]);



return (
  <div>
    {movie[0]?.name &&
      <>
        <div className={styles.sectionHead}>
          <h2 className={styles.sectionTitle}>Код кімнати - {roomID}</h2>
        </div>
        <p style={{marginTop: '1rem'}}>Скопіюй та відрав цей код другу для спільного перегляду.</p>
```

```
  </>
}

<div className={styles.contentWrapper}>
  <div className={styles.mediaWrapper}>

    <h2>{movie[0]?.name}</h2>
    <MediaPlayer movie={movie} isHideHeader={true}/>

  </div>

  {movie[0]?.name &&
    <>
      <div className={styles.chatWrapper}>
        <h2>Чат</h2>

        <div className={styles.chat}>
          <div className={styles.chatMessages}>
            {messages.map(item => (
              <div key={item.id} className={`${styles.messageItem} ${styles.myMessageItem}`}>
                <span>{item.name}</span>
                <p>{item.text}</p>
              </div>
            ))}
          </div>
```

```
        <div className={styles.chatFooter}>
          <textarea></textarea>

          <button className={styles.chatButton}><i className="fa-solid fa-arrow-up"></i></button>
        </div>
       </div>
     </div>
   </>
  }
</div>

{/* {!data ? */}
  {/* <Grid style={{marginTop: '5rem'}} container columns={10} columnSpacing='1.5rem' rowSpacing={5}>
    {[...Array(20)].map((_, index) => (
      <Grid key={index} item xs={2} style={{ margin: 0}}>
        <Skeleton width='100%' variant="rectangular" height='22rem'/>
        <Skeleton width='100%' variant="text" sx={{ fontSize: '2rem' }}/>
        <Skeleton width='100%' variant="text" sx={{ fontSize: '1rem' }}/>
      </Grid>
    ))}
  </Grid> */}
  {/* :
  <MediaList
    data={searchResults}
```

```
        isCollapsed={false}
       />
     } */}

   </div>
  );
};

export default WatchTogether;
```

FavoriteListPage.jsx:

```
import { Grid, Skeleton } from '@mui/material';
import { useEffect, useState } from 'react';
import { useParams } from 'react-router-dom';
import useFetch from '../../../hooks/useFetch';
import MediaList from '../../ui/MediaList/MediaList';
import styles from './FavoriteListPage.module.scss';



const FavoriteListPage = ({ type }) => {
  const { query } = useParams();

  const { data: data, loading: loadingData } = useFetch(`/search/movie?query=${query}&include_adult=false`);
  const [searchResults, setSearchResults] = useState([]);
```

```
useEffect(() => {
  data && setSearchResults(() => data?.results);
}, [data]);



return (
  <div>
    <div className={styles.sectionHead}>
      <h2 className={styles.sectionTitle}>Улюблені фільми</h2>

    </div>

    {/* {!data ? */}
      <Grid style={{marginTop: '5rem'}} container columns={10} columnSpacing='1.5rem' rowSpacing={5}>
        {[...Array(20)].map((_, index) => (
          <Grid key={index} item xs={2} style={{ margin: 0}}>
            <Skeleton width='100%' variant="rectangular" height='22rem'/>
            <Skeleton width='100%' variant="text" sx={{ fontSize: '2rem' }}/>
            <Skeleton width='100%' variant="text" sx={{ fontSize: '1rem' }}/>
          </Grid>
        ))}
      </Grid>
      :
      <MediaList
```

```
        data={searchResults}
        isCollapsed={false}
       />
     }

   </div>
  );
};

export default FavoriteListPage;
```

SearchPage.jsx:

```
import { Grid, Skeleton } from '@mui/material';
import { useEffect, useState } from 'react';
import { useParams } from 'react-router-dom';
import useFetch from '../../../hooks/useFetch';
import MediaList from '../../ui/MediaList/MediaList';
import styles from './SearchPage.module.scss';



const SearchPage = ({ type }) => {
  const { query } = useParams();

  const { data: data, loading: loadingData } = useFetch(`/search/movie?query=${query}&include_adult=false`);
```

```
const [searchResults, setSearchResults] = useState([]);

useEffect(() => {
  data && setSearchResults(() => data?.results);
}, [data]);



return (
  <div>
    <div className={styles.sectionHead}>
      <h2 className={styles.sectionTitle}>Результати за запитом "{query}"</h2>

    </div>

    {!data ?
      <Grid style={{marginTop: '5rem'}} container columns={10} columnSpacing='1.5rem' rowSpacing={5}>
        {[...Array(20)].map((_, index) => (
          <Grid key={index} item xs={2} style={{ margin: 0}}>
            <Skeleton width='100%' variant="rectangular" height='22rem'/>
            <Skeleton width='100%' variant="text" sx={{ fontSize: '2rem' }}/>
            <Skeleton width='100%' variant="text" sx={{ fontSize: '1rem' }}/>
          </Grid>
        ))}
      </Grid>
      :
```

```
        <MediaList
          data={searchResults}
          isCollapsed={false}
        />
      }

    </div>
  );
};

export default SearchPage;
```