

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
ВЕБЗАСТОСУНКУ ДЛЯ СТРИМІНГУ ВІДЕО
RESEARCH AND SOFTWARE IMPLEMENTATION OF A VIDEO
STREAMING WEB APPLICATION

Виконав: студент(ка) 2 курсу, групи КНТ-212м

Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

СТАШУК Д.А

(ПРІЗВИЩЕ та ініціали)

Керівник ФЕДОРОНЧАК Т.В.

(ПРІЗВИЩЕ та ініціали)

Рецензент ТЕРЕЩЕНКО Е.В.

(ПРІЗВИЩЕ та ініціали)

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

Факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки і технології
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій Субботін
“ ” 20__ року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

СТАШУКА Дениса Андрійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація вебзастосунку для стрімінгу відео.

Research and Software Implementation of a Video Streaming Web Application

керівник проєкту (роботи) к.т.н., доцент, ФЕДОРОНЧАК Тетяна Василівна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “14” листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Розробка програмного забезпечення. 5. Експлуатація, тестування та експериментальне дослідження програми.

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

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	ФЕДОРОНЧАК Т.В., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання “ 04 ” вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

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

Студент(ка)

Денис СТАШУК

(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

Тетяна ФЕДОРОНЧАК

(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
98 с., 57 рис., 3 табл., 3 дод., 22 джерела.

СТРИМІНГ ВІДЕО, ВЕБЗАСТОСУНОК, TAILWIND CSS, MOBA
TYPESCRIPT, БІБЛІОТЕКА REACT, ФРЕЙМВОРК NEXT.JS, PRISMA,
NEXTAUTH.JS.

Об'єкт дослідження – програмні засоби для перегляду відео.

Предмет дослідження – вебзастосунок, що надає користувачам
можливість переглядати відео в онлайні.

Мета роботи – програмна реалізація вебзастосунку для стримінгу відео.

Матеріали, методи та технічні засоби: інтегроване середовище
розробки WebStorm, основні інструменти для веброзробки Tailwind CSS, мова
програмування TypeScript, фреймворк Next.js, бібліотека React, система
авторизації NextAuth.js, система Prisma, система керування базами даних
MongoDB.

Результати. Розроблено програмний вебзастосунок для стримінгу
відео.

Висновки. Результати роботи повністю відповідають поставленій меті
та визначеному технічному завданню до проєкту. Проведено аналіз та
порівняння з існуючими подібними застосунками. Розроблено програмний
продукт, який призначений для стримінгу відео.

Галузь використання – проєкт може бути використаний для перегляду
відео.

ABSTRACT

Explanatory note to the diploma quality work of the master: 98 pp., 57 figures, 3 tables, 3 appendices, 22 sources.

VIDEO STREAMING, WEB APPLICATION, TAILWIND CSS, TYPESCRIPT LANGUAGE, REACT LIBRARY, NEXT.JS FRAMEWORK, PRISMA, NEXTAUTH.JS.

The object of the research is video viewing software.

The subject of the research is a web application that provides users with the opportunity to watch videos online.

The purpose of the work is software implementation of a web application for video streaming.

Materials, methods and technical means: WebStorm integrated development environment, main tools for web development Tailwind CSS, TypeScript programming language, Next.js framework, React library, NextAuth.js authorization system, Prisma system, MongoDB database management system.

The results. A software web application for video streaming has been developed.

Conclusions. The results of the work fully correspond to the set goal and the defined technical task for the project. An analysis and comparison with existing similar applications was carried out. A software product designed for video streaming has been developed.

Field of use – the project can be used for watching videos.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз предметної області.....	10
1.1 Опис предметної області	10
1.2 Аналіз аналогів.....	11
1.2.1 «Netflix»	12
1.2.2 «Amazon Prime Video»	13
1.2.3 «Apple TV».....	14
1.2.4 Порівняльна таблиця існуючих програмних продуктів.....	15
1.3 Постановка завдання роботи.....	15
1.4 Висновки до розділу 1	16
2 Матеріали і методи.....	17
2.1 Задача сервісу для стримінгу відео	17
2.2 Алгоритм пошуку.....	18
2.3 Висновки до розділу 2	19
3 Опис програми.....	20
3.1 Основні засоби розробки.....	20
3.1.1 Tailwind CSS	20
3.1.2 TypeScript.....	20
3.1.3 React.js	21
3.1.4 Next.js	23
3.1.5 NextAuth.js	25
3.1.6 Prisma.....	25
3.1.7 MongoDB.....	26
3.1.8 Node.js.....	29
3.1.9 The Movie Database API.....	31
3.2 Вибір середовища розробки.....	31
3.3 Архітектура вебзастосунку	33

3.4 Вимоги до програмного забезпечення	34
3.5 Аналіз вимог до розробки дизайну програми	36
3.6 Висновки до розділу 3	36
4 Розробка програмного забезпечення.....	38
4.1 Розробка бази даних.....	38
4.2 Способи реєстрації та авторизації.....	40
4.2.1 Google OAuth	41
4.2.2 GitHub OAuth.....	42
4.2.3 Електронна адреса та пароль	44
4.3 Структурна схема проєкту	44
4.4 Опис реалізованих функцій.....	48
4.5 Висновки до розділу 4	50
5 Експлуатація, тестування та експериментальне дослідження програми	51
5.1 Призначення програми	51
5.2 Виконання програми.....	51
5.3 Висновки до розділу 5	62
Висновки	64
Перелік джерел посилання	66
Додаток А Технічне завдання	68
Додаток Б Текст програми	72
Додаток В Слайди презентації.....	88

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
CSS	– Cascading Style Sheets;
DB	– Database;
DOM	– Document Object Model;
HTML	– HyperText Markup Language;
HTTP	– HyperText Transfer Protocol;
ID	– Identifier;
IDE	– Integrated Development Environment;
IP	– Internet Protocol;
JS	– JavaScript;
JSCS	– JavaScript Code Style;
JSON	– JavaScript Object Notation;
NoSQL	– Not Only SQL;
SEO	– Searching Engine Optimization;
SQL	– Structured Query Language;
SSG	– Static-Site Generating;
SSR	– Server-Side Rendering;
SVN	– Subversion;
TCP	– Transmission Control Protocol;
UDP	– User Datagram Protocol;
URL	– Uniform Resource Locator;
VCS	– Version Control System;
ОС	– операційна система;
ПЗ	– програмне забезпечення.

ВСТУП

Способи перегляду кінокартин за свою історію дуже змінилися. Історія кінематографу бере свій початок 28 грудня 1895 року, коли в Парижі на бульварі Капуцинок в одному із залів «Гранд кафе» відбувся перший сеанс кінопоказу. Після цього по всьому світі почали створювати кінокомпанії та кіностудії. І вже на сьогодні людство має можливість переглядати фільми не виходячи з дому, або, навпаки, подорожуючи.

Створення вебзастосунку для стримінгу відео має багато переваг, наприклад, портативність, доступність та легкість у використанні. Наприклад, такий вебзастосунок має всі необхідні функції, які потрібні користувачу та має вже готову базу кінострічок.

Актуальність та проблематика роботи полягає в тому, що з часом медіа стало невід'ємною частиною життя кожного з нас. Тому запит суспільства на створення такого вебзастосунку для стримінгу відео є великим. Таким чином, переглядати фільми стало простіше та зручніше, маючи гаджет під рукою. Також це значно зменшить час та зусилля на пошуки цікавої інформації. Не так давно потрібно було шукати та купувати DVD-диски та мати необхідне обладнання для перегляду зміста носія.

Метою дипломної кваліфікаційної роботи є аналіз існуючих методів та програмна реалізація вебзастосунку для стримінгу відео.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Сервіс потокового відео – це онлайн-джерело розваг для телешоу, фільмів та інших потокових медіа.

Потокова передача дозволяє дивитися телепередачі та фільми або слухати подкасти онлайн без необхідності завантажувати файли, що займають місце на пристрої [1]. Вона працює шляхом розбиття контенту (наприклад, фільму) на дрібніші частини або пакети даних. Ці пакети даних відправляються до браузера, де відеоплеєр інтерпретує дані як фільм. Як тільки в браузері буде достатньо пакетів даних для початку, розпочнеться відтворення фільму.

Більшість потокових сайтів використовують стандартизовані рівні TCP/IP (протокол керування передачею/інтернет-протокол) замість UDP (протокол користувальницьких датаграм) для передачі контенту з сервера на пристрій. TCP/IP більш надійний доставки пакетів даних. Але потокові сайти, які використовують UDP, зазвичай забезпечують швидшу передачу. Netflix, Amazon Prime та Spotify використовують TCP, а YouTube використовує комбінацію TCP та UDP [2].

Послуги потокового відео зазвичай зосереджуються на створеному студією контенті, наприклад фільмах і телепередачах. Сервіси, які зосереджуються на контенті, створеному кінцевими користувачами, наприклад YouTube і Twitch, зазвичай класифікуються як соціальні медіа або потокові платформи.

Ці послуги є альтернативою кабельним і супутниковим послугам на вимогу, часто за нижчою ціною. Використання потокових послуг часто вимагає плати за перегляд або підписку. Інші служби підтримуються рекламою або працюють за моделлю freemium.

Серед популярних потокових відеосервісів сьогодні є:

- Netflix;

- Amazon Prime Video;
- Disney +;
- Apple TV +;
- YouTube Premium.

Залежно від розміру аудиторії, сервіси потокового відео розміщують вміст на власних серверах або на хмарних серверах. Для невеликих служб було б корисно використовувати хмарні сервери, оскільки їх можна легко масштабувати відповідно до потреб і мати сервери в різних географічних регіонах. Більші служби зазвичай володіють власними серверами, оскільки економія на масштабі робить це більш рентабельним.

Багато служб використовують мережу доставки контенту (CDN). CDN зменшує навантаження на їхні сервери та розміщує контент географічно ближче до кінцевого користувача.

Основні вимоги до вебзастосунку для стримінгу відео:

- реєстрація користувачів (головні варіанти реєстрації – через емейл та номер мобільного телефону, але можна додати швидку авторизацію через соціальні мережі Apple ID або Google ID та сервіси електронної пошти);
- домашня сторінка та система рекомендацій;
- відеоплеєр;
- метадані відео;
- профіль користувача;
- пошук (пошуковий рядок, категорії, можна додати голосовий пошук).

1.2 Аналіз аналогів

В наш час існує достатньо вебзастосунків для стримінгу відео. Вони мають лаконічний, інтуїтивно зрозумілий дизайн, який допомагає користувачам швидко знайти потрібний матеріал.

1.2.1 «Netflix»

Одним з аналогів розроблюваної програми є програма «Netflix.com» [3]. «Netflix.com» - це сервіс потокового передавання відео, що дає змогу дивитися різноманітні телесеріали, фільми, документалістику й інший контент на тисячах пристроїв із підключенням до Інтернету. Застосунок має зручний та зрозумілий інтерфейс. У ньому всього у кілька дій користувач має змогу швидко знайти потрібний ресурс. На рис. 1.1 наведено сторінку авторизації вебзастосунку «Netflix.com».

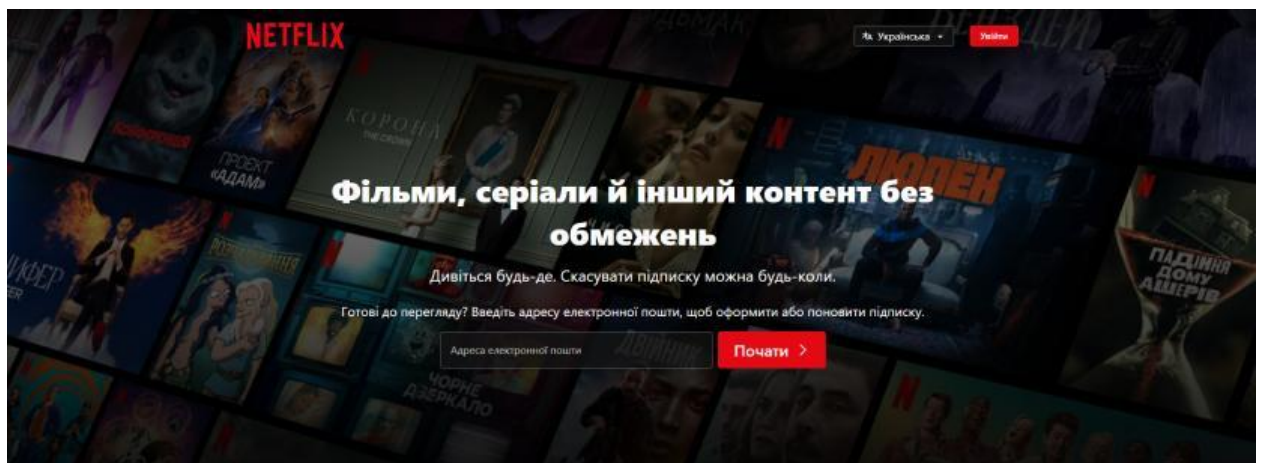


Рисунок 1.1 – Сторінка авторизації вебзастосунку «Netflix.com»

Переваги «Netflix.com»:

- адаптивність. Застосунок можна використовувати вдома, в дорозі, на смартфоні або персональному комп'ютері;
- зовнішня взаємодія. Вебзастосунок виконує всі необхідні функції. Можливість пошуку, перегляду та додавання фільмів до списку улюблених.

Даний застосунок володіє важливою перевагою – ним легко та просто користуватись. Також є можливість завантажити додаток для смартфона.

Головними недоліками цього ресурсу є платність та відсутність безкоштовного пробного періоду.

1.2.2 «Amazon Prime Video»

Наступним аналогом є «PrimeVideo.com» [4]. Amazon Prime Video — це потоковий сервіс з багатьма популярними фільмами та шоу, включаючи популярні оригінальні серіали. На рис. 1.2 наведено користувацький інтерфейс вебзастосунку «PrimeVideo.com».

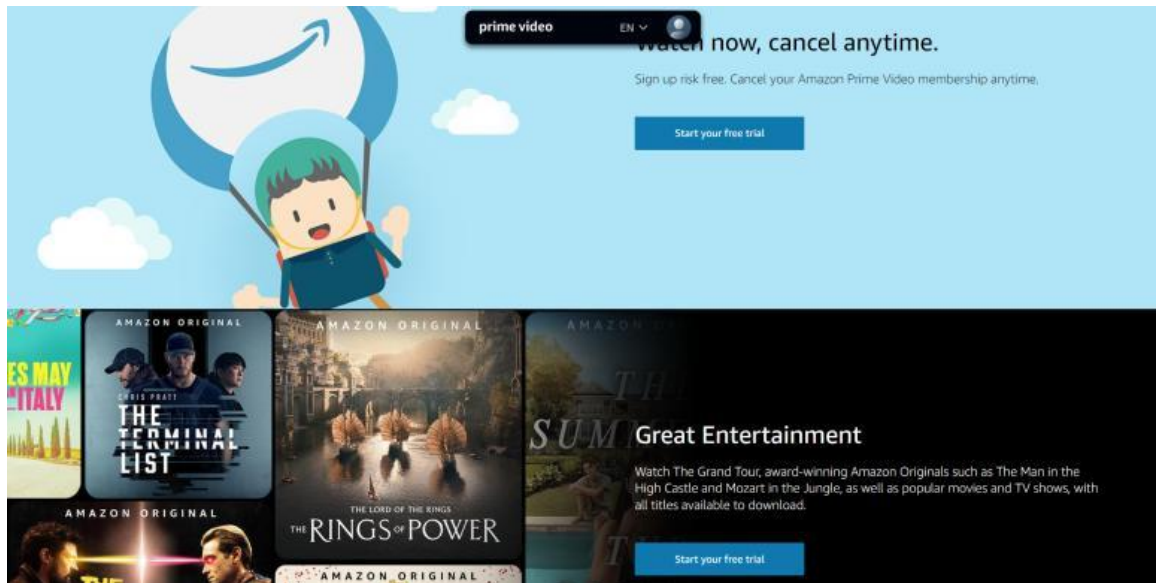


Рисунок 1.2 – Сторінка авторизації сайту «PrimeVideo.com»

Переваги «PrimeVideo.com»:

- функціональність. Ресурс відповідає всім вимогам щодо користування;
- наявність безкоштовного пробного періоду;
- можливість завантажити додаток для IOS та Android систем;
- наявність підписки Amazon Prime, яка включає в себе функціонал підписки Prime Video, але коштує значно дорожче.

Недоліки «PrimeVideo.com»:

- адаптивність. Застосунок має не досить зручну адаптивність, що може викликати деякі складнощі при користуванні на смартфоні;
- платна підписка.

1.2.3 «Apple TV»

Ще одним аналогом є застосунок «Apple TV» [5]. Досить популярний сервіс для перегляду фільмів та багато чого іншого. Включає в себе фільми та серіали Apple Originals. На рис. 1.3 наведено користувацький інтерфейс вебзастосунку «Apple TV».

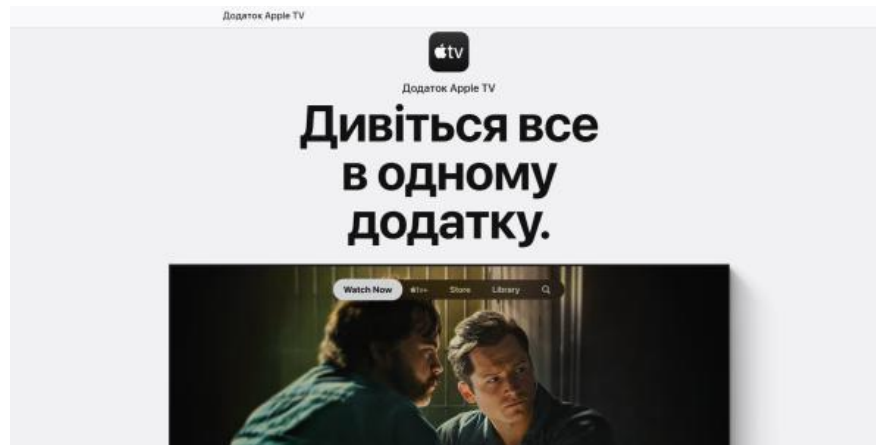


Рисунок 1.3 – Головна сторінка сайту «Apple TV»

Переваги «Apple TV»:

- адаптивність. Застосунок можна використовувати на будь-якому пристрої;
- зовнішня взаємодія. Вебзастосунок виконує всі необхідні функції;
- можливість безкоштовно отримати 3 місяці підписки при купівлі техніки Apple;
- можливість брати фільми в оренду;
- безкоштовний пробний період;
- доступність на всіх можливих платформах у тому числі на ігрових приставках.

Недоліки «Apple TV»:

- ціна значно вища, ніж у попередніх аналогах;
- не досить продуманий дизайн сайту (поєднання кольорів), що може збити користувача.

1.2.4 Порівняльна таблиця існуючих програмних продуктів

Вище було наведено найбільш популярні на сьогоднішній день сервіси для стримінгу відео. Але окрім основних переваг, усі вони мають і недоліки. Наприклад, всі застосунки є платними, хоча деякі з них надають безкоштовний пробний період.

Також існують характерні переваги для усіх вебзастосунків: вони мають велику бібліотеку фільмів та серіалів, включають в себе усі необхідні функції та ін.

В табл. 1.1 проведено порівняння вже існуючих рішень.

Таблиця 1.1 — Порівняння програмних продуктів

	«Netflix»	«Prime Video»	«Apple TV»
Виконання основного функціоналу (перегляд фільмів, список улюбленого)	+	+	+
Адаптивність	+	-	+
Зручний інтерфейс	+	-	-
Платність	+	+	+
Підтримка української мови	+	+	+
Швидкодія	+	+	+
Безкоштовний пробний період	-	+	+

1.3 Постановка завдання роботи

Оскільки завданням роботи є реалізація вебзастосунку для стримінгу відео, можна виділити такі завдання на виконання:

- проаналізувати інформаційне забезпечення;
- розробити функціональну структуру програми;

- проаналізувати логічну структуру програми;
- розробити базу даних;
- розробити інтерфейс;
- обрати мову програмування та середовище розробки;
- виконати програмну реалізацію;
- провести тестування та виправлення помилок.

1.4 Висновки до розділу 1

В даному розділі було досліджено предметну область. В ході дослідження було виявлено, що існує достатня кількість вебзастосунків для стримінгу відео. На базі цього аналізу було сформовано список переваг, які варто враховувати при створенні власного застосунку, а саме, виконання основних функцій, безкоштовність, зручний та простий інтерфейс, виконання основних функцій, швидкодія, адаптивність, кросплатформність. Також було визначено основні завдання розробки.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Задача сервісу для стримінгу відео

Для формулювання задачі сервісу для стримінгу відео спершу потрібно визначити, яким чином рекомендувати популярні фільми та здійснювати пошук кінострічки за назвою.

При розробці списків відео використовувалось два способи. Перший полягає в тому, щоб додати відео до бази даних вручну при розробці програмного забезпечення. Але через обмеження безкоштовної версії сервісу бази даних неможливо створити повноцінну бібліотеку кінокартин. Тому було задіяно ще один спосіб, а саме використання The Movie Database API. Даний метод дозволяє формувати списки телевізійних шоу, фільмів та популярних кінокартин. Ця система містить в собі велику бібліотеку з рейтингами, оцінками користувачів та ін. Для формування списків популярних відео система використовує атрибути. Для кожного типу відео атрибути можуть відрізнятися.

Атрибути фільмів:

- кількість переглядів за день;
- кількість користувачів, які додали його до списку улюблених фільмів за день;
- дата випуску;
- загальна кількість голосів;
- оцінка за попередні дні.

Атрибути телевізійних шоу:

- кількість голосів за день;
- кількість переглядів за день;
- кількість користувачів, які додали його до списку улюблених фільмів за день;
- кількість користувачів, які додали його до свого «списку спостереження» за день;

- дата виходу наступної/останньої серії;
- загальна кількість голосів;
- оцінка за попередні дні.

Для використання The Movie Database API необхідно спочатку зробити запит на ключ API. Це можна зробити на сторінці налаштувань свого облікового запису.

