Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» _____ 2024 р

Пояснювальна записка

до кваліфікаційної роботи
бакалавра

на тему: «МОДЕЛЬ ОНЛАЙН КІНОТЕАТРУ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ SSR КОМПОНЕНТІВ»

Захищено на засіданні
Атестаційної комісії № 42
протокол № __ від __.06.2024 р.
Оцінка _____ / _____
Голова Атестаційної комісії
_____ **СКОБ Ю. О.**

Виконав:
студент 4 курсу, групи КІ–41
Галузь знань: 12 – Інформаційні технології
Спеціальність: 123 – Комп'ютерна інженерія.
КОЛГАНОВ Кирило Євгенович_____

Керівник: к.т.н., доцент ЗВО
КОРОБЧИНСЬКИЙ Кирил Петрович

Рецензент: к.т.н., доцент ЗВО, в.о. завідувача кафедри теоретичної та прикладної інформатики
МЕНЯЙЛОВ Євгеній Сергійович____

Харків – 2024

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і двох додатків. Загальний обсяг роботи складає 68 сторінок, із яких 50 сторінок з основної частини з 47 рисунками, 7 найменуваннями списку використаних джерел та трьома додатками.

Мета даної дипломної роботи полягає у дослідженні ринку стрімінгових сервісів та розробці веб-додатку з використанням технології SSR. Для досягнення цієї мети в роботі проведено аналіз сучасного ринку стрімінгових платформ, розглянуто теоретичні аспекти створення веб-додатків з використанням SSR, а також особливості застосування технології Next.js для реалізації таких рішень.

Об'єкт дослідження – процес розробки веб-додатку для перегляду серіалів з використанням технології NextJs SSR (Server-Side Rendering). Дослідження охоплює всі аспекти цього процесу, включаючи аналіз сучасних тенденцій у веб-розробці, вивчення можливостей та особливостей технології NextJs SSR, а також розробку та реалізацію функціональності вебсайту.

Предмет дослідження – технологія Server-Side Rendering(SSR) у веб-розробці стрімінг-платформи.

Область застосування – розроблений програмний продукт може широко використовуватися в сфері розваг та розвитку, маркетингу та реклами, підтримці українських студій озвучування та людей постраждалих від російської агресії проти України.

Ключові слова: API, SSR, DOM, SEO, CDN, DDoS, Next.Js, Laravel, React, Javascript, ORM, Бази даних, Фреймворк, Паттерн, Програмування.

ABSTRACT

The explanatory note for the bachelor's qualification work consists of an introduction, four chapters, conclusions, a list of references and two appendices. The total volume of work is 68 pages, of which 50 pages of the main part with 47 figures, 7 titles in the list of references, and three appendices.

The goal of this thesis is to study the market of streaming services and develop a web application using SSR (Server-Side Rendering) technology. To achieve this goal, the thesis includes an analysis of the current market of streaming platforms, examines the theoretical aspects of creating web applications using SSR, and explores the specific features of applying Next.js technology for the implementation of such solutions.

Object of the research: The process of developing a web application for watching TV series using the NextJs SSR (Server-Side Rendering) technology. The research encompasses all aspects of this process, including the analysis of modern trends in web development, the study of the capabilities and features of the NextJs SSR technology, as well as the development and implementation of the website's functionality.

Subject of the research: The analysis and implementation of optimal strategies for using Server-Side Rendering (SSR) technology in the web development of a streaming platform.

Scope of application: The developed software product can be widely used in the fields of entertainment and development, marketing and advertising, supporting Ukrainian dubbing studios and helping people affected by russian aggression against Ukraine.

Keywords: API, SSR, DOM, SEO, CDN, DDoS, Next.Js, Laravel, React, Javascript, ORM, Databases, Framework, Pattern, Programming.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

SSR	–	Server-Side Rendering
БД	–	База даних
DOM	–	Document Object Model
SEO	–	Search Engine Optimization
CDN	–	Content Delivery Network
DDoS	–	(distributed) denial-of-service attack
ORM	–	Object-Relational Mapping
URL	–	Uniform Resource Locator
SPA	–	Single Page Application
UML	–	Unified Modeling Language
API	–	Application Programming Interface

ВСТУП

Розвиток інтернет-технологій спричинив значне зростання попиту на стрімінгові сервіси, що дозволяють користувачам отримувати доступ до медіа-контенту у режимі реального часу.

Актуальність роботи. Сучасні стрімінгові платформи стають невід'ємною частиною цифрового життя, пропонуючи широкий спектр послуг від відео- та аудіоконтенту до живих трансляцій. Цей тренд зумовлює необхідність створення ефективних і надійних веб-додатків, які здатні забезпечити високу якість обслуговування та задовольнити потреби користувачів. Одним з ключових аспектів розробки сучасних веб-додатків є використання технології відображення даних на стороні сервера (Server-Side Rendering, SSR). Ця технологія дозволяє значно покращити продуктивність додатків, скоротити час завантаження сторінок і забезпечити кращий користувацький досвід, особливо на мобільних пристроях та в умовах обмеженої пропускної здатності мережі.