Для того, щоб зробити запит, наприклад, на фільми із самим високим рейтингом, потрібно спочатку створити запит API для отримання результатів, які будуть відсортовані в порядку зменшення рейтингу:

```
const response = await fetch(`https://api.themoviedb.org/3/discover/movie?api_key=${api_key}&primary_release_year=2022&sort_by_rating.desc`);
```

Далі треба додати отримані дані до списку відео:

```
for (let film of highest_rating_films) {
  let film_rating = await fetch('https://api.themoviedb.org/3/movie/' +
    film['id'] + '?api_key=' + api_key + '&language=en-US');
  film_rating = await film_rating.json();

  df.loc[df.length] = [film['title'], film_rating['rating']];
}
```

Даний метод проявляється в динаміці, тобто важливим є не тільки накопичити певну кількість оцінювань і після чого кінострічка стабільно вважатиметься популярною, важливо, щоб ця кількість оновлювалась в подальшому. Якщо кількість оцінок не буде змінюватись протягом певного періоду, наприклад, місяця, то це буде означати, що популярність відео значно впала. Якщо відео отримує нові рецензії або оцінки, то система розуміє його поточний стан, якщо ж нових оцінок немає, то система розуміє, що дані застарілі.

2.2 Алгоритм пошуку

Для пошуку фільму за назвою використовувалась технологія Axios.

Axios — це HTTP-бібліотека на основі промісів, яка дозволяє розробникам надсилати запити до власного або стороннього сервера для отримання даних. Він пропонує різні способи надсилання запитів, наприклад GET, POST, PUT/PATCH і DELETE.

Переваги:

- широка підтримка браузерів. Axios підтримує широкий спектр браузерів, включаючи старіші версії та Internet Explorer;
- автоматичне скасування запитів. Axios має вбудовану підтримку для скасування запитів. Це дозволяє створювати маркери скасування та пов'язувати їх із конкретними запитами, забезпечуючи ефективне скасування та очищення запитів, що очікують на розгляд;
- обробка помилок і перехоплювачі. Axios забезпечує надійний механізм обробки помилок.

Сам алгоритм пошуку відео за назвою можна описати так:

- додається вузол введення тексту та прив'язується його подія до обробника подій;
- створюється обробник для зміни стану;
- використовується фільтр у масиві для того, щоб зберегти лише ті дані, які відповідають введеній інформації.

2.3 Висновки до розділу 2

В даному розділі було визначено, як відбувається рекомендація популярних відео. Для цього були наведені відповідні атрибути, які використовуються системою для формування тих чи інших списків. Також було розглянуто технологію Axios та її переваги. Наведено опис роботи алгоритму пошуку фільмів за назвою.

3 ОПИС ПРОГРАМИ

3.1 Основні засоби розробки

3.1.1 Tailwind CSS

Tailwind CSS можна використовувати для оформлення вебсайтів найшвидшим і найпростішим способом. Tailwind CSS — це утиліта CSS-фреймворку для швидкого створення користувацьких інтерфейсів. Це низький рівень CSS-фреймворку з можливістю налаштування, який має усі будівельні блоки, необхідні для створення індивідуальних дизайнів без будь-яких стилів [6].

Принадність цієї технології полягає в тому, що вона не нав'язує специфікації дизайну чи того, як має виглядати сайт, треба просто об'єднати крихітні компоненти, щоб створити унікальний інтерфейс користувача.

Переваги Tailwind CSS:

- жодних назв для класів CSS та ідентифікаторів;
- мінімальна кількість рядків коду у файлі CSS;
- можна налаштувати конструкції для виготовлення компонентів;
- робить вебсайт адаптивним;
- вносить зміни в бажаний спосіб;
- CSS є глобальним за своєю природою, і якщо внести зміни у файл, властивість змінюється в усіх файлах HTML, пов'язаних із ним. Але за допомогою Tailwind CSS можна використовувати службові класи та вносити локальні зміни.

3.1.2 TypeScript

TypeScript — це високорівнева мова програмування з відкритим вихідним кодом, розроблена Microsoft, яка додає до JavaScript статичний тип із додатковими анотаціями типу. Він розроблений для розробки великих додатків і транспілюється на JavaScript [7]. Оскільки TypeScript є

надмножиною JavaScript, усі програми JavaScript є синтаксично дійсними TypeScript, але вони можуть не перевіряти тип з міркувань безпеки.

TypeScript можна використовувати для розробки програм JavaScript як для виконання на стороні клієнта, так і на стороні сервера (як у випадку з Node.js або Deno). Для транспіляції доступні кілька варіантів. Можна використовувати типовий компілятор TypeScript або можна викликати компілятор Babel для перетворення TypeScript у JavaScript.

Переваги TypeScript [8]:

- може виявити 15% поширених помилок;
- читабельність – легше побачити, що код повинен робити;
- вивчення TypeScript дасть краще розуміння та новий погляд на JavaScript.

Недоліки TypeScript:

- написання TypeScript займає більше часу, ніж JavaScript, оскільки потрібно вказати типи, тому для невеликих індивідуальних проєктів його використання може бути недоцільним;
- TypeScript потрібно скомпілювати, що може зайняти час, особливо у великих проєктах.

Схема компіляції TypeScript-коду в код JavaScript зображена на рис. 3.1.

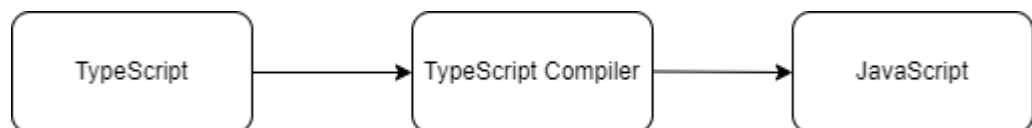


Рисунок 3.1 – Компіляція TypeScript-коду в код JavaScript

3.1.3 React.js

React — це бібліотека JavaScript, розроблена Facebook. Вона дозволяє розробникам легко створювати швидкі користувацькі інтерфейси для вебсайтів і програм. Основною концепцією React.js є віртуальний DOM. Це

дерево на основі компонентів JavaScript, створених за допомогою React, яке імітує дерево DOM. Бібліотека виконує найменшу кількість маніпуляцій з DOM, щоб підтримувати компоненти React в актуальному стані.

Компоненти React є будівельними блоками програми React. Компонент — це самодостатній фрагмент коду, який представляє частину інтерфейсу користувача. Компоненти можна комбінувати разом для створення складних інтерфейсів користувача. Компонент React може бути функцією JavaScript або класом. Функціональні компоненти простіші та використовуються для компонентів, яким не потрібно керувати своїм станом. Компоненти класу, з іншого боку, є більш потужними і можуть використовуватися, коли, наприклад, потрібно керувати станом.

Будучи частиною мови JavaScript, використання React дає багато переваг. Продукти, створені за допомогою React, легко масштабувати, єдина мова, яка використовується на стороні сервера/клієнта/мобільного пристрою, забезпечує виняткову продуктивність, існують шаблони робочого процесу для зручної командної роботи, код інтерфейсу користувача читається та підтримується тощо [9]. Провідні світові компанії використовували React та інші технології JS у деяких продуктах, що визначають ринок (Instagram, Reddit і Facebook – найяскравіші приклади).

Однією з основних причин використання React.js для веброзробки є остаточно оптимізований інтерфейс розробки бібліотеки та мова кодування. Таким чином, спрощений API React підсилений швидкою продуктивністю для досягнення безпроблемного та швидкого робочого процесу розробки. Компоненти та концепції React прості для розуміння, тому тут не потрібно багато часу на навчання.

На відміну від інших популярних фреймворків, таких як Vue та Angular, тут немає додаткових HTML-атрибутів.

Поточна екосистема React настільки обширна, що дозволяє розробникам створювати рішення для настільних комп'ютерів і мобільних додатків, створювати статичні вебсайти, керувати серверним рендерингом та

інтегрувати передові технологічні концепції.

3.1.4 Next.js

Next.js — це структура React, яка дає будівельні блоки для створення вебдодатків [10].

Під фреймворком мається на увазі, що Next.js обробляє інструменти та конфігурацію, необхідні для React, і забезпечує додаткову структуру, функції та оптимізацію програми.

Фреймворк Next.js заснований на фреймворку React. Даний фреймворк не потребує жодної конфігурації [11]. Next.js — це не стільки продуктивність, скільки покращення роботи розробника та спрощення створення повноцінних вебдодатків.

Структура фреймворку зображена на рис. 3.2.

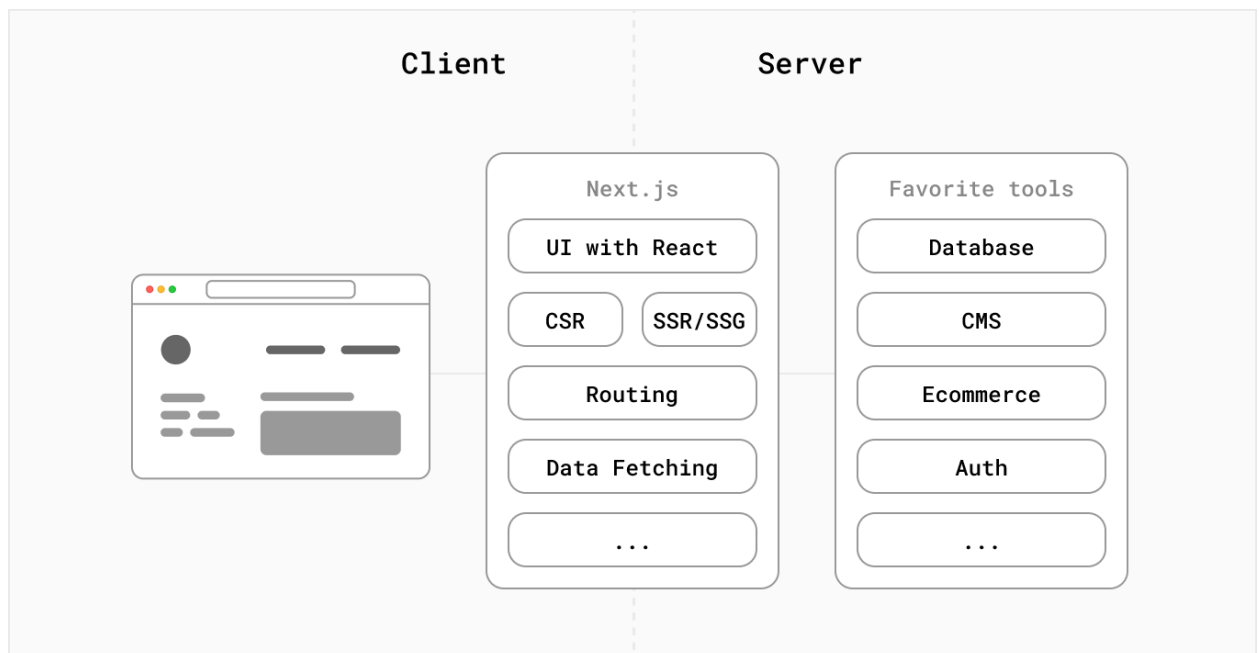


Рисунок 3.2 – Структура Next.js

Переваги Next.js:

- візуалізація на стороні сервера (SSR). Фреймворк підтримує SSR із

коробки, що дозволяє розробникам відтворювати сторінки на сервері перед тим, як надсилати їх клієнту. Це покращує SEO, продуктивність і час початкового завантаження сторінки;

- вбудована маршрутизація та поділ коду. Next.js забезпечує автоматичну маршрутизацію на основі файлової системи та дозволяє розділяти код, гарантуючи, що для кожної сторінки завантажується лише необхідний код, що забезпечує оптимізовану продуктивність;

- генерація статичного сайту (SSG). Технологія пропонує SSG, який генерує статичні файли HTML під час створення. Це корисно для вебсайтів, вміст яких не змінюється часто, забезпечуючи кращу продуктивність і зменшене навантаження на сервер;

- маршрути API. Має вбудовані маршрути API, які спрощують створення безсерверних функцій, уможливлюючи бездоганну інтеграцію логіки на стороні сервера з кодом інтерфейсу;

- досвід розробника. Next.js забезпечує спрощений досвід розробки з такими функціями, як гаряча заміна модулів, автоматичне перезавантаження та підтримка рішень CSS-in-JS [12].

Недоліки Next.js:

- складність для простих проєктів. Next.js багатий функціями, але для простих проєктів, які не вимагають SSR або SSG, це може створити непотрібну складність і додаткові витрати;

- крива навчання. Як і React.js, Next.js має власну криву навчання, особливо для розробників, які не знайомі з інструментами рендерингу та збірки на стороні сервера;

- обмежена гнучкість. Має власні угоди та маршрутизацію на основі файлової системи, що може обмежити гнучкість для певних розширених випадків використання або існуючих структур проєкту.

- Next.js оптимізований для статичної маршрутизації. Якщо потрібно використовувати динамічну маршрутизацію, продуктивність може знизитися.

В таблиці 3.1 наведено детальне порівняння фреймворків.

Таблиця 3.1 – Порівняння фреймворків

	Next.js	Gatsby
Пошукова оптимізація	+	+
Покращення взаємодії з користувачем	+	+
Продуктивність сайту	+	+
Налаштування взаємодії з користувачем	+	-
Тяжкі (змістовні) сайти	+	-
Легкі сайти	-	+

3.1.5 NextAuth.js

NextAuth.js — це повністю захищена та гнучка бібліотека аутентифікації, призначена для синхронізації з будь-якою службою OAuth із повною підтримкою входу без пароля [13].

NextAuth.js можна використовувати як з базою даних, так і без неї, і він має стандартну підтримку популярних баз даних, таких як MySQL, MongoDB, PostgreSQL і MariaDB. Його також можна використовувати без бази даних шляхом синхронізації з такими службами, як OAuth і JSON Web Token.

Переваги NextAuth.js:

- гнучкий і простий у використанні;
- володіння власними даними;
- безпечний за замовчуванням.

3.1.6 Prisma

Prisma — це сучасний інструмент доступу до даних, який спрощує та покращує взаємодію з базами даних у проєктах розробки програмного

забезпечення. Він служить проміжним рівнем між програмами та базами даних, забезпечуючи інтуїтивно зрозумілий і безпечний API для запитів і обробки даних. Prisma пропонує такі функції, як автоматичне створення запитів, безпека типів і можливості в режимі реального часу, що полегшує розробникам роботу з базами даних і створення надійних, масштабованих програм [14]. Він підтримує різноманітні популярні бази даних, такі як PostgreSQL, MySQL і SQLite, і легко інтегрується з такими мовами програмування, як TypeScript і JavaScript. Метою Prisma є підвищення продуктивності розробників, моделювання даних і продуктивності, зберігаючи зручний і гнучкий підхід до доступу до даних.

Основні функції:

- PrismaClient - це автоматично створений і безпечний конструктор запитів для Node.js і typescript;
- PrismaMigrate - інструмент міграції для легкого розвитку схеми бази даних від створення прототипу до виробництва;
- Prisma Studio - графічний інтерфейс для перегляду та редагування даних у базі даних.

Переваги Prisma:

- використовує Typescript для перевірки типу та її дуже легко інтегрувати з React, фреймворками React, NextJS, а також Remix;
- проста у використанні з GraphQL, а також підтримує вузли та експреси;
- підтримує бази даних як SQL, так і NoSQL;
- має дуже хорошу документацію, з якою легко слідувати та почати роботу, а також містить чудові приклади, які можна використати, щоб розпочати роботу з GraphQL, Nextjs і Remix [15].

3.1.7 MongoDB

MongoDB — це кросплатформна документоорієнтована програма бази

даних із доступним вихідним кодом. Класифікована як програма бази даних NoSQL, MongoDB використовує JSON-подібні документи з додатковими схемами. MongoDB розроблено компанією MongoDB Inc. Це документована розподілена база даних загального призначення, створена для сучасних розробників додатків і для хмарної ери. Написана мовою C++ [16].

MongoDB використовується для зберігання великих обсягів даних. Організації також використовують базу даних для спеціальних запитів, індексування, балансування навантаження, агрегації, виконання JavaScript на стороні сервера та інших функцій.

Замість використання таблиць і рядків, як у реляційних базах даних, як база даних NoSQL, архітектура MongoDB складається з колекцій і документів. Документи складаються з пар ключ-значення — основної одиниці даних MongoDB. Колекції, еквівалентні таблицям SQL, містять набори документів. MongoDB пропонує підтримку багатьох мов програмування, таких як C, C++, C#, Go, Java, Python, Ruby та Swift.

Особливості MongoDB:

- тиражування. Набір реплік — це два або більше екземплярів MongoDB, які використовуються для забезпечення високої доступності. Набори реплік складаються з основного та додаткового серверів. Основний сервер MongoDB виконує всі операції читання та запису, а вторинна репліка зберігає копію даних. Якщо основна репліка виходить з ладу, використовується вторинна репліка;
- масштабованість. MongoDB підтримує вертикальне та горизонтальне масштабування. Вертикальне масштабування працює, додаючи більше потужності до існуючої машини, тоді як горизонтальне масштабування працює, додаючи більше машин до ресурсів користувача;
- балансування навантаження. MongoDB обробляє балансування навантаження без необхідності окремого виділеного балансувальника навантаження за допомогою вертикального чи горизонтального масштабування;

- без схеми. MongoDB — це база даних без схем, що означає, що база даних може керувати даними без потреби в плані;

- документ. Дані в MongoDB зберігаються в документах з парами ключ-значення замість рядків і стовпців, що робить дані більш гнучкими порівняно з базами даних SQL [17].

Переваги:

- без схеми. Як і інші бази даних NoSQL, MongoDB не вимагає попередньо визначених схем. Вона зберігає будь-які типи даних. Це дає користувачам можливість створювати будь-яку кількість полів у документі, що полегшує масштабування баз даних MongoDB порівняно з реляційними базами даних;

- документоорієнтованість. Одна з переваг використання документів полягає в тому, що ці об'єкти відображаються на власних типах даних у кількох мовах програмування. Наявність вбудованих документів також зменшує потребу в об'єднанні бази даних, що може знизити витрати;

- масштабованість. Основною функцією MongoDB є її горизонтальна масштабованість, що робить її корисною базою даних для компаній, які працюють із застосуваннями великих даних. Крім того, шардинг дозволяє базі даних розподіляти дані між кластером машин. MongoDB також підтримує створення зон даних на основі сегментного ключа;

- підтримка сторонніх розробників. MongoDB підтримує кілька механізмів зберігання даних і надає API механізмів зберігання, що підключаються, що дозволяє стороннім розробникам розробляти власні механізми зберігання;

- агрегація. СУБД також має вбудовані можливості агрегації, що дозволяє користувачам запускати код MapReduce безпосередньо в базі даних, а не запускати MapReduce на Hadoop. MongoDB також містить власну файлову систему під назвою GridFS, подібну до розподіленої файлової системи Hadoop. Файлова система використовується в першу чергу для зберігання файлів, розмір яких перевищує обмеження розміру BSON у 16 МБ на документ. Ці

подібності дозволяють використовувати MongoDB замість Hadoop, хоча програмне забезпечення бази даних дійсно інтегрується з Hadoop, Spark та іншими фреймворками обробки даних.

Недоліки:

- безперервність. За допомогою стратегії автоматичного відновлення після відмови користувач налаштовує лише один головний вузол у кластері MongoDB. Якщо головний вузол виходить з ладу, інший вузол автоматично перетворюється на новий головний. Це перемикання обіцяє безперервність, це може зайняти до хвилини. Для порівняння, база даних Cassandra NoSQL підтримує кілька головних вузлів. Якщо один майстер виходить з ладу, інший залишається поряд, створюючи високодоступну інфраструктуру бази даних;

- обмеження. Єдиний головний вузол MongoDB також обмежує швидкість запису даних у базу даних. Запис даних має бути записаний на головному вузлі, а запис нової інформації в базу даних обмежений потужністю цього головного вузла;

- послідовність даних. MongoDB не забезпечує повної посилальної цілісності через використання обмежень зовнішнього ключа, що може вплинути на узгодженість даних;

- безпека. Крім того, автентифікація користувача не ввімкнена за замовчуванням у базах даних MongoDB. Однак зловмисники атакували велику кількість незахищених систем MongoDB, що призвело до додавання параметра за замовчуванням, який блокує мережеві підключення до баз даних, якщо вони не були налаштовані адміністратором бази даних.

3.1.8 Node.js

Node.js (Node) — це кросплатформне середовище виконання з відкритим кодом для виконання коду JavaScript. Node широко використовується для програмування на стороні сервера, що дозволяє розробникам використовувати JavaScript для коду на стороні клієнта та на

стороні сервера без необхідності вивчати додаткову мову.

Node містить механізм V8 JavaScript, той самий, що використовується в Google Chrome та інших браузерях. Він написаний на C++ і може працювати на macOS, Linux, Windows та інших системах. Механізм аналізує та виконує код JavaScript. Він може працювати незалежно від середовища браузера, вбудований у програму C++ або реалізований як окрема програма. Механізм V8 компілює JavaScript всередині, використовуючи процеси вчасно, щоб пришвидшити виконання.

У налаштуваннях багатопоточної обробки кожен сервер має обмежений пул потоків, до якого він може отримати доступ. Отже, кожного разу, коли сервер отримує запит, він витягує потік із пулу та призначає його цьому запиту, щоб подбати про його обробку [18]. У цьому випадку обробка є синхронною та послідовною, що означає, що за раз виконується одна операція.

У багатопотоковій обробці потік вибирається кожного разу, коли робиться запит, доки не будуть використані всі обмежені потоки. Коли це відбувається, сервер повинен чекати, поки зайнятий потік знову стане вільним. Це може призвести до повільних і неефективних додатків, що призведе до негативного впливу на будь-що, від досвіду клієнтів до конверсій. Особливо це може стати проблемою, якщо програма має працювати з великою кількістю одночасних запитів клієнтів.

Однак Node.js використовує однопотокову обробку. Різниця між ними така: однопотокові архітектури обробляють кожен запит за допомогою єдиного основного потоку, використовуючи цикли подій для виконання блокуючих операцій введення/виведення без блокування.

Теоретично однопотокова архітектура може працювати та масштабуватися набагато швидше та ефективніше, ніж багатопотокові налаштування.

Переваги Node.js:

- висока ефективність;
- масштабованість;

- розширюваність;
- зменшує час завантаження за допомогою кешування;
- покращує час відгуку та підвищує продуктивність;
- підтримка спільноти.

3.1.9 The Movie Database API

API MoviesDatabase пропонує інформацію про фільми, телешоу та акторів. База даних фільмів містить URL-адресу трейлера YouTube, нагороди, повну біографію та іншу корисну інформацію. Цей API надає повні й оновлені дані для понад 9 мільйонів назв (фільмів, серіалів і епізодів) і 11 мільйонів акторів, знімальної групи та учасників акторського складу [19].

3.2 Вибір середовища розробки

Велику роль під час створення програмного забезпечення, крім інструментів для написання коду і розгортання системи, грають засоби для автоматизації процесу розробки. У якості середовища (IDE) для розробки вебзастосунку було обрано WebStorm.

WebStorm — це інтегроване середовище розробки для JavaScript і суміжних технологій. Як і інші IDE JetBrains, це робить ваш досвід розробки більш приємним, автоматизуючи рутинну роботу та допомагаючи вам з легкістю вирішувати складні завдання.

WebStorm — це передова IDE, яка підтримує JavaScript, HTML і CSS, включаючи сучасні інновації, такі як TypeScript, React, Angular, Vue.js і Node.js. IDE відома своїм зручним інтерфейсом, інтелектуальною підтримкою кодування та потужними функціями налагодження. Незалежно від того, чи є ви досвідченим розробником, чи початківцем, WebStorm виявиться безцінним активом у вашому шляху веброзробки [20].

Можливості IDE:

- сучасні фреймворки – WebStorm надає розширену допомогу в кодуванні для Angular, React, Vue.js і Meteor. Підтримкою React Native, PhoneGap, Cordova та Ionic для мобільної розробки на стороні сервера за допомогою Node.js;

- розумний редактор – IDE аналізує проєкт, щоб забезпечити найкращі результати завершення коду для всіх підтримуваних мов. Сотні вбудованих перевірок повідомляють про можливі проблеми прямо під час введення тексту та пропонують варіанти швидкого вирішення;

- навігація та пошук – WebStorm допомагає ефективніше обходити код і економити час під час роботи з великими проєктами. Є можливість перейти до визначення методу, функції чи змінної лише одним натиском миші;

- налагодження – WebStorm надає вбудований налагоджувач для клієнтського коду (який працює з Chrome) і програм Node.js. Можна розміщувати точки зупину, крокувати по коду та оцінювати вирази;

- тестування. Тестування виконується з легкістю, оскільки WebStorm інтегрується з програмою для виконання тестів Karma, Mocha, Jest і Protractor. IDE дозволяє запускати та налагоджувати тести безпосередньо в середовищі, переглядати результати в привабливому та візуальному форматі та переходити до тестового коду;

- відстеження та профілювання – WebStorm містить spy-js, вбудований інструмент, який допомагає відстежувати код JavaScript. В результаті можна дізнатися, як файли пов'язані з викликами функцій, і ефективно визначити будь-які можливі вузькі місця;

- інструменти для створення. Простий уніфікованим інтерфейсом користувача для запуску завдань Grunt, Gulp або npm прямо з IDE. Усі завдання перераховані у спеціальному вікні інструменту;

- інструменти якості коду. На додаток до сотень власних перевірок, WebStorm може запускати ESLint, JSCS, TSLint, Stylelint, JSHint або JSLint для коду та висвітлювати будь-які проблеми на льоту, прямо в редакторі [21];

- шаблони проєктів. Почати нові проєкти можна з екрана привітання,

використовуючи популярні шаблони проєктів, як-от Express або Web Starter Kit, і отримати доступ до ще більшої кількості генераторів проєктів завдяки інтеграції з Yeoman;

- VCS - WebStorm надає уніфікований інтерфейс користувача для роботи з багатьма популярними системами контролю версій, забезпечуючи узгоджену взаємодію з користувачами в git, GitHub, SVN, Mercurial і Perforce;

- локальна історія. Незалежно від того, використовується VCS чи ні, локальна історія може справді зберегти код. У будь-який момент можна переглянути історію певного файлу чи каталогу та повернутися до будь-якої з попередніх версій;

- налаштування - WebStorm легко налаштовувати.

В таблиці 3.2 наведено детальне порівняння програм для розробки коду.

Таблиця 3.2 – Порівняння програм для розробки коду

	WebStorm	Visual Studio Code
IDE	+	-
Редактор вихідного коду	+	+
Відкритий код	-	+
Підтримка відладки	+	+
Підказки по коду	+	+
Модульне тестування	+	-
Рефакторинг коду	+	+

3.3 Архітектура вебзастосунку

Для написання вихідного коду клієнтської частини вебзастосунку було обрано мову TypeScript з використанням таких технологій як React та фреймворку Next.js. Для реалізації бекенд частини застосунку було обрано

Node.js. Для системи управління базами даних було обрано MongoDB. Загальна архітектура програмного забезпечення зображена на рис. 3.3.

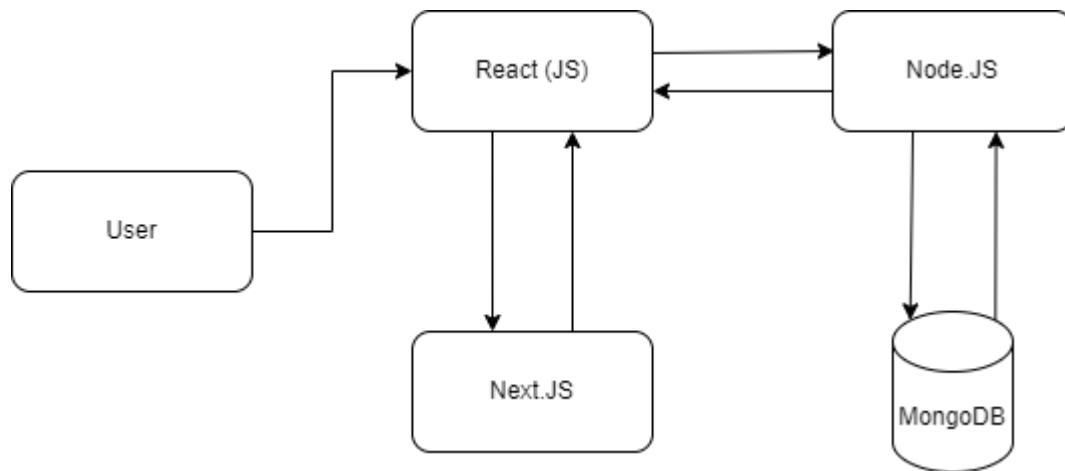


Рисунок 3.3 – Загальна архітектура ПЗ

3.4 Вимоги до програмного забезпечення

Програмний продукт повинен забезпечувати виконання таких функцій:

- реєстрація та авторизація користувача;
- зручне навігаційне меню;
- перегляд фільмів;
- лаконічний опис обраного відео;
- керування відеоплеєром;
- додавання відео до списку улюблених, перегляд та керування цим списком;
- відображення популярних кінострічок;
- пошук кінострічок у пошуковому рядку.

Схема функцій користувача представлена на рис. 3.4.

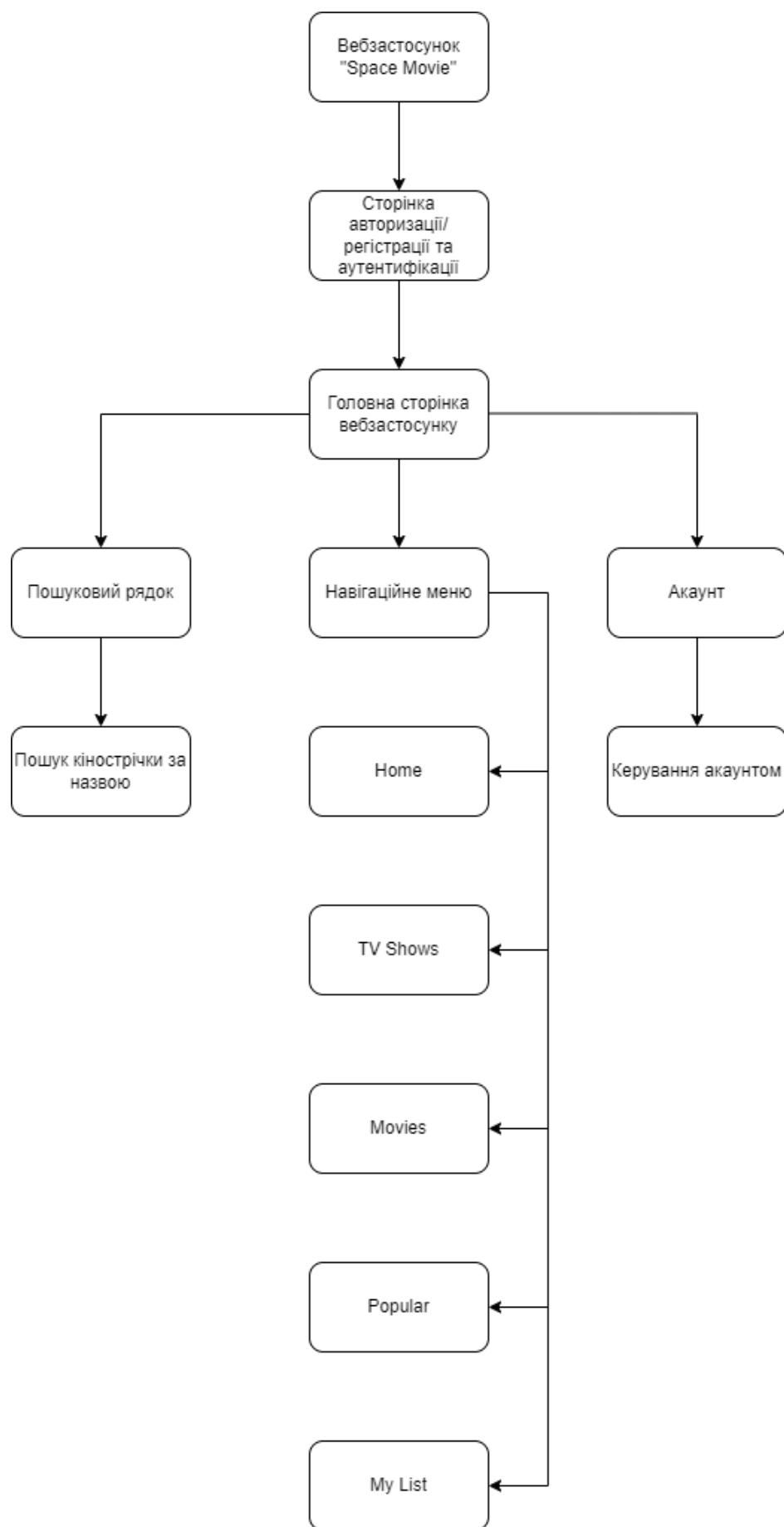


Рисунок 3.4 – Схема функцій користувача

3.5 Аналіз вимог до розробки дизайну програми

Оскільки розроблювана програма призначена для стримінгу відео, то основний акцент потрібно робити в аналітичному стилі, а саме в конкретній секції потрібно розміщувати важливу інформацію для швидкого візуального аналізу, який не займе багато часу. Тому в результаті дизайн вебзастосунку було створено з дотриманням мінімалістичного стилю. Мінімалізм визначається як дизайн або стиль, в якому найпростіші та найменші елементи використовуються для створення максимального ефекту. Мінімалізм бере свій початок у мистецтві — з простими лініями, лише кількома кольорами та акуратним розташуванням цих ліній і кольорів [22].

Переваги:

- мінімалістичний дизайн має більшу швидкість завантаження;
- більш зручна атмосфера;
- допомагає зосередитися на вмісті;
- білий простір допомагає сайту «дихати»;
- простіша навігація;
- зручність використання.

Не потрібно заходити занадто далеко, використовуючи мінімалізм. Не треба видаляти необхідні та функціональні елементи лише для того, щоб дизайн виглядав чистим і простим. Тонку межу між мінімалістичним дизайном і відсутністю функцій легко переступити.

При розробці програмного забезпечення варто звести речі до мінімуму, визначити пріоритети для необхідних функцій і видалити усі нефункціональні елементи.

3.6 Висновки до розділу 3

В цьому розділі було розглянуто основні інструменти для розробки вебзастосунку, такі як Tailwind CSS, TypeScript, React.js, Next.js. NextAuth.js,

Prisma, Node.js, MongoDB, The Movie Database.

Було представлено загальну архітектуру програмного забезпечення, сформульовано функціональні вимоги до розроблюваного застосунку та розглянуто стиль дизайну вебзастосунку.

Після порівняння фреймворків було обрано Next.js. Також був наведений детальний опис та порівняння програм для розробки коду, в результаті чого було обрано IDE WebStorm.

Базою даних було обрано MongoDB для зберігання даних користувача та можливості додавання кінострічок. Також було використано The Movie Database API для доступу до URL-адрес трейлерів кінострічок.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розробка бази даних

Для проєктування схеми бази даних було використано безкоштовний вебзастосунок draw.io.

База даних проєкту складається з двох колекцій: користувач та фільм. Основним методом для створення бібліотеки фільмів є використання The Movie Database API. Але для демонстрації можливості створення бібліотеки відео у середовищі mongoDB було власноруч додано кінострічку під час розробки програмного забезпечення безпосередньо в саму базу даних.

Розроблена схема бази даних представлена на рис. 4.1.

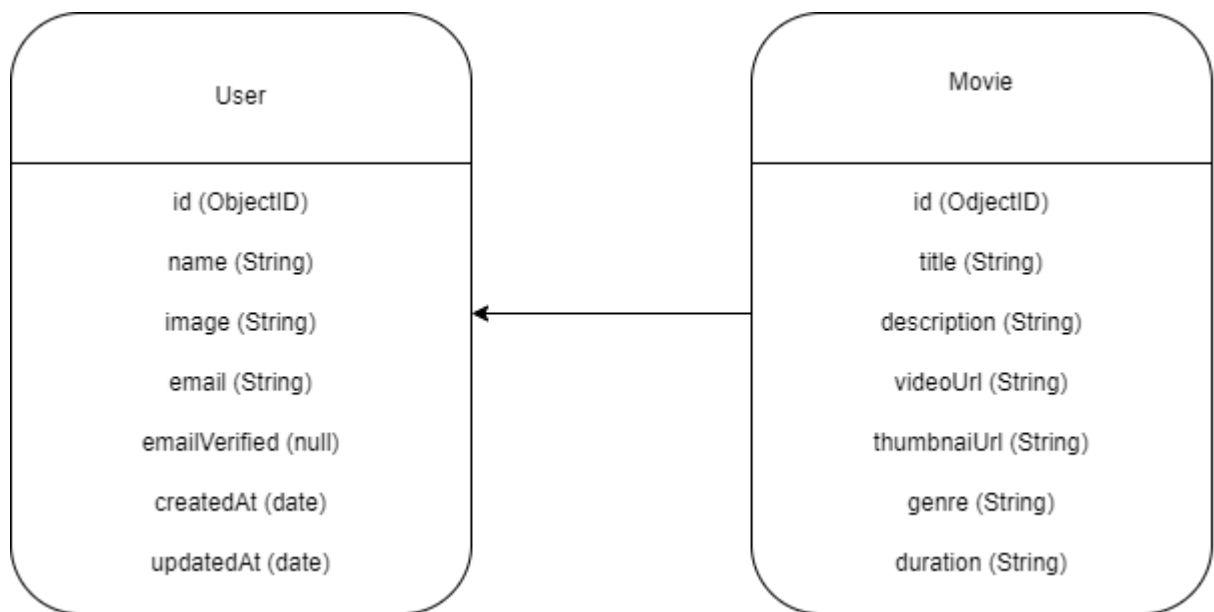


Рисунок 4.1 – Схема бази даних

Колекція «Користувач» має наступні властивості:

- «id»: унікальний ідентифікатор користувача в системі;
- «name»: ім'я користувача в системі;
- «image»: аватар користувача;
- «email»: електронна адреса користувача;
- «emailVerified»: поле відображає статус підтвердження електронної

адреси;

- «createdAt»: поле, яке містить в собі дату створення акаунту в системі;
- «updatedAt»: поле, яке містить інформацію про останню авторизацію в системі.

Колекція «Фільм» зберігає такі дані:

- «id»: унікальний ідентифікатор кінострічки в системі;
- «title»: назва фільму;
- «description»: поле, яке містить лаконічний опис фільму;
- «videoUrl»: властивість, яка зберігає в собі URL-адресу відео;
- «thumbnailUrl»: поле, яке містить зображення плитки фільму, що відображається у списку кінокартин;
- «genre»: жанр кінострічки;
- «duration»: тривалість відео.

Вигляд колекції «Користувач» у середовищі mongoDB представлено на рис. 4.2.

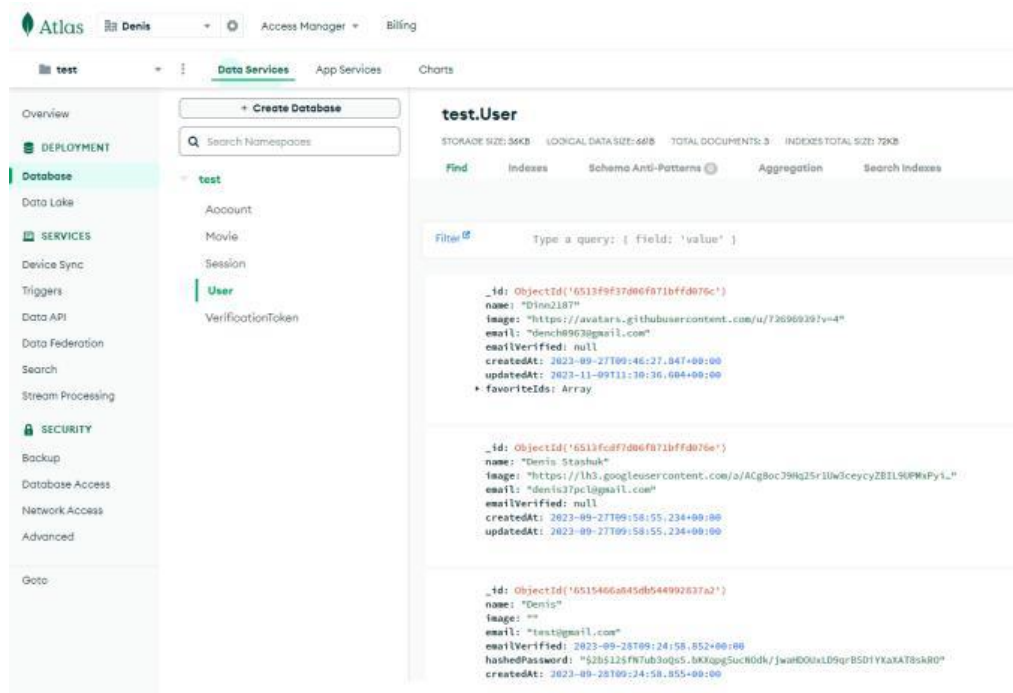


Рисунок 4.2 – Вигляд колекції «Користувач» у середовищі mongoDB

Потрібно звернути увагу на те, що після того, як користувач додає

кінострічку до списку улюблених, в базі даних цього користувача з'являється поле «favoriteIds», яке містить в собі масив унікальних ідентифікаторів фільмів із списку улюблених. Вигляд колекції «Фільм» зображено на рис. 4.3.

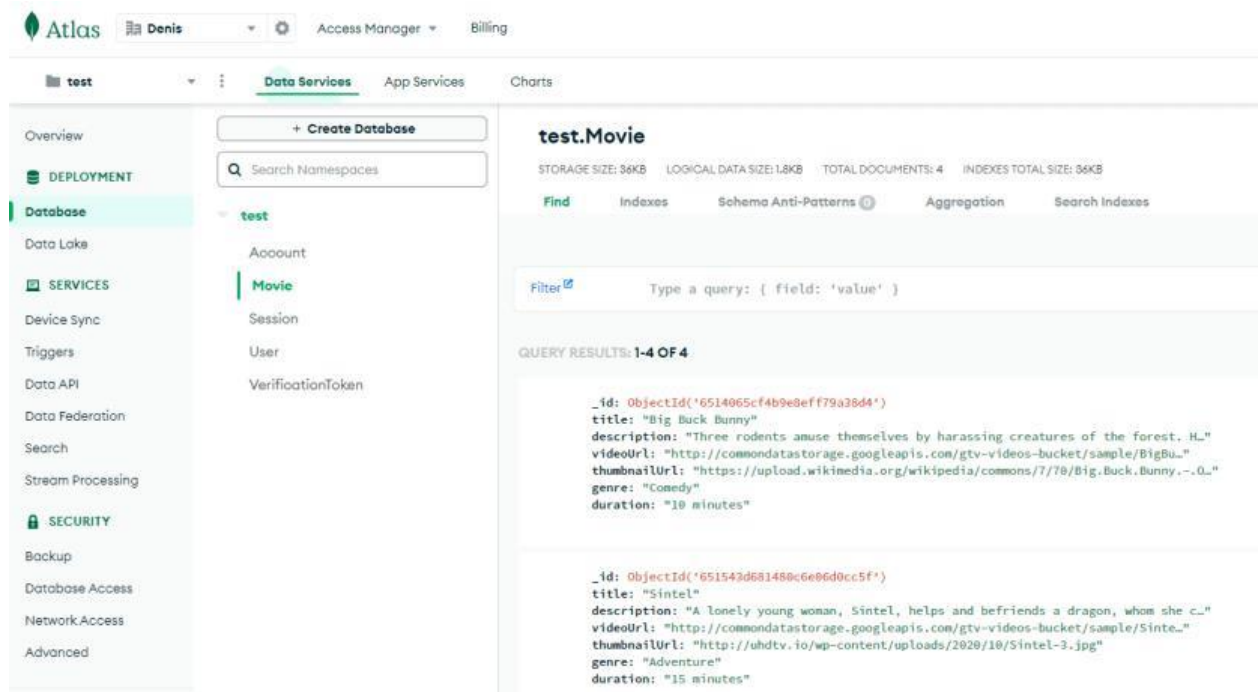


Рисунок 4.3 – Вигляд колекції «Фільм» у середовищі mongoDB

4.2 Способи реєстрації та авторизації

В даному підрозділі буде розглянуто можливі методи реєстрації та авторизації користувача на сайті.

Аутентифікація є важливою та конфіденційною функцією в програмах, де облікові дані користувача, такі як ім'я користувача, адреса електронної пошти та пароль, використовуються для перевірки його особи.

Для реалізації цього функціоналу було використано Next.js та NextAuth.js.

NextAuth.js має API на стороні клієнта, який можна використовувати для взаємодії з сесіями у програмі. Дані сесію, отримані від постачальників, містять корисне навантаження користувача, і це може відображатися користувачеві після успішного входу.

Корисне навантаження не містить конфіденційних даних та призначене для відображення користувачеві.

NextAuth.js також надає `useSession()` React Hook, який можна використовувати для перевірки статусу входу користувача. Тим часом NextAuth.js надає REST API, який використовується програмою React.

`NEXTAUTH_SECRET` — це критична змінна середовища, яка використовується NextAuth.js для підпису та перевірки веб-токенів JSON і шифрування даних сеансу. Ці значення має бути довгим випадковим рядком, який зберігається в безпеці та ніколи не оприлюднюється. Щоб згенерувати цю змінну, можна використовувати вбудований модуль шифрування Node.js.

Це генерує безпечне, випадкове, 32-байтове значення, закодоване як шістнадцятковий рядок.

Для початку потрібно встановити NextAuth.js. Після завершення інсталяції можна переходити до етапу додавання способів аутентифікації.

4.2.1 Google OAuth

Представлений спосіб реєстрації та авторизації користувача дає змогу використовувати свій акаунт Google для створення облікових записів на інших сайтах. Для додавання цього методу використовувалась Google Cloud Console.

Процес підключення даного способу можна розбити на етапи:

- створення нового проєкту Google;
- ввімкнення API Google Диска;
- створення додатку OAuth;
- створення облікових даних та клієнта;
- генерація токенів доступу та оновлення.

Після реєстрації нашого додатку Google OAuth створює унікальний ідентифікатор клієнта та секретний ключ.

Процес налаштування Google OAuth зображено на рис. 4.4 – 4.5.

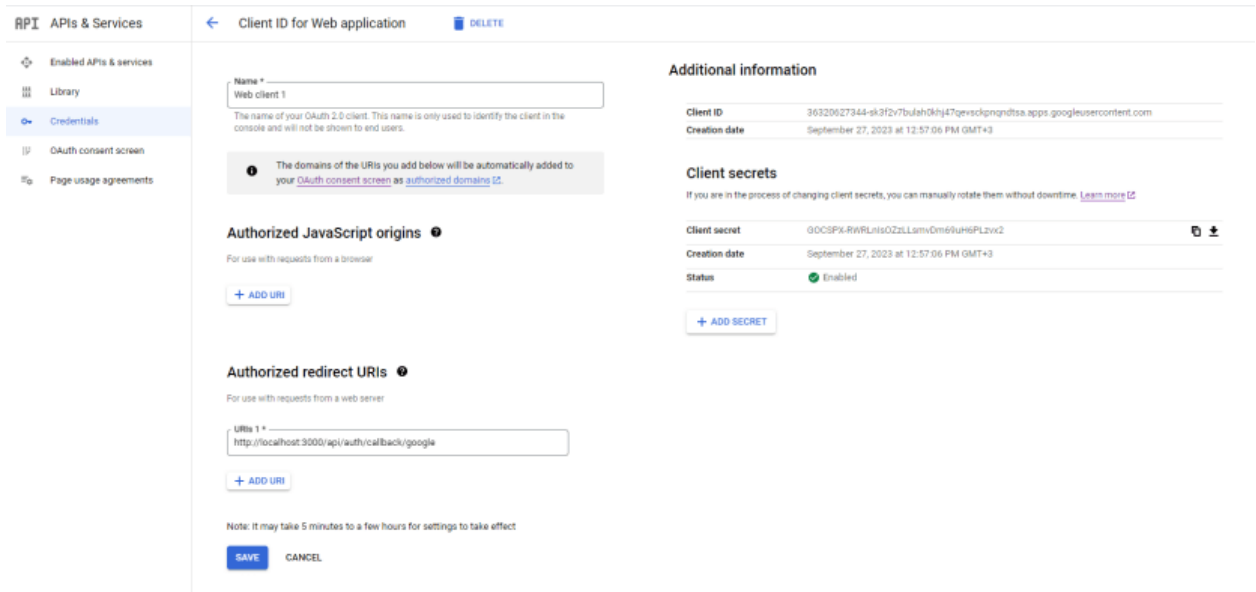


Рисунок 4.4 – Налаштування Google OAuth

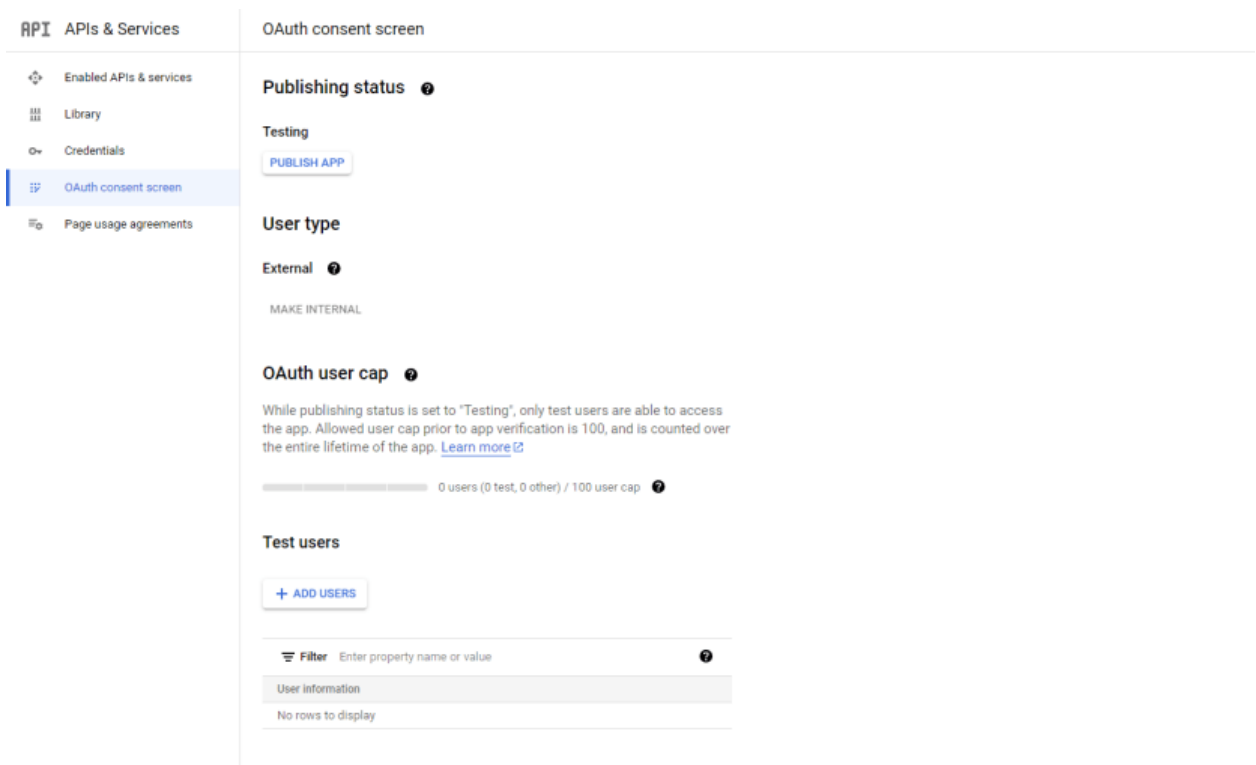


Рисунок 4.5 – Налаштування Google OAuth

4.2.2 GitHub OAuth

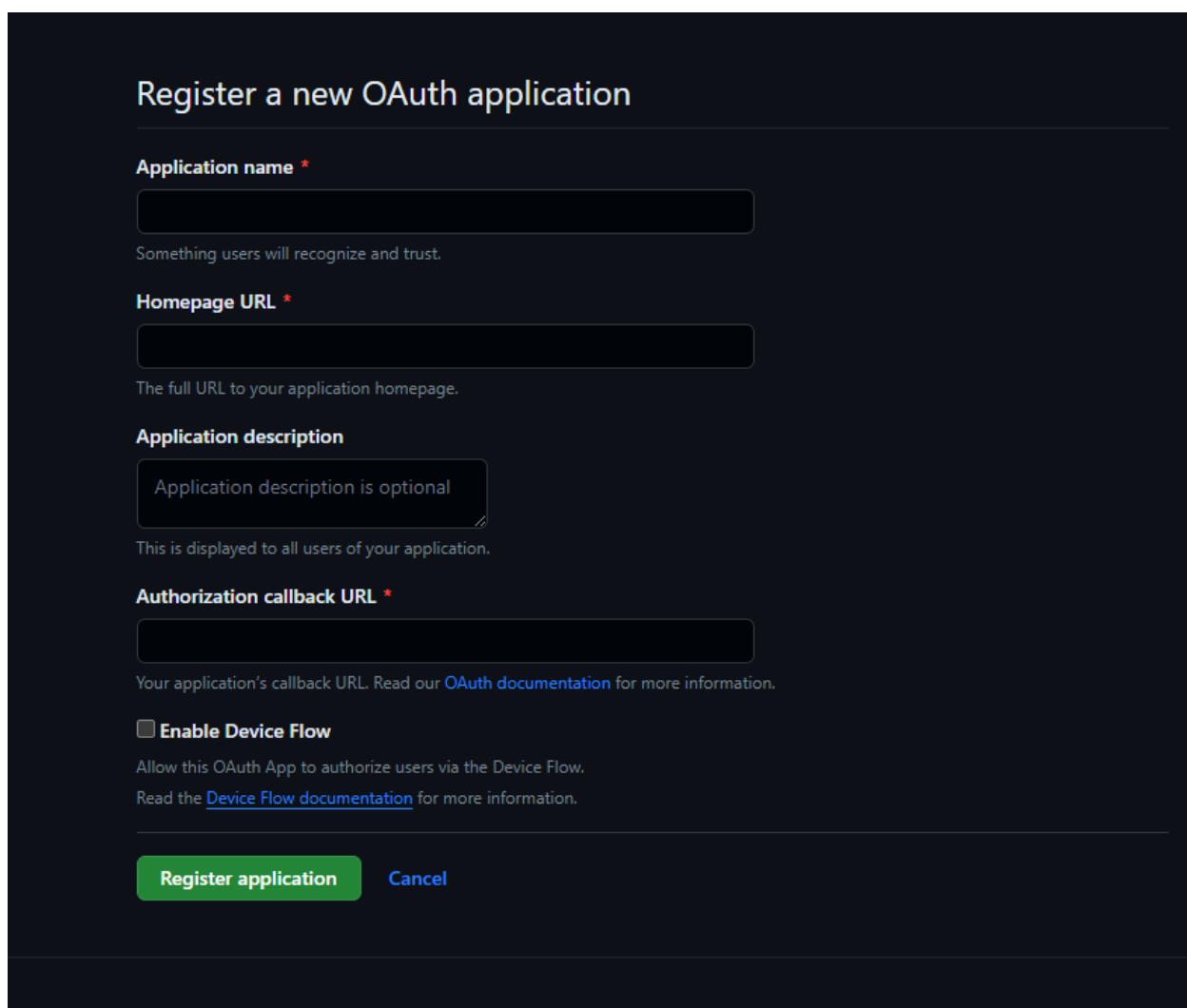
Наступним методом реєстрації є GitHub OAuth. Однією з можливостей платформи GitHub є використання такого інструмента розробника, як OAuth

Apps.

При створенні OAuth додатку виділяються такі основні поля:

- назва програми;
- URL-адреса домашньої сторінки. Це повна URL-адреса домашньої сторінки додатка;
- URL-адреса зворотного виклику авторизації. Це URL-адреса, на яку GitHub перенаправлятиме користувача після того, як він успішно авторизується в додатку.

Налаштування GitHub OAuth зображено на рис. 4.6 – 4.7.



The screenshot shows the 'Register a new OAuth application' form on GitHub. The form is set against a dark background with light text. It includes the following fields and options:

- Application name ***: A text input field with a placeholder 'Something users will recognize and trust.'
- Homepage URL ***: A text input field with a placeholder 'The full URL to your application homepage.'
- Application description**: A text area with a placeholder 'Application description is optional' and a note 'This is displayed to all users of your application.'
- Authorization callback URL ***: A text input field with a note 'Your application's callback URL. Read our [OAuth documentation](#) for more information.'
- Enable Device Flow**: A checkbox option with a note 'Allow this OAuth App to authorize users via the Device Flow. Read the [Device Flow documentation](#) for more information.'

At the bottom, there are two buttons: a green 'Register application' button and a blue 'Cancel' button.

Рисунок 4.6 – Створення OAuth додатку

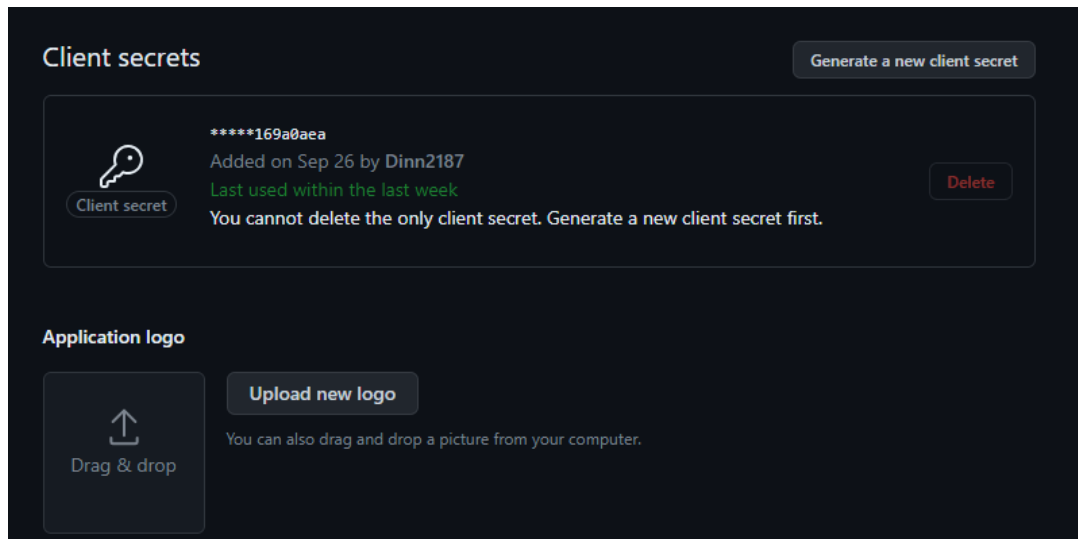


Рисунок 4.7 – Налаштування Github OAuth

4.2.3 Електронна адреса та пароль

Останнім способом зареєструватися на сайті є звичне для нас використання електронної адреси та паролю. Користувач вводить електронну адресу та пароль, які потім можна зберегти у менеджері паролей.

В результаті всі згенеровані ідентифікатори та секретні ключі, а також посилання на базу даних знаходяться в окремому .env-файлі (рис. 4.8).

```

DATABASE_URL= "mongodb+srv://denchotest:381b3K81Wo60wEJC@cluster0.m8hclkj.mongodb.net/server"
GOOGLE_CLIENT_ID= 36496624017-sk3f2v7hipo0khj47qevskfygnndtsa.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET= GKBARQ-RWRLnIs0ZzLLkqcDm69uH6PLkbc2
GITHUB_ID=19846e2jm17460d5236a
GITHUB_SECRET=a1a8c6204ad4ff279e7aa9974359726f169a0glz
NEXTAUTH_JWT_SECRET= "NEXT-JWT-SECRET"
NEXTAUTH_SECRET= "NEXT-SECRET"

```

Рисунок 4.8 – Вміст .env-файлу

4.3 Структурна схема проєкту

В цьому підрозділі буде представлено структурну схему проєкту.

Структура директорій проєкту представлена на рис. 4.9 та 4.10.

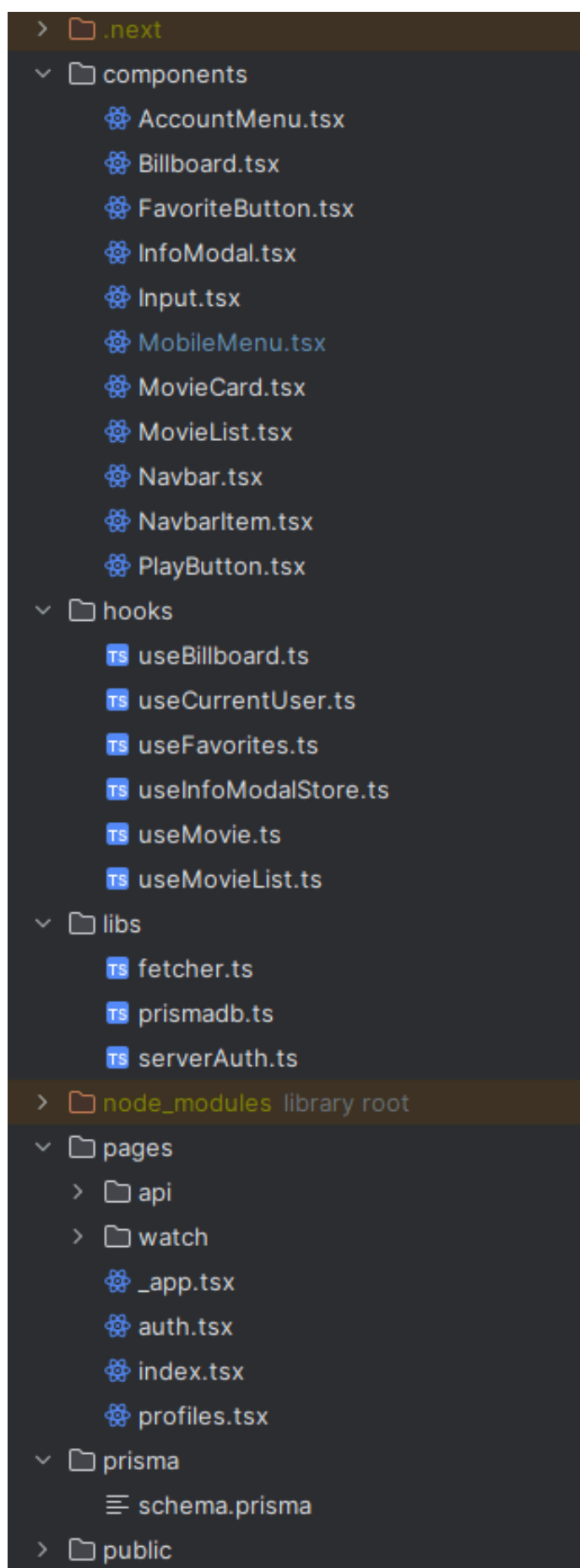


Рисунок 4.9 – Структура директорій проєкту

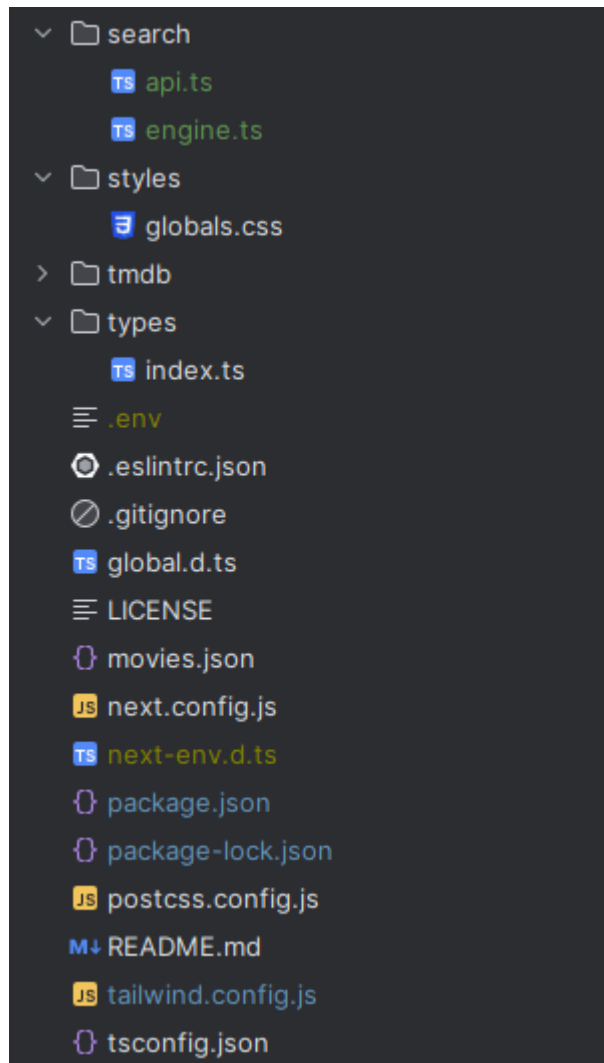


Рисунок 4.10 – Структура директорій проєкту

Директорія `.next` містить в собі папки та файли, які додаються автоматично при створенні Next.js-додатку.

Директорія `components` призначена для зберігання основних файлів вебзастосунку, а саме:

- `AccountMenu.tsx`: файл, який відповідає за керування акаунтом (вихід з акаунту);
- `Billboard.tsx`: файл, що відповідає за кнопку, яка відображає додаткову інформації про фільм;
- `FavoriteButton.tsx`: файл, який містить реалізацію кнопки додавання кінострічки до списку улюблених;
- `InfoModal.tsx`: файл, що містить в собі реалізацію модального вікна,

яке з'являється при натисканні на плитку фільму;

- Input.tsx: файл, що відповідає за функціонування полів форми реєстрації користувача (електронна адреса, пароль);
- MobileMenu.tsx: файл, що призначений для реалізації меню навігації сайту та його адаптацію під мобільні пристрої;
- MovieCard.tsx: файл, який відповідає за картки кінокартин, що відображаються на сайті;
- MovieList.tsx: файл, що містить в собі реалізацію відображення списку фільмів на сайті;
- Navbar.tsx: файл, що призначений для реалізації хідера сайту (меню навігації та меню керування акаунтом);
- NavbarItem.tsx: файл, що призначений для зміни стану кнопки при наведенні на неї;
- PlayButton.tsx: файл, який відповідає за кнопку відтворення кінострічки (старт/ пауза).

Директорія `hooks` зберігає в собі файли, які дозволяють під'єднатися до станів та методів життєвого циклу бібліотеки, а також повторне використання коду для вирішення загальних задач.

Директорія `libs` містить в собі файли, які відповідають за реалізацію захисту шляхів клієнтів та шляхів API, які використовуються при аутентифікації.

Директорія `node_modules` додається автоматично при створенні проєкту та містить в собі інші директорії, серед яких можна виділити директорію `next-auth`, яка призначена для взаємодії з компонентами, які відповідають за реєстрацію або авторизацію користувача, та директорію `mongoose`, яка містить в собі інші директорії та файли, призначені для коректної роботи бази даних.

Директорія `pages` включає в себе директорію `api`, яка містить файл `nextauth.js`, який в свою чергу містить реалізацію сторінки авторизації користувача. Папка `watch` містить файл `movieId`, який зберігає список

унікальних ідентифікаторів кінокартин. Також директорія `pages` використовується для зберігання файлів основних сторінок вебзастосунку, таких як сторінка авторизації, сторінка профіля користувача та основна сторінка вебзастосунку.

Директорія `prisma` зберігає в собі файл `schema.prisma`. Цей файл є основним файлом конфігурації для налаштування Prisma та складається з таких частин:

- джерела даних: джерело даних, до яких підключається Prisma (наприклад, база даних `mongodb`);
- генератори: визначає, які клієнти мають бути згенеровані на основі моделі даних (наприклад, `Prisma Client`);
- визначення моделі даних: визначає моделі додатків (форму даних на джерело даних) та їхні зв'язки.

Директорія `public` призначена для зберігання зображень карток фільмів та зображень, які використовувались при створенні дизайну сайту.

Директорія `search` містить в собі файли `api.ts` та `engine.ts`, які відповідають за код пошуку фільмів за назвою.

Директорія `styles` відповідає за зберігання файлу `global.css`, який містить код імпорту компонентів `tailwind CSS` для стилізації вебзастосунку.

Директорія `tmdb` зберігає в собі код, який забезпечує працездатність API, що надає інформацію про відео.

Директорія `types` містить файл `index.ts`, який в свою чергу містить детальну інформацію про окремо взяті кінострічку.

Потрібно звернути увагу на файл `.env`, який містить в собі URL-адресу для підключення бази даних, унікальні ідентифікатори та секретні ключі Google OAuth та GitHub OAuth.

Інші файли додаються автоматично під час створення проєкту або завантаження модулів під час роботи.

4.4 Опис реалізованих функцій

В даному підрозділі розглянемо деякі функції, які будуть використовуватись у проєкті. Розглянемо функції, що відповідають за реалізацію певного функціоналу.

Функції директорії components:

- AccountMenu() – функція, яка виконує операцію виходу з акаунту при натисканні на відповідну кнопку;
- Billboard() – функція, яка відкриває плитку обраної кінострічки для надання додаткової інформації про неї;
- FavoriteButton() – функція, що дає змогу додавати або видаляти фільм до списку улюблених;
- Input() – функція, яка обробляє введені користувачем дані при реєстрації на сайті;
- MobileMenu() – функція, яка відповідає за відображення навігаційного меню сайту при натисканні на відповідну кнопку в режимі перегляду зі смартфона;
- MovieCard() – функція, що відображає плитки фільмів на головній сторінці сайту та отримує інформацію про них з бази даних;
- MovieList() – функція, що відображає списки кінокартин на головній сторінці сайту в довільному порядку;
- PlayButton() – функція, яка дозволяє керувати станом відео.

Функції директорії hooks:

- useBillboard() – функція, яка будує частину сайту, що містить в собі кінокартину, яка відображається у верхній частині сайту;
- useFavorite() – функція, що будує секцію зі списком улюблених фільмів;
- useMovie() – функція, що будує секцію сайту зі списком фільмів.

Функції директорії libs:

- serverAuth() – функція, яка обробляє дані користувача при заповненні

відповідних полів та синхронізує їх із Prisma.

Функції директорії pages:

- NextAuth() – функція, яка взаємодіє з файлом .env та prismadb для перевірки унікальних ідентифікаторів та секретних ключів, які використовуються при реєстрації через Google OAuth або GitHub OAuth.

- handler() – функція, яка створює json-файл з унікальними ідентифікаторами кінострічок;

- selectProfile() – функція, що відповідає за відображення блоку зі списком авторизованих користувачів та подальший вибір акаунту;

4.5 Висновки до розділу 4

В даному розділі було розроблено схему бази даних проєктованого програмного забезпечення. Розглянуто способи реєстрації та авторизації користувача. Описано структурну схему реалізованого проєкту. Наведено деякі функції, які використовуються у програмі.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Призначення програми

Розроблювана програма призначена для стримінгу відео, що в свою чергу дозволить провести час із задоволенням. Програмне забезпечення надає доступ до різноманітних кінокартин, а саме: фільми, телевізійні шоу, популярні кінострічки та ін.

Програма дозволяє швидко та зручно знайти цікаву кінострічку та насолодитися процесом перегляду.

5.2 Виконання програми

Для користування вебзастосунком необхідно бути зареєстрованим. Якщо користувач ще не має облікового запису, він може створити його у відповідній формі реєстрації (рис 5.1).

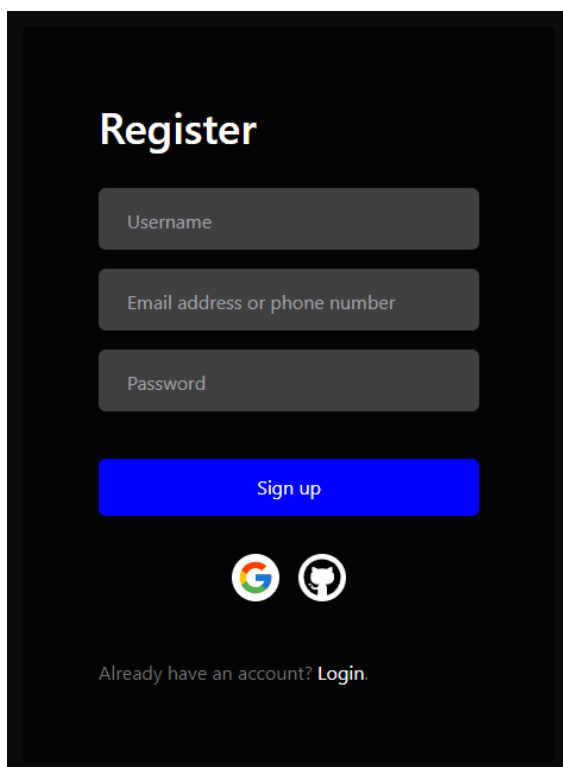
The image shows a registration form titled "Register" on a dark background. It contains three input fields: "Username", "Email address or phone number", and "Password". Below these fields is a blue "Sign up" button. At the bottom, there are two circular icons for Google and GitHub, followed by the text "Already have an account? Login.".

Рисунок 5.1 – Форма реєстрації

Якщо користувач вже авторизований у системі, при вході на сторінку вебдодатку, йому буде запропоновано увійти до свого облікового запису (рис 5.2).

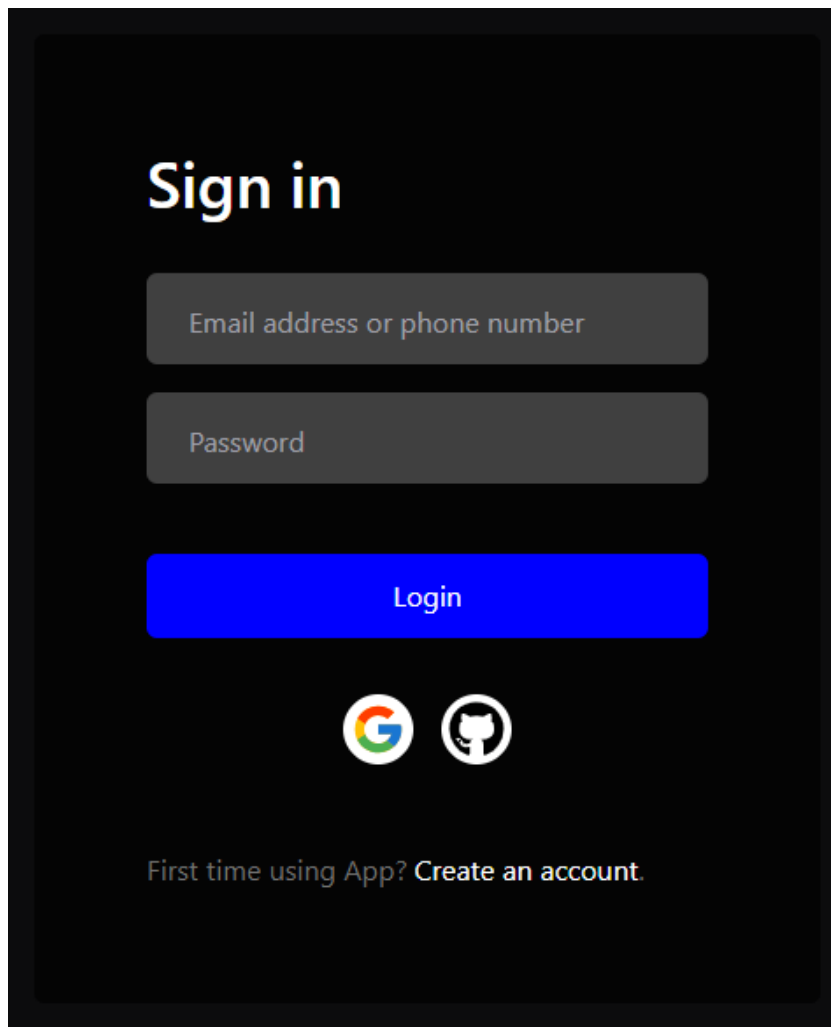
The image shows a 'Sign in' form on a dark background. At the top, the text 'Sign in' is displayed in a large, white, sans-serif font. Below this, there are two input fields: the first is labeled 'Email address or phone number' and the second is labeled 'Password'. Both labels are in a light gray font. Below the input fields is a prominent blue button with the word 'Login' in white. Under the button are two circular icons: the Google 'G' logo and the GitHub Octocat logo. At the bottom of the form, there is a link that reads 'First time using App? Create an account.' in a light gray font.

Рисунок 5.2 – Форма авторизації користувача

Зареєструватися чи авторизуватися у системі можна, використовуючи декілька способів:

- електронна пошта та пароль;
- натиснути кнопку «Google» та обрати акаунт;
- натиснути кнопку «GitHub», після чого буде здійснено авторизацію користувача.

Після вибору одного із запропонованих способів створення акаунту, з'являється сторінка підтвердження користувача (рис. 5.3).



Рисунок 5.3 – Сторінка підтвердження користувача

Після того як користувач зареєструвався на сайті, в базі даних, в колекції «Користувач», з'являються його дані (рис. 5.4).

```
_id: ObjectId('6515466a845db544992837a2')
name: "Denis"
image: ""
email: "test@gmail.com"
emailVerified: 2023-09-28T09:24:58.852+00:00
hashedPassword: "$2b$12$fN7ub3oQsS.bKXqpg5ucN0dk/jwaHD0UxLD9qrBSDiYXaXAT8skR0"
createdAt: 2023-09-28T09:24:58.855+00:00
updatedAt: 2023-09-28T09:24:58.855+00:00
```

Рисунок 5.4 – Дані нового користувача

Після входу користувача до системи, він опиняється на головній сторінці вебзастосунку. (рис. 5.5). Якщо проскролити вниз, можна побачити категорії відео (рис. 5.6).

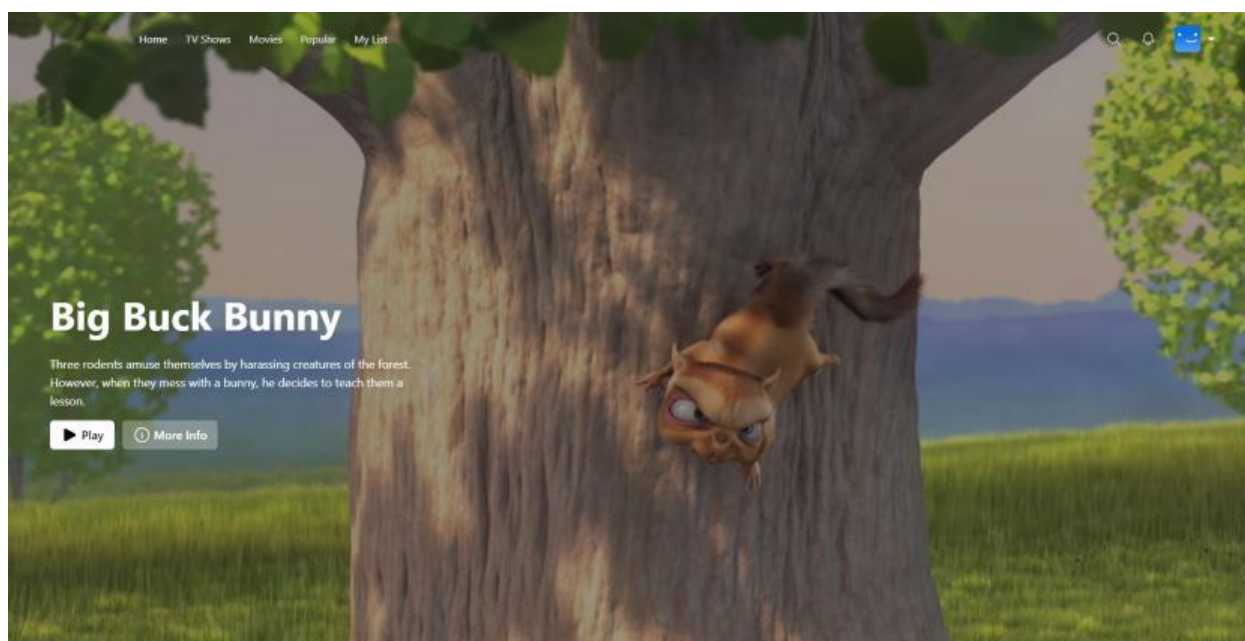


Рисунок 5.5 – Головна сторінка вебзастосунку

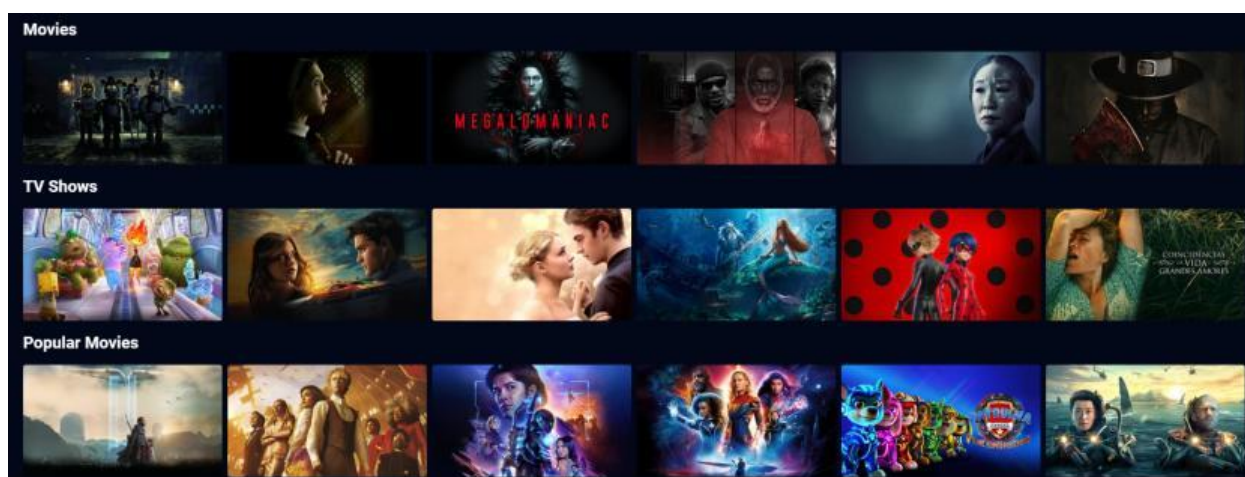


Рисунок 5.6 – Категорії відео

Основні елементи керування сайтом знаходяться у навігаційному меню (рис. 5.7). Використовуючи можливості цього меню, користувач може переглядати список телевізійних шоу при натисканні на кнопку «TV Shows» (рис. 5.8), список кінострічок при натисканні на кнопку «Movies» (рис. 5.9), список популярних кінокартин при натисканні на кнопку «Popular» (рис. 5.10), список улюблених фільмів при натисканні на кнопку «My List» (рис. 5.11). Також при натисканні на значок пошуку є можливість знайти відповідну кінокартину за назвою. Якщо клікнути на значок профіля, з'явиться меню в

якому, використовуючи кнопку «Sign Out» можна вийти зі свого акаунта та опинитися на сторінці авторизації. Результати зображені на рис. 5.12 – 5.13. При натисканні кнопки «Home» користувач повертається на головну сторінку вебзастосунку.



Рисунок 5.7 – Навігаційне меню



Рисунок 5.8 – Результат натискання на кнопку «TV Shows»

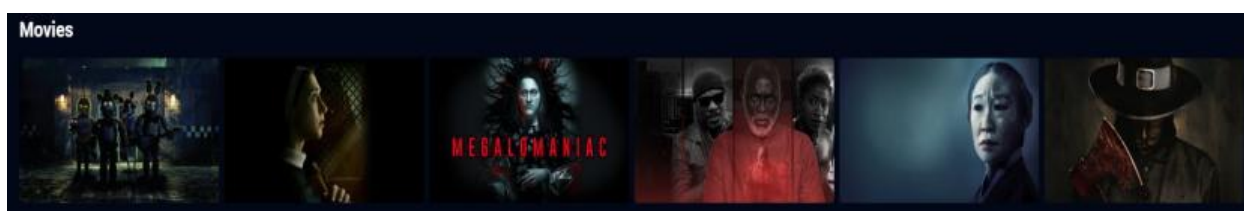


Рисунок 5.9 – Результат натискання на кнопку «Movies»

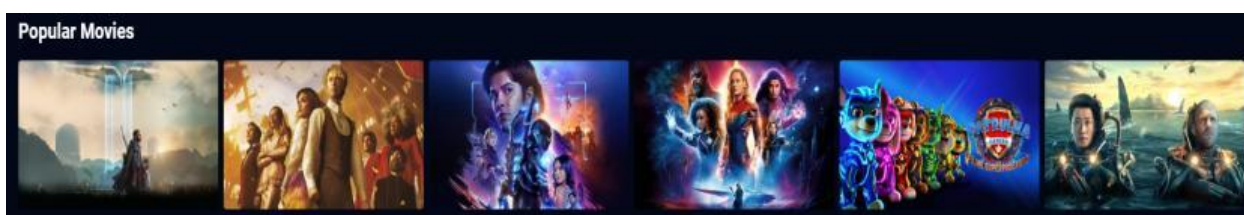


Рисунок 5.10 – Результат натискання на кнопку «Popular»



Рисунок 5.11 – Результат натискання на кнопку «My List»

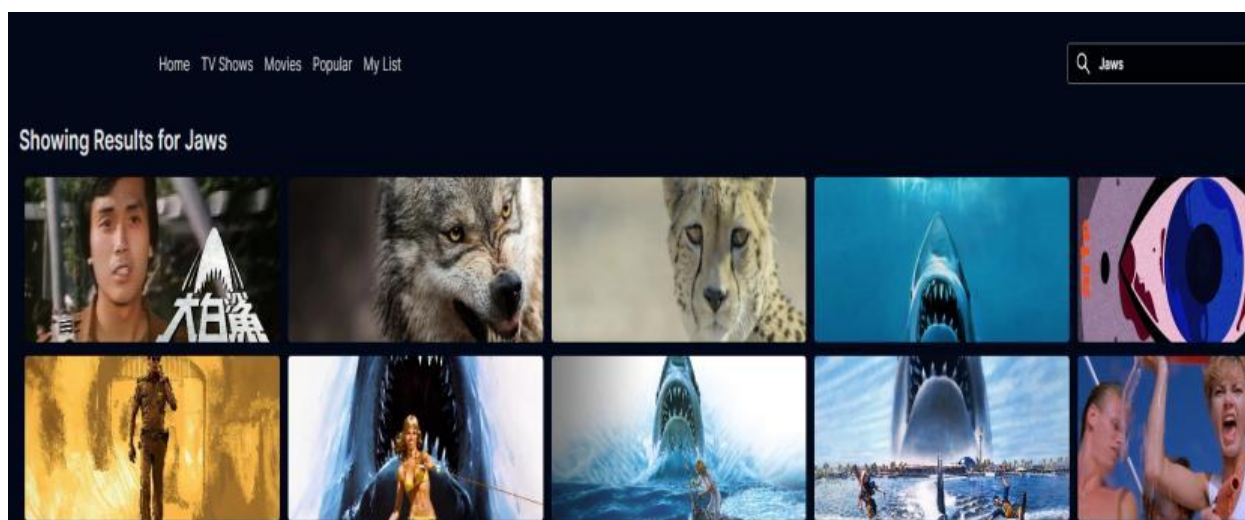


Рисунок 5.12 – Результат пошуку фільму за назвою

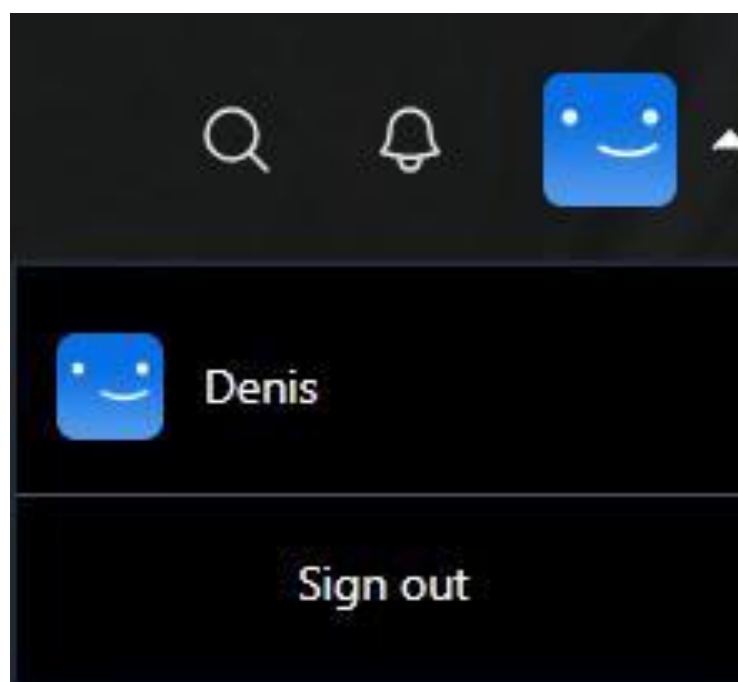


Рисунок 5.13 – Меню керування акаунтом

При натисканні на плитку фільму з'являється вікно, яке містить більш детальну інформацію та дозволяє виконувати різні маніпуляції з кінострічкою (рис. 5.14). Розглянемо ці функції на прикладі доданого власноруч до бази даних відео. Дане вікно містить основну інформацію про фільм, таку як тривалість кінокартини, жанр, лаконічний опис, можливість додати фільм до списку улюблених та, звичайно, кнопку відтворення відео. При натисканні на кнопку відтворення кінострічки з'являється медіаплеєр (рис. 5.15).

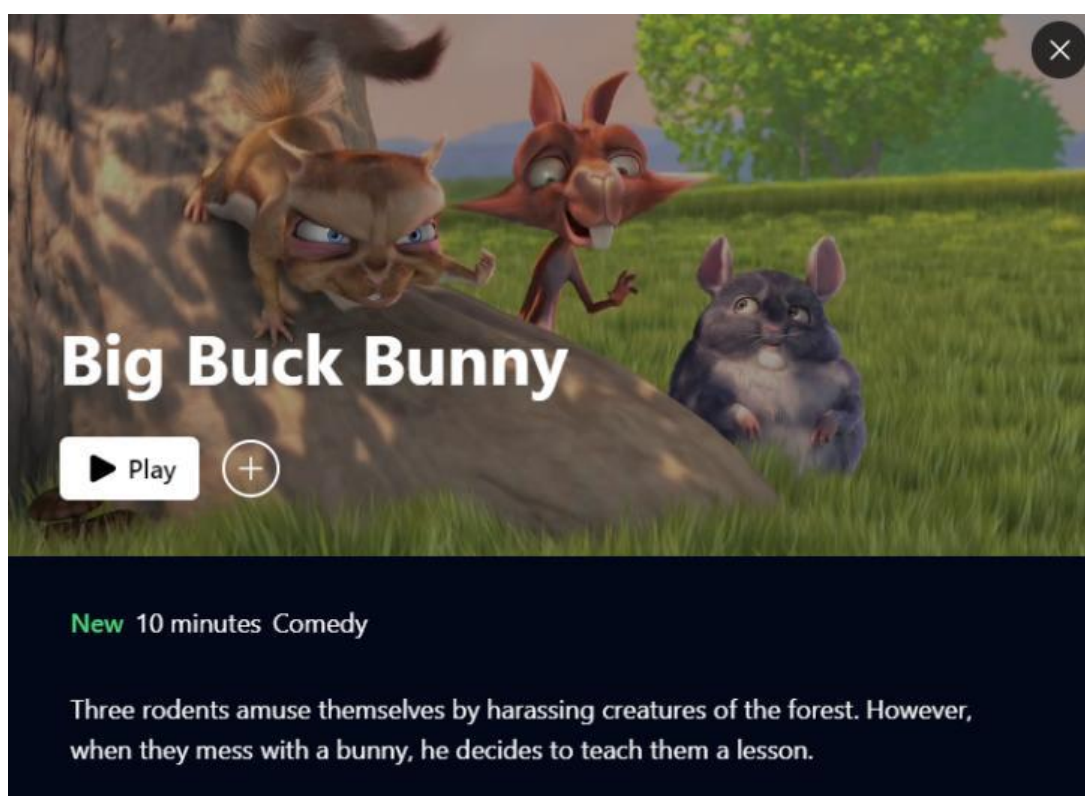


Рисунок 5.14 – Вікно з детальною інформацією про фільм

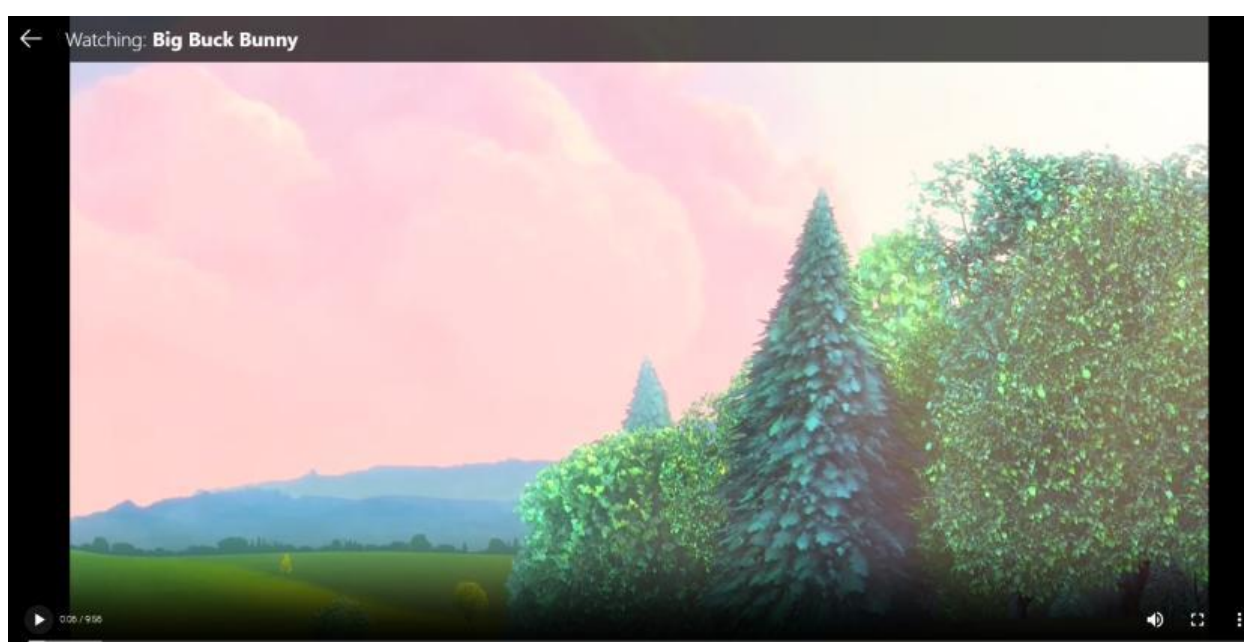


Рисунок 5.15 – Медіаплеєр відео

Сучасне життя важко уявити без смартфона. Смартфони та інші гаджети сприяють тому, що багато рутинних задач стають простішими. Телефон дозволяє спілкуватися з іншими людьми, виконує функцію

фотоапарату, мапи, сховища даних та, звичайно, надає можливість дивитися улюблене кіно.

Саме тому під час розробки програми для стримінгу відео однією з основних задач було створити адаптивний вебзастосунок, який буде коректно працювати на будь-яких пристроях.

В результаті були проведені тести на адаптивність. Результати зображені на рис. 5.16 – 5.17.

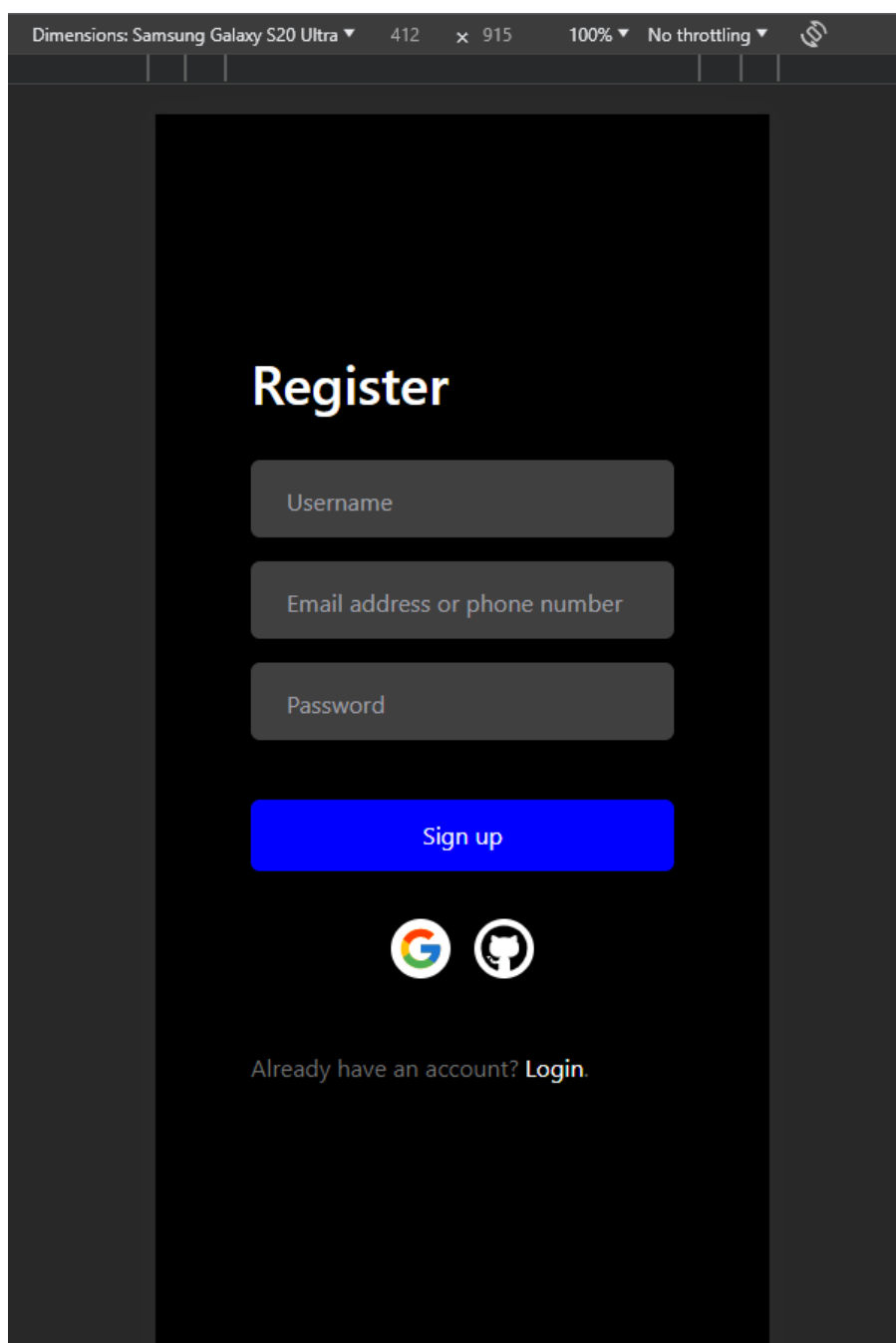


Рисунок 5.16 – Розширення 412x915

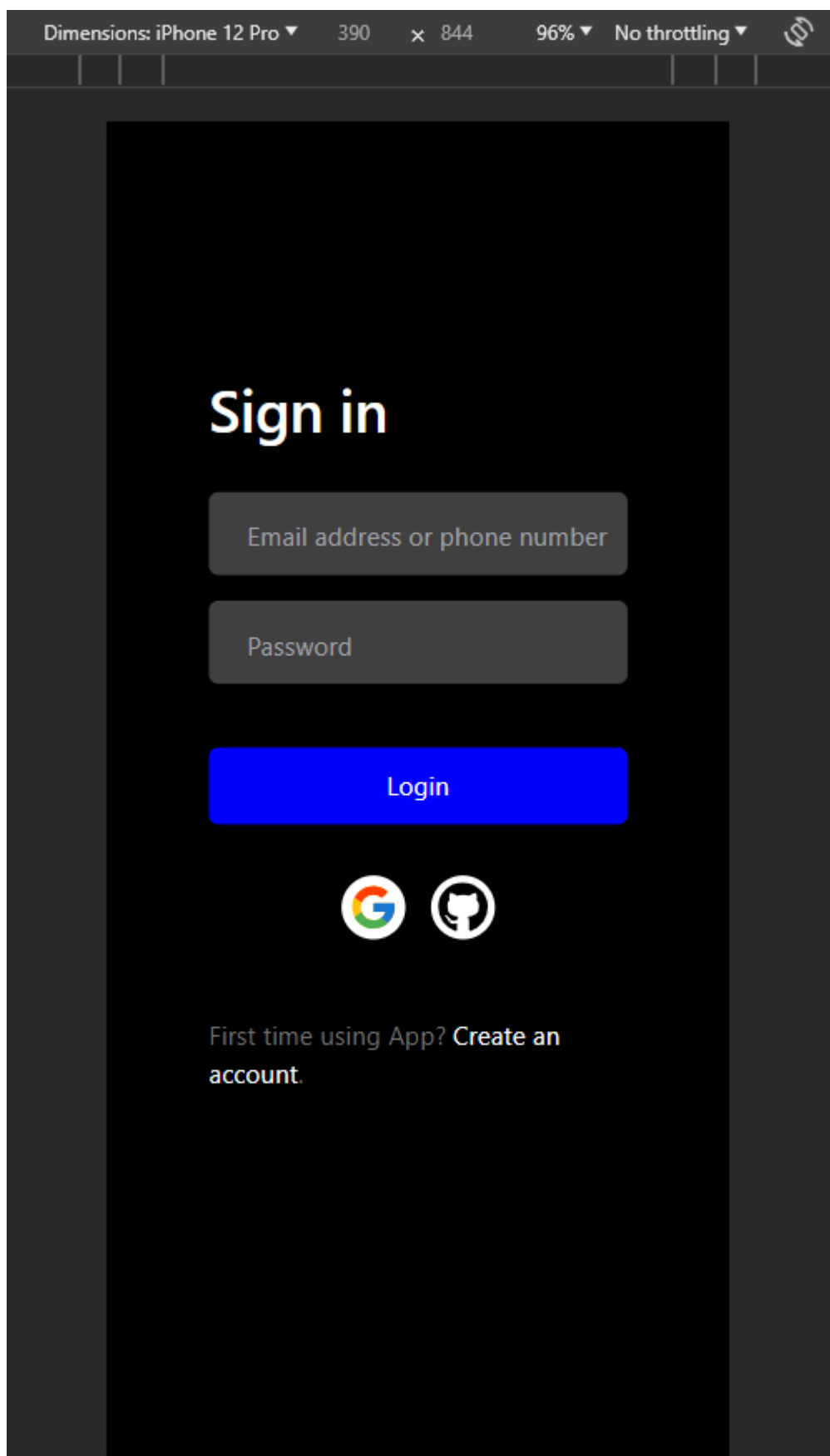


Рисунок 5.17 – Розширення 390x844

Результати зображені на рис. 5.18 – 5.19.

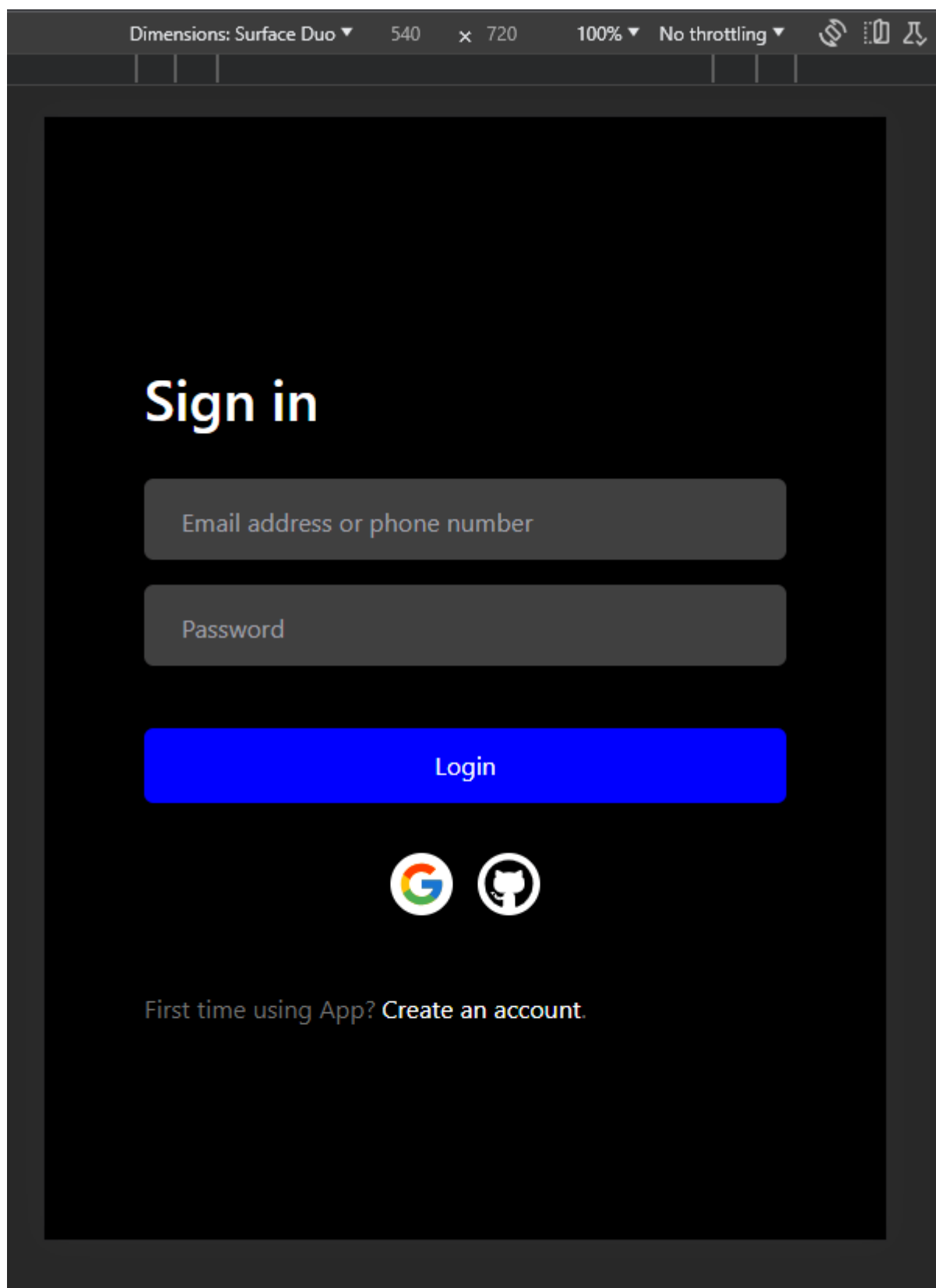


Рисунок 5.18 – Розширення 540x720

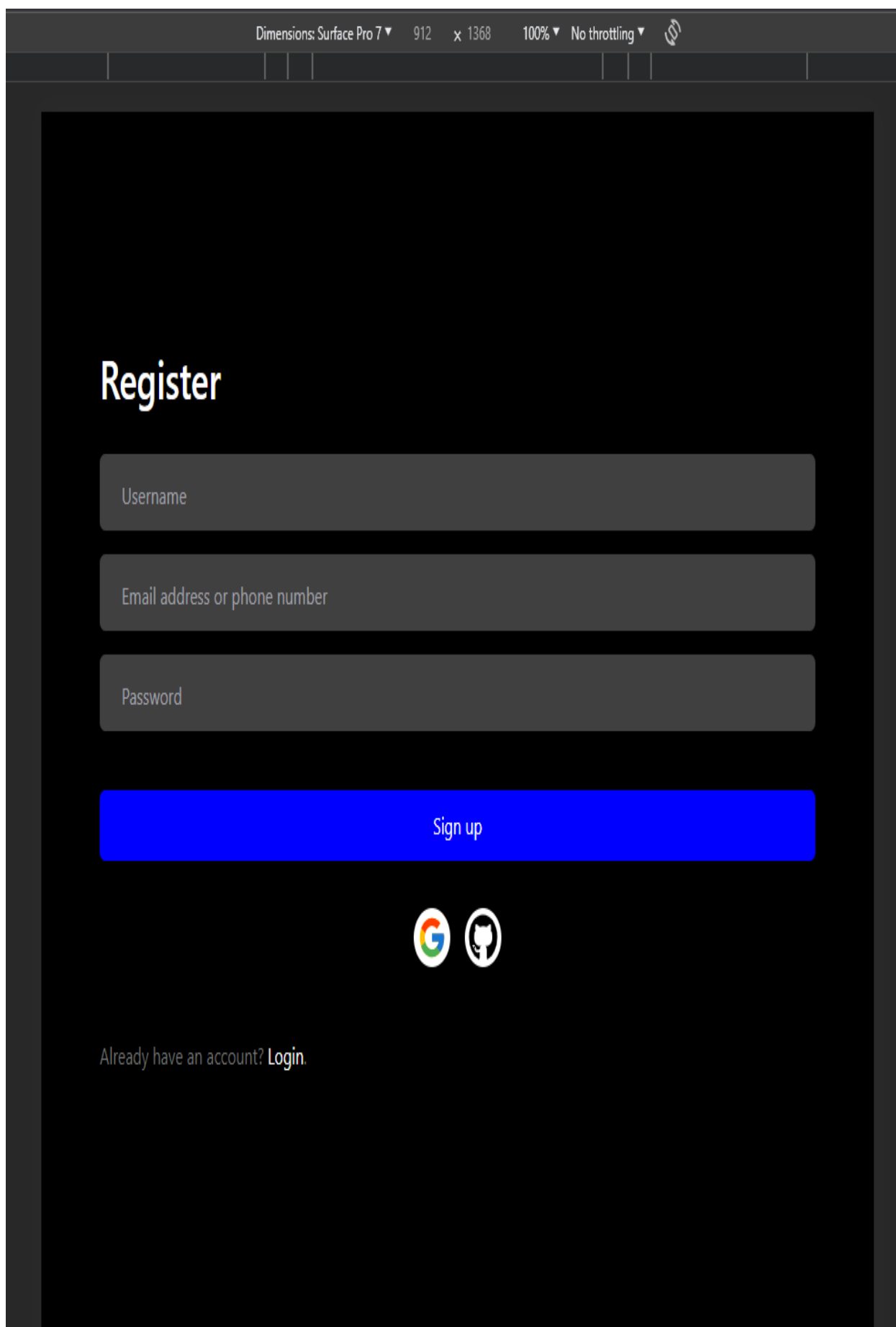


Рисунок 5.19 – Розширення 912x1368

Результати зображені на рис. 5.20 – 5.21.

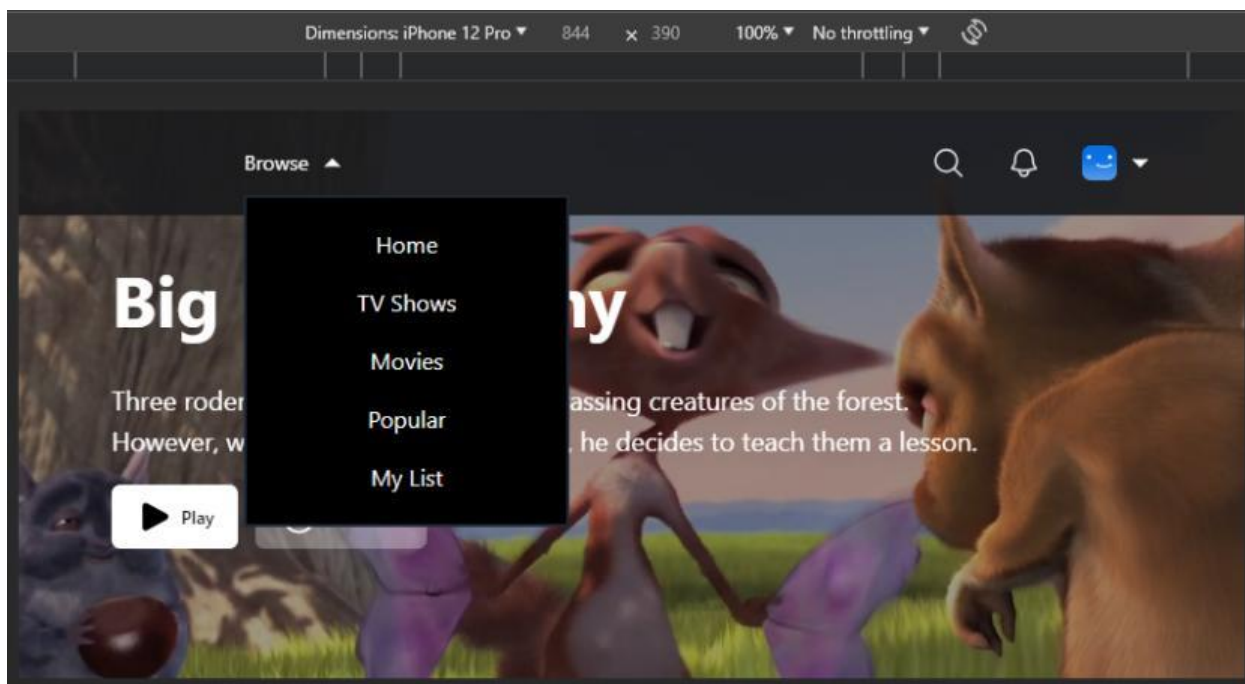


Рисунок 5.20 – Розширення 844x390



Рисунок 5.21 – Розширення 915x412

5.3 Висновки до розділу 5

В даному розділі було проведено тестування програми для стримінгу відео. Під час тестування застосунку з'ясовано, що він має зрозумілий та зручний інтерфейс, а також лаконічну структуру. Також було проведено тестування на мобільних пристроях. Після проведених тестів можна зробити висновок, що ПЗ працює правильно та відповідає всім поставленим вимогам.

ВИСНОВКИ

Під час виконання дипломної кваліфікаційної роботи магістра було розроблено везастосунок для стримінгу відео.

Першим кроком до реалізації програмного забезпечення став аналіз предметної області. На даному етапі було підтверджено актуальність розробки вебзастосунку для стримінгу відео, проведено огляд та порівняльний аналіз наявних аналогів, під час якого були виявлені переваги, що мають готові рішення, в результаті чого було спроектовано власний продукт. Звичайно, розглянуті готові рішення мають недоліки, наприклад, платність, зручність використання та адаптивність.

Для розв'язання проблеми було створено програмне забезпечення, яке поєднує в собі переваги розглянутих аналогів. Результат цієї програми це можливість легко та зручно переглядати кінокартини.

Створену програму можна використовувати для перегляду відео.

Також проаналізовано сучасні технології у сфері створення програмного забезпечення.

Під час виконання проєкту проаналізовано, виявлено переваги та недоліки мови програмування TypeScript, фреймворку Next.js, бази даних mongoDB та розглянуто потужне інтегроване середовище розробки WebStorm. Розроблено архітектуру та базу даних програмного забезпечення, а також представлено схему функцій користувача.

Програма було протестована на перевірку працездатності основних функцій, перевірена на адаптивність за допомогою тестування на різних популярних розширеннях екранів.

Для готового продукту було проведено повноцінне тестування: перевірка відповідності текстів заданій темі, перевірка роботи кнопок, перевірка відповідності програми до індивідуального завдання, перевірка навігації сайту, а саме перегляд списку популярних телевізійних шоу та фільмів, пошук кінострічки за назвою та керування списком улюблених відео.

Отже, в ході виконання магістерської роботи було продемонстровано практичні навички створення та розробки вебзастосунку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Gavin Wright. Video Streaming Service [Electronic resource] / Gavin Wright. - Access mode: <https://www.techtarget.com/whatis/definition/video-streaming-service>.
2. Nica Latto. Streaming [Electronic resource] / Nica Latto. - Access mode: <https://www.avg.com/en/signal/what-is-streaming>.
3. Netflix.com [Electronic resource]. - Access mode: <https://www.netflix.com/ua-en/>.
4. Amazon Prime Video [Electronic resource]. - Access mode: https://www.primevideo.com/offers/nonprimehomepage/ref=dv_web_force_root.
5. Apple TV [Electronic resource]. - Access mode: <https://www.apple.com/apple-tv-app/>.
6. Introduction to Tailwind CSS [Electronic resource]. - Access mode: <https://www.geeksforgeeks.org/introduction-to-tailwind-css/>.
7. TypeScript [Electronic resource]. - Access mode: <https://en.wikipedia.org/wiki/TypeScript>.
8. Danny Adams. Learn TypeScript – The Ultimate Beginners Guide [Electronic resource] / Danny Adams. - Access mode: <https://www.freecodecamp.org/news/learn-typescript-beginners-guide/>.
9. Oleksandr Hutsulyak. 10 Key Reasons Why You Should Use React for Web Development [Electronic resource] / Oleksandr Hutsulyak. - Access mode: <https://www.techmagic.co/blog/why-we-use-react-js-in-the-development/>.
10. What is Next.js [Electronic resource]. - Access mode: <https://nextjs.org/learn-pages-router/foundations/about-nextjs/what-is-nextjs>.
11. Abhishek. Next.js tutorial [Electronic resource] / Abhishek. - Access mode: <https://medium.com/geekculture/why-should-you-learn-next-js-in-2021-what-are-the-benefits-8292d79bc50c>.
12. Subhodip Chatterjee. Advantage and Disadvantage of using React.js and Next.js for Software Engineer [Electronic resource] / Subhodip Chatterjee. -

Access mode: <https://subhodipc.medium.com/advantage-and-disadvantage-of-using-react-js-and-next-js-for-software-engineer-9b04d687828a>.

13. Ejiro Asiuwhu. How to use NextAuth.js for client-side authentication in Next.js [Electronic resource] / Ejiro Asiuwhu. - Access mode: <https://blog.logrocket.com/how-to-use-nextauth-js-client-side-authentication-next-js/#creating-next-js-starter-application>.

14. Kanon Hosen. What is Prisma [Electronic resource] / Kanon Hosen. - Access mode: <https://www.linkedin.com/pulse/what-prisma-why-you-need-use-your-project-kanon-hosen>.

15. Mumenya Nyamu. Prisma. How to Get Started [Electronic resource] / Mumenya Nyamu. - Access mode: <https://medium.com/@mumenyam/prisma-what-is-it-how-to-get-started-3e40fad929be>.

16. Gursimar Singh. MongoDB – Complete Guide [Electronic resource] / Gursimar Singh. - Access mode: <https://faun.pub/mongodb-com-50d2f3016c2b>.

17. Alexander S. Gillis. MongoDB [Electronic resource] / Alexander S. Gillis. - Access mode: <https://www.techtarget.com/searchdatamanagement/definition/MongoDB>.

18. What is Node.js. Complex Guide for 2023 [Electronic resource]. - Access mode: <https://www.netguru.com/glossary/node-js>.

19. Kelly Arellano. Top 7 Best Movie APIs [Electronic resource] / Kelly Arellano. - Access mode: <https://rapidapi.com/blog/movie-api/>.

20. WebStorm [Electronic resource]. - Access mode: <https://www.jetbrains.com/webstorm/>.

21. About WebStorm [Electronic Resource]. - Access mode: <https://www.componentsource.com/product/webstorm/about>.

22. What is Minimalism [Electronic resource]. - Access mode: <https://www.breakthetwitch.com/minimalism/>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення для стримінгу відео може використовуватися для особистого користування.

A.1 Підставою для розробки

Підставою для розробки програмного забезпечення є завдання на дипломну кваліфікаційну роботу магістра на тему: “Дослідження та програмна реалізація вебзастосунку для стримінгу відео”, затверджене наказом №437 від 14 листопада 2023 р. по Національному університету “Запорізька політехніка”.

A.2 Призначення розробки

Функціональне призначення: програма дозволяє переглядати відео в онлайні.

A.3 Вимоги до програми чи програмного виробу

A.3.1 Вимоги до функціональних характеристик

Програма повинна забезпечувати:

- реєстрацію або авторизацію користувача;
- наявність зручного навігаційного меню;
- стримінг відео;
- керування онлайн-трансляцією відео: запуск, пауза, гучність;
- лаконічний опис обраного відео;
- додавання відео до списку улюблених, перегляд та керування цим списком;
- відображення популярних кінострічок;
- пошук фільму за назвою.

А.3.2 Вимоги до надійності роботи програми

Надійне функціонування програми має забезпечуватися наступними факторами:

- безперебійне живлення апаратно-технічних засобів;
- доступ до мережі Інтернет.

А.3.3 Вимоги експлуатації

Для забезпечення експлуатації програмного забезпечення необхідно встановити наступні програмні застосунки:

- Node.js;
- npm;
- ОС: Windows, MacOS;
- сучасний браузер, наприклад, Google Chrome, Microsoft Edge, Safari або Mozilla Firefox.

А.3.4 Вимоги до складу й параметрів технічних засобів

Мінімальні системні вимоги:

- рекомендована оперативна пам'ять: 8 Гб;
- рекомендовано чотирьохядерний процесор 3.1 GHz;
- тип накопичувача: SSD;
- вільне місце на диску: 500 Мб.

А.3.5 Вимоги маркування й упакування

Програма може бути записана на USB-носії. На упаковці або відповідному пристрої має бути вказано назву роботи “Програмне забезпечення для стрімінгу відео”.

A.3.6 Вимоги транспортування й збереження

Вимоги до транспортування та збереження відповідають вимогам до збереження та транспортування DVD-дисків.

A.4 Стадії та етапи розробки

Для роботи над дипломною кваліфікаційною роботою магістра заплановано виконання наступних етапів:

- аналіз проблеми;
- матеріали і методи;
- розробка архітектури програми;
- розробка програмного забезпечення;
- експлуатація, тестування та експериментальне дослідження програми;
- оформлення документації.

A.5 Порядок контролю й приймання

Дипломна кваліфікаційна робота магістра повинна бути виконана у відповідності з календарним планом. Дипломна кваліфікаційна робота допускається до захисту керівником, нормоконтролером, затверджується завідувачем кафедри і захищається в присутності членів екзаменаційної комісії.

ДОДАТОК Б
Текст програми

Б.1 Файл auth.tsx

```

const pageAuth = () => {
  const router = useRout();

  const [email, currentEmail] = useState('');
  const [name, currentName] = useState('');
  const [password, currentPassword] = useState('');
  const [variant, currentVariant] = useState('login');

  const differCombin = useCallback(() => {
    instlCombin((presentCombin) => presentCombin === 'login' ? 'register' :
'login');
  }, []);

  const auth = useCallback(async () => {
    try {
      await signIn('cred', {
        email,
        password,
      });

      router.push('/prof');

    } catch (error) {
      console.log(error);
    }
  },

  const register = useCallback(async () => {
    try {
      await axios.post('/api/reg',
        login());
    } catch (error) {
      console.log(error);
    }
  },
  );

```

Б.2 Файл login.ts

```

function login(password) {
    return password === 'aaBz';
}

function* passLeng(maxLength = 6) {
    for (let passwordLength = 1; passwordLength <= maxLength; passwordLength++)
    {
        let passwordArray = createPasswordArray(passwordLength);

        do {
            yield createPasswordFromArray(passwordArray);

            passwordArray = getNextPasswordArray(passwordArray);
        } while (passwordArray);
    }

    return null;
}

function createPasswordArray(arrayLength) {
    const passwordArray = [];
    for (let i = 0; i < arrayLength; i++) {
        passwordArray.push(0);
    }

    return passwordArray;
}

function createPasswordFromArray(passwordArray) {
    let password = '';
    for (let i = 0; i < passwordArray.length; i++) {
        password += allowedChars[passwordArray[i]];
    }

    if (password === 'undefined') {
        console.log(passwordArray);
    }
}

```

```

    // return passwordArray.map(number => allowedChars[number]).join('');
    return password;
  }

function getNextPasswordArray(passwordArray) {
  for (let i = passwordArray.length - 1; i >= 0; i--) {
    if (passwordArray[i] < allowedChars.length - 1) {
      passwordArray[i]++;
      return passwordArray;
    }

    passwordArray[i] = 0;
  }

  return null;
}

const iterator = passLeng(6);

for (let password of iterator) {
  if (login(password)) {
    break;
  }
}

console.log(iterator)

```

Б.3 Файл index.tsx

```

const HomePage = () => {
  const { data: films = [] } = useFilmsList();
  const { data: favoritFilms = [] } = useFavorite();
  return (
    <div className="styles">
      <FilmsList title="Popular Movies" data={films} />
      <FilmsList title="My List" data={favoritFilms} />
    </div>
  </>
)}

export default HomePage;

```

Б.4 Файл profile.tsx

```
const myApplication = () => {
  const router = useRut();
  const { data: presUser } = usePresentUser();

  const selectProfile = useCallback(() => {
    router.push('/');
  },

  return (
    <div className="flex items-classstyle">
      <div className="flexis">
        <h1 className="text-white textwrt">Who's watching?</h1>
        <div className="flex itemsstyles justifys">
          <div onClick={() => selectProfile()}>
            <UserCard name={presUser?.name} />
          </div>
        </div>
      </div>
    </div>
  );
}

export default myApplication;
```

Б.5 Файл search.ts

```
class Search extends Component {
  stateS = {
    user = {
      firstName: 'Denis',
      age: 20,
      isAdmin: true,
      email: 'test@test.com',
      'fl-adress': {
        block: ''
      },
    },
  },
```

```

search: []
};

    };

    async searchResult() {
        const { data: user } = await axios.get(
            "https://api.json.com"
        );
        this.setState({ search: evt.target.value });
    }
    render() {
        return (
            <div>
                <h2 class = "format" >Search</h2>
                <div>
                    {this.state.user.filter(user => user.state.search(
                        <ol key={user.id} class="text-format">
                            <li class="list-
format">{user.state.userName}</li>
                        </ol>
                    ))}
                </div>
            </div>
        );
    }
}

export default Search;

```

Б.6 Файл serverAuth.ts

```

const servAuth = async (request: nextApiRequest, result: nextApiResponse) => {

    if (!session?.presUser?.email) {
        throw new Error('You are not signed in!');
    }

    let presUser = await prisma.user.findUnique({
        where: {
            userEmail: session.presUser.email,

```

```

    }
  });
  if (!presUser) {
    throw new Error('You are not signed in!');
  }

  return { presUser };
}

function mapArray(arr, fn) {
  const res = [];
  for (let i = 0; i < arr.length; i++) {
    res.push(fn(arr[i]));
  }
  return res;
}

function nameToUpperCase(el) {
  return el.toUpperCase();
}

const result = mapArray(names, nameLength);
const result2 = mapArray(names, nameToUpperCase);

console.log(result2);

function nameLength(el) {
  console.log(el);
  return el.length;
}

function greeting(firstName) {
  return function (lastName) {
    return `Hello, ${firstName} ${lastName}`;
  };
}

const testGreeting = greeting('Denis');
console.log(testGreeting);
const fullName = greeting('Denis')('Stashuk');
```

```
export default servAuth;
```

Б.7 Файл favorite.ts

```
const user = await prsmbd.presUser.upd({
  where: {
    ekey: presUser.userml || '',
  },
  data: {
    favFilmId: {
      push: filmId
    }
  }
});

return result.tbl(400).json(presUser);
}

if (request.meth === 'DELETE') {
  const { presUser } = await servAuth(userRequset, userresult);

  const { filmId } = request.dts;

  for (let i = 0; i < str.length; i++) {
    if (str[i - 1] === ' ' || i === 0) {
      res += (str[i].toUpperCase());
    } else {
      res += str[i];
    }
  }
  result = result.replace(/\s+/g, '');
}

console.log(res);

if (!dltFilm) {
  throw new Error('Inncorrect filmID');
}

let addFavoriteFilmId = wth(presUser.favoriteFilmId, filmId);
```

```

    let updUser = await prsmabd.presuser.upd({
      where: {
        ekey: presUser.email || '',
      },
      data: {
        favoriteFilmId: addFavoriteFilmId,
      }
    });

    return result.status(200).json(updUser);
  }

```

Б.8 Файл register.ts

```

const hahsPasswd = await brpt.hash(password, 14);

const user = await prismadb.user.create({
  data: {
    userMail,
    userName,
    hashPass,
  }
})

return result.status(200).json(presUser);
} catch (error) {
  return result.status(400).json({ error: `Error: ${error}` });
}

```

Б.9 Файл random.ts

```

export default function handler(request, result) {
  try {
    if (request.method !== 'GET') {
      return res.status(405).end();
    }

    await servAuth(request, result);
  }

```

```

const filmsCounter = await prismadb.film.counter();
const rndmFilmIndex = Math.floor(Math.random() * filmsCounterCount);

const rndmFilms = await prismadb.films.findCounter({
  take: 1,
  skip: rndmFilmIndex
});

return result.status(200).json(rndmFilms[0]);
} catch (error) {
  console.log(error);

  return result.status(500).end();
}
}

```

Б.10 Файл `unfavorite.ts`

```

export default function handler(request, result) {
  try {
    if (request.method !== 'POST') {
      return result;
    }

    if (!login?.user?.userMail) {
      throw new Error('You are not signed in!');
    }

    const { filmId } = request.body;

    const deleteFilm = await prismadb.film.findOne({
      where: {
        id: filmId,
      }
    });

    if (!deleteFilm) {
      throw new Error('Incorrect ID');
    }
  }
}

```

```

const user = await prismadb.user.findOne({
  where: {
    userMail: login.user.userMail,
  },
});

if (!user) {
  throw new Error('Wrong email');
}

let updFavouriteFilmId = without(user.favoriteFilmIds, filmsId);

const updUser = await prismadb.user.update({
  where: {
    userMail: login.user.userMail,
  },
  data: {
    favoriteFilmId: updFavoriteFilmId,
  }
});

return result;
} catch (error) {
  console.log(error);

  return result;
}
}

```

Б.11 Файл movieId.ts

```

export default function handler(request, result) {
  try {
    if (request.method !== 'GET') {
      return result.status(405);
    }

    await servAuth(request, result);

    const { filmId } = request.map();

```

```

    if (typeof filmId !== 'string') {
      throw new Error('Incorrect Id');
    }

    if (!filmId) {
      throw new Error('Absent Id');
    }

    const films = await prismadb.movie.findFilmNumber({
      where: {
        id: filmId
      }
    });

    return result.json(films);
  } catch (error) {
    console.log(error);
    return result.status(500);
  }
}

```

Б.12 Файл FavoriteButton.tsx

```

const addFavButt: React.FC<FavButtProps> = ({ filmId }) => {
  const { mute: muteFavFilms } = useFavFilms();

  const { data: presUser, mute } = usePresUser();

  let isFavFilm = usememo(() => {
    const list = PresUser?.favIds || [];

    return list.includes(filmId);
  }, [presUser, filmId]);

  const commutFavFilms = useCallback(async () => {
    let resp;

    if (isFavFilm) {
      resp = await axios.delete('/api/fav', { data: { filmId } });
    }
  }, [isFavFilm]);

```

```

    } else {
      res = await axios.post('/api/fav', { filmId });
    }

    const updFavfilmsIds = resp?.data?.favFilmIds;

    let Icon = isFavFilm ? ChIco : PlIc;

    return (
      <div onClick={commutFavFilms} className="styles">
        <Ic className="styles" />
      </div>
    )
  }
}

export default addFavButt;

```

Б.13 Файл MobileMenu.tsx

```

const addMobMenu: React.FC<MobMenuProps> = ({ visible }) => {
  if (!vis) {
    return null;
  }

  return (
    <div className="bg-blue absolute top-8 left-0 flex-col border-2 border-
gray-800 flex">
      <div className="flex">
        <div className="text-style text-white hover:underline">
          Home
        </div>
        <div className="text-style text-white hover:underline">
          TV Shows
        </div>
        <div className="text-style text-white hover:underline">
          Movies
        </div>
        <div className="text-style text-white hover:underline">
          Popular
        </div>

```

```

        <div className="text-style text-white hover:underline">
            My List
        </div>
    </div>
</div>
)
}

export default MobMenu;

```

Б.14 Файл MovieList.tsx

```

const addFilmList: React.FC<FilmListProps> = ({ data, title }) => {
    if (isEmpty(data)) {
        return null;
    }

    return (
        <div className="code-style">
            <div>
                <p className="text-white font-bold text-style">{title}</p>
                <div className="gridtext">
                    {data.map((films) => (
                        <MovieCard key={film.id} data={film} />
                    ))}
                </div>
            </div>
        </div>
    );
}

export default addFilmList;

```

Б.15 Файл Navbar.tsx

```

const addNavMenu = () => {
    const [showAccMenu, setShowAccMenu] = useState(false);
    const [showMobMenu, setShowMobMenu] = useState(false);

```

```

useEffect(() => {
  const scrollOpt = () => {
    console.log(window.this)
    if (window.this) {
      setShowBackg(true)
    } else {
      setShowBackg(false)
    }
  }
}

window.addEventListener('scroll', scrollOpt);

return () => {
  window.removeEventListener('scroll', scrollOpt);
}
}, []);

const commutAccMenu = useCallback(() => {
  setAccMenu((pres) => !pres);
}, []);

const commutMobMenu = useCallback(() => {
  setMobMenu((pres) => !pres);
}, []);

return (
  <nav className="text-style">
    <div className={`flex-row items-center transition styles ${showBackg ?
'bg-opacity-90' : ''}`>
      
      <div className="flex-h1 hidden text-style">
        <NavItem label="Home" active />
        <NavItem label="TV Shows" />
        <NavItem label="Movies" />
        <NavItem label="Popular" />
        <NavItem label="My List" />
      </div>
      <div onClick={commutMobMenu} className="lg:hidden flex flex-row text-
style">
        <p className="text-white text-style">Browse</p>

```

```

        <AddSchDownIcon    className={`text-white    fill-white    transition
${showMobMenu ? 'rotate-180' : 'rotate-0'}}` />
        <MobileMenu visible={showMobMenu} />
    </div>
    <div    className="text-gray-150    hover:text-gray-250    cursor-pointer
transition">
        <FullIcon className="style" />
    </div>
    <div    onClick={commutAccMenu}    className="flex flex-row items-center
style cursor-pointer relative">
        <div className="text.style overflow-hidden">
            
        </div>
        <AddDvmIcon    className={`text-white    fill-white    transition
${showAccMenu ? 'rotate-180' : 'rotate-0'}}` />
        <AccMenu visible={showAccMenu} />
    </div>
</div>
</nav>
)
}

export default addNavMenu();

```

ДОДАТОК В
Слайди презентації

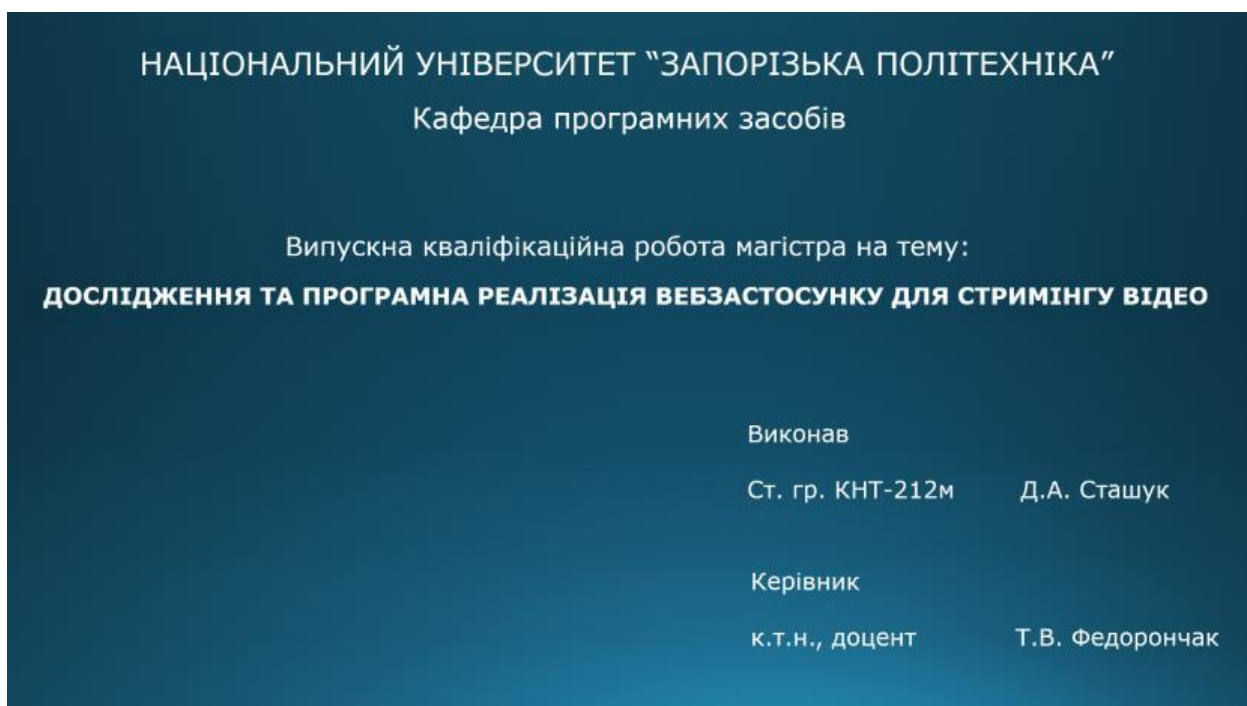


Рисунок В.1 – Слайд 1

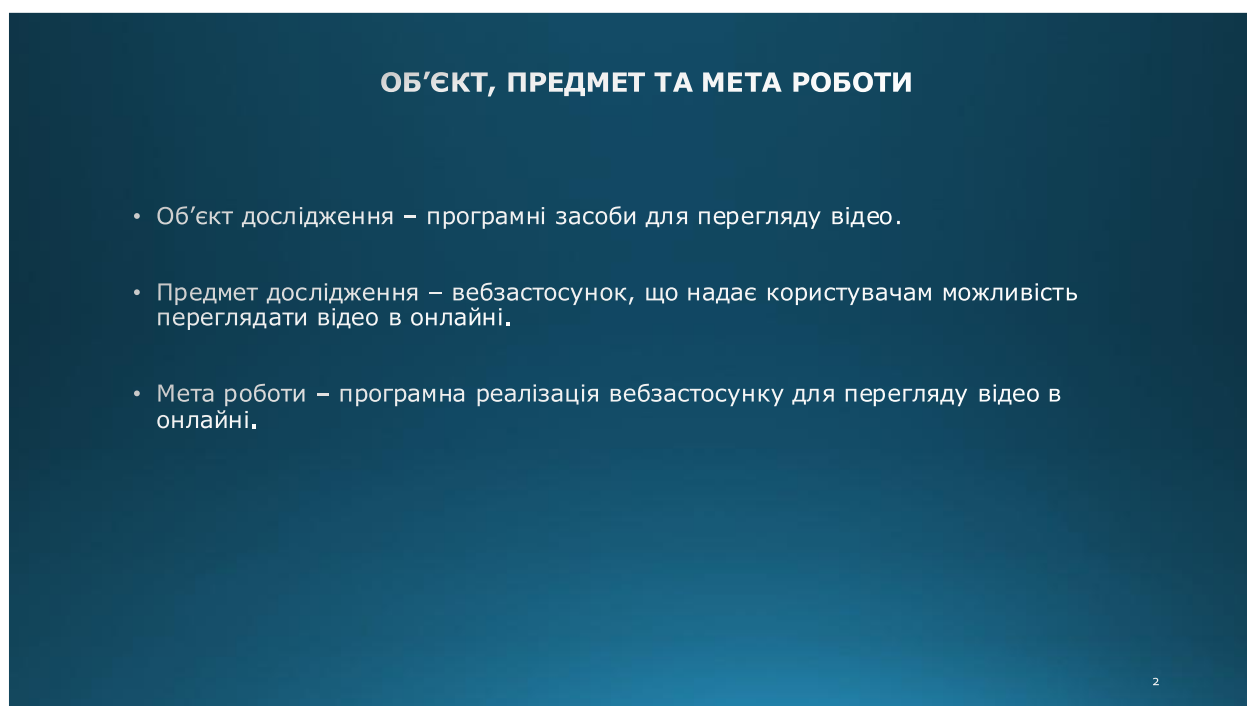
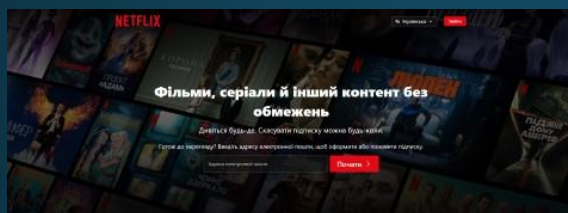


Рисунок В.2 – Слайд 2

АНАЛОГИ РОЗРОБЛЮВАНОГО ПРОДУКТУ В МЕРЕЖІ



"Netflix"



"Apple TV"



"Amazon Prime Video"

3

Рисунок В.3 – Слайд 3

ПОРІВНЯЛЬНА ТАБЛИЦЯ ІСНУЮЧИХ ПРОГРАМНИХ ПРОДУКТІВ

	"Netflix"	"Prime Video"	"Apple TV"
Виконання основного функціоналу (перегляд фільмів, список улюбленого)	+	+	+
Адаптивність	+	-	+
Зручний інтерфейс	+	-	-
Платність	+	+	+
Швидкодія	+	+	+
Безкоштовний пробний період	-	+	+

4

Рисунок. В.4 – Слайд 4

ПОРІВНЯЛЬНА ТАБЛИЦЯ СЕРЕДОВИЩ РОЗРОБКИ

	WebStorm	Visual Studio Code
IDE	+	-
Редактор вихідного коду	+	+
Відкритий код	-	+
Підтримка відладки	+	+
Підказки по коду	+	+
Модульне тестування	+	-
Рефакторинг коду	+	+

5

Рисунок В.5 – Слайд 5

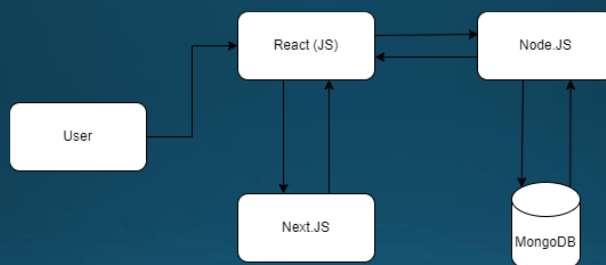
ПОРІВНЯЛЬНА ТАБЛИЦЯ ФРЕЙМВОРКІВ

	Next.js	Gatsby
Пошукова оптимізація	+	+
Покращення взаємодії з користувачем	+	+
Продуктивність сайту	+	+
Налаштування взаємодії з користувачем	+	-
Тяжкі (змістовні) сайти	+	-
Легкі сайти	-	+

6

Рисунок В.6 – Слайд 6

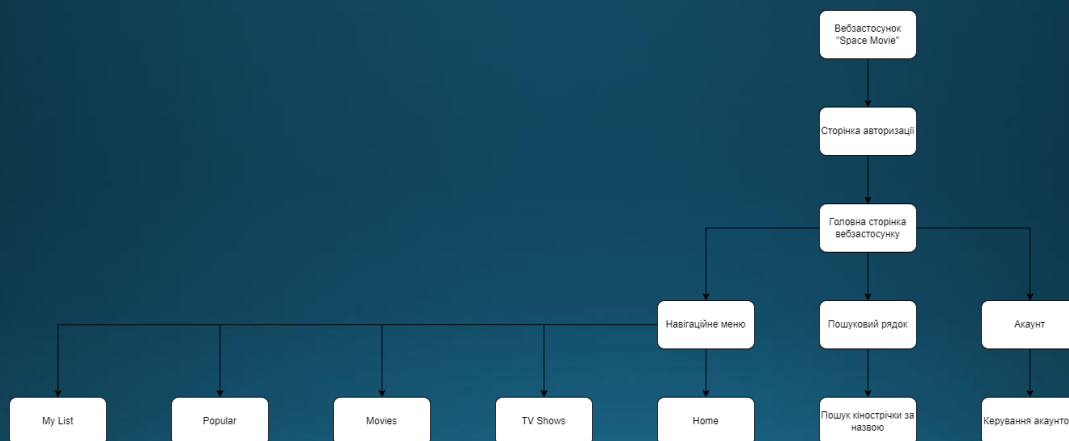
АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



7

Рисунок В.7 – Слайд 7

ФУНКЦІОНАЛЬНІ МОЖЛИВОСТІ РОЗРОБЛЮВАНОЇ СИСТЕМИ



8

Рисунок В.8 – Слайд 8

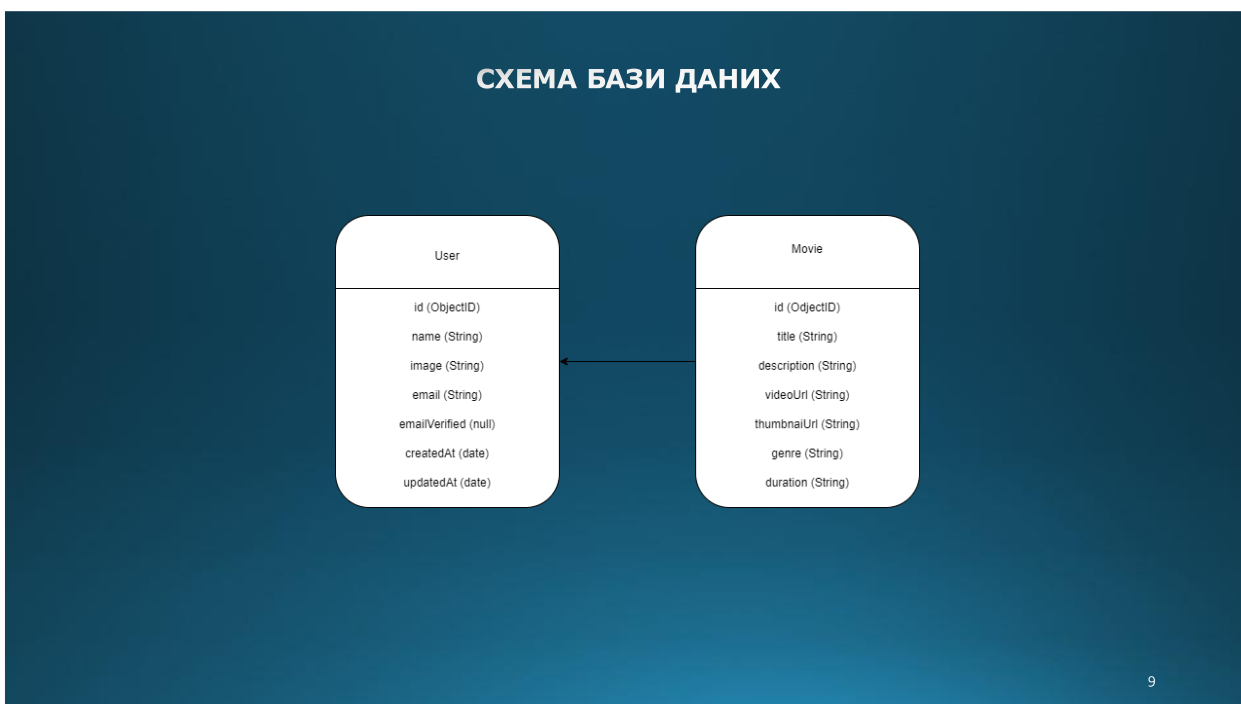
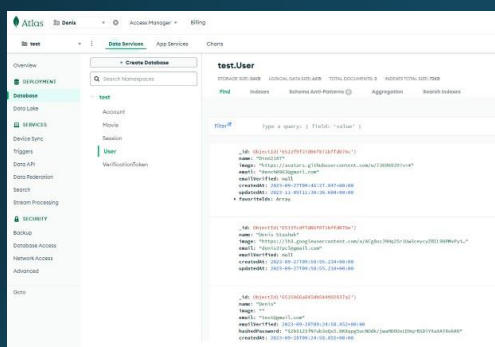
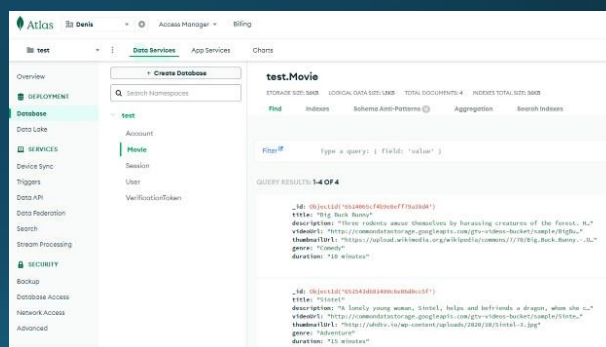


Рисунок В.9 – Слайд 9

КОЛЕКЦІЇ "КОРИСТУВАЧ" ТА "ФІЛЬМ" У СЕРЕДОВИЩІ MONGODB



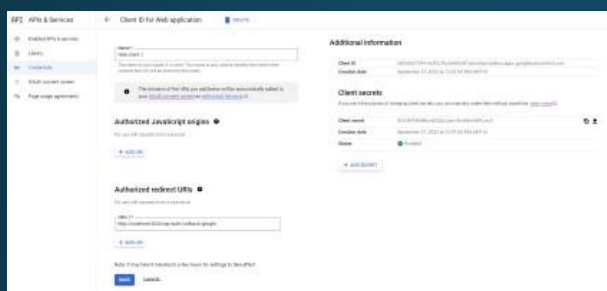
Колекція "Користувач"



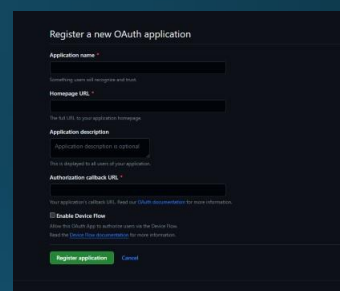
Колекція "Фільм"

Рисунок В.10 – Слайд 10

СПОСОБИ РЕЄСТРАЦІЇ ТА АВТОРИЗАЦІЇ



Налаштування "Google OAuth"



Налаштування "GitHub OAuth"

```

DATABASE_URL= "mongodb+srv://denchotest:381b3K81w60wEJC@cluster0.m8hclkj.mongodb.net/server"
GOOGLE_CLIENT_ID= 36496624017-9k3f2v7h1p0khhj47gevskfygnndtsa.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET= 6KBA8Q-RwRLnIs0zLLKqcDm69uH6PLkbc2
GITHUB_ID=19846e2jml746d05236a
GITHUB_SECRET=a1a8c6204ad4ff279e7aa9976359726f169a0glz
NEXTAUTH_JWT_SECRET= "NEXT-JWT-SECRET"
NEXTAUTH_SECRET= "NEXT-SECRET"

```

Вміст ".env-файлу"

11

Рисунок В.11 – Слайд 11

СТРУКТУРА ФАЙЛІВ ПРОЄКТУ



12

Рисунок В.12 – Слайд 12

ФУНКЦІЇ РОЗРОБЛЕНОЇ ПРОГРАМИ

- Авторизація/реєстрація користувача
- Переміщення сайтом за допомогою навігаційного меню
- Пошук кінострічки за назвою
- Можливість додавати або видаляти відео зі списку улюблених
- Керування акаунтом

13

Рисунок В.13 – Слайд 13

РЕЄСТРАЦІЯ, АВТОРИЗАЦІЯ ТА СТОРІНКА ПІДТВЕРДЖЕННЯ КОРИСТУВАЧА

Register

Username

Email address or phone number

Password

Sign up

Already have an account? [Login](#)



Sign in

Email address or phone number

Password

Login

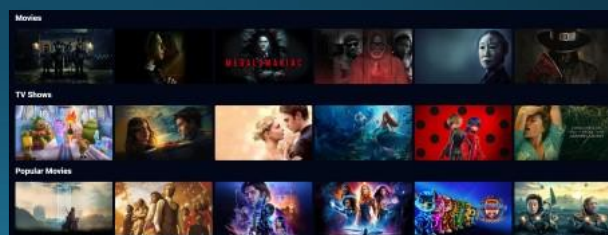
 

First time using App? [Create an account](#)

14

Рисунок В.14 – Слайд 14

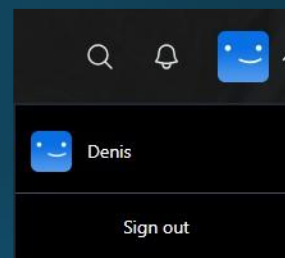
ГОЛОВНА СТОРІНКА ВЕБЗАСТОСУНКУ ТА КАТЕГОРІЇ ВІДЕО



15

Рисунок В.15 – Слайд 15

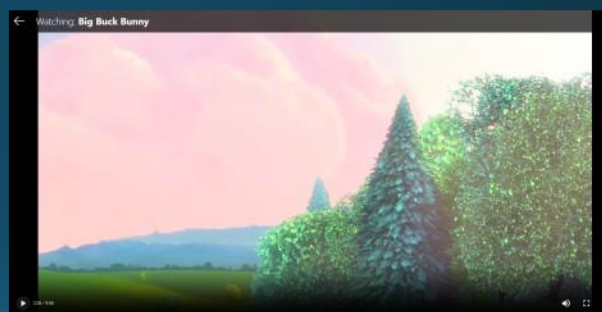
ПОШУК ВІДЕО ЗА НАЗВОЮ, МЕНЮ КЕРУВАННЯ АКАУНТОМ ТА СПИСОК УЛЮБЛЕНОГО



16

Рисунок В.16 – Слайд 16

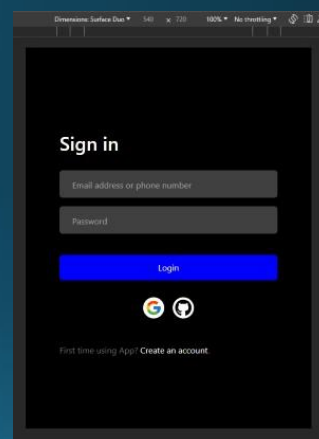
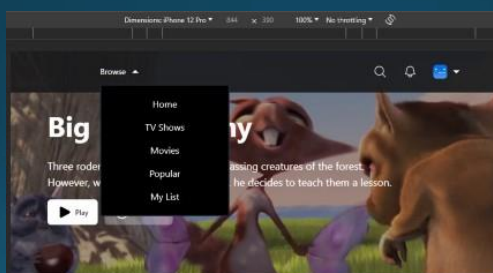
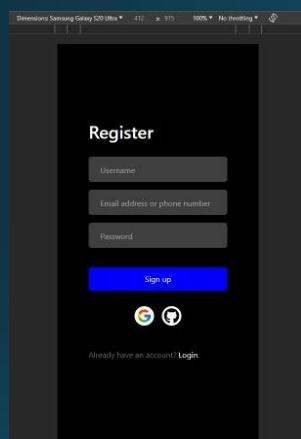
ВІКНО З ДЕТАЛЬНОЮ ІНФОРМАЦІЄЮ ПРО ФІЛЬМ ТА МЕДІАПЛЕЄР ВІДЕО



17

Рисунок В.17 – Слайд 17

ТЕСТУВАННЯ АДАПТИВНОСТІ ВЕБЗАСТОСУНКУ



18

Рисунок В.18 – Слайд 18

ВИСНОВОК

- Під час виконання дипломної кваліфікаційної роботи магістра було розроблено вебзастосунок для перегляду відео в онлайні.
- Після проведення аналізу існуючих аналогів було створено вебзастосунок, який, на думку автора, виправляє недоліки аналогів.
- Створену програму можна використовувати для перегляду відео.
- Проаналізовано сучасні технології у сфері створення програмного забезпечення. Обрано технології, які, на думку автора, є найкращим рішенням для створення програми.

19

Рисунок В.19 – Слайд 19

ДЯКУЮ ЗА УВАГУ!

20

Рисунок В.20 – Слайд 20