Метою дослідження є аналіз ринку стрімінгових сервісів та розробці веб-додатку з використанням технології SSR. Для досягнення цієї мети в роботі проведено аналіз сучасного ринку стрімінгових платформ, розглянуто теоретичні аспекти створення веб-додатків з використанням SSR, а також особливості застосування технології Next.js для реалізації таких рішень.

Об’єкт дослідження – процес розробки веб-додатку для перегляду серіалів з використанням технології NextJs SSR (Server-Side Rendering). Дослідження охоплює всі аспекти цього процесу, включаючи аналіз сучасних тенденцій у веб-розробці, вивчення можливостей та особливостей технології NextJs SSR, а також розробку та реалізацію функціональності вебсайту.

Методи дослідження: аналіз літератури та вторинних джерел інформації, методи системного аналізу, емпіричні методи, методи тестування, методи моделювання, методи розгортання та апробації.

Предмет дослідження – технологія Server-Side Rendering(SSR) у веб-розробці стрімінг-платформи.

Завдання дослідження

1. Провести аналіз наукової літератури.
2. Провести аналіз конкурентів і пошук статей відповідно до вибраної теми.
3. Побудувати модель веб-додатку і бази даних.
4. Розробити веб-додаток і базу даних на базі моделі.
5. Використати SSR компоненти в проекті.

РОЗДІЛ 1
АНАЛІЗ РИНКУ СТРІМІНГОВИХ СЕРВІСІВ ТА ТЕОРЕТИЧНИХ АСПЕКТІВ РОЗРОБКИ ВЕБ-ДОДАТКУ З ВІДОБРАЖЕННЯМ ДАНИХ НА СТОРОНІ СЕРВЕРА(SSR)

1.1 Аналіз ринку стрімінгових сервісів

За даними сайту fortunebusinessinsights [1] на 2023-2024 рік глобальний ринок стрімінгового відео оцінювався в 554,33 мільярда доларів США, і за прогнозами, до 2032 року зросте до 2486,51 мільярда доларів США, що вказує на прогнозовану річну зростання на рівні 17,8% (з 671,89 мільярда доларів США у 2024 до 2032). У 2023 році ринкова вартість в Північній Америці склала 217,23 мільярда доларів США. Звіт охоплює різноманітні платформи для передачі програм та контенту, представлені компаніями, такими як Netflix Inc., Hulu LLC, IBM Corporation, Amazon.com Inc. та інші. Ці організації пропонують різноманітні стрімінгові канали та платформи, включаючи HBO Max, Amazon Prime Video, Crackle, Paramount Plus, Disney Plus, Acorn TV та інші.

Популярність соціальних медіа-платформ і зростання доступу до Інтернету стимулюють ростову динаміку глобального ринку. Наприклад, на травень 2021 року платформа соціальних мереж Meta (Facebook Inc.) мала понад 2,40 мільярда користувачів. Крім того, соціальні медіа-платформи, такі як WhatsApp і YouTube, мають понад 1,00 мільярд користувачів кожна.

Також, зростання обсягу відеоданих по всьому світу внаслідок значного попиту на контент з високою роздільною здатністю сприяє розвитку ринку. Крім того, учасники ринку активно розробляють передові платформи для задоволення зростаючих потреб в освітньому секторі.

Наразі стрімінгові сервіси стали основним джерелом розваг для багатьох користувачів по всьому світу. Популярність кожного сервісу може варіюватися в залежності від регіону, а також від доступності контенту та ціноутворення. Змагання між стрімінговими сервісами в основному базується

на якості та різноманітності контенту. Кожен сервіс старається вирізнятися своїм унікальним контентом, таким як оригінальні серіали, фільми та ексклюзивні шоу. Стрімінгові сервіси постійно вдосконалюють свої технології для забезпечення кращого користувацького досвіду, що включає в себе вдосконалення відео- та аудіоякості, розробку інтерфейсів користувача, персоналізацію рекомендацій та інші інновації.

1.2 Концепція веб-сайтів з використанням SSR

Концепція веб-сайтів з використання SSR в основному полягає в тому, що сторінки додатку генеруються на сервері кожного разу, коли користувач запитує їх, і відправляються клієнту як готовий HTML-код. Це відбувається до передачі сторінки браузеру.

Порівняємо HTML-код двох додатків, перший з яких – написаний на чистому React, другий – на Next.Js (який безпосередньо використовує SSR)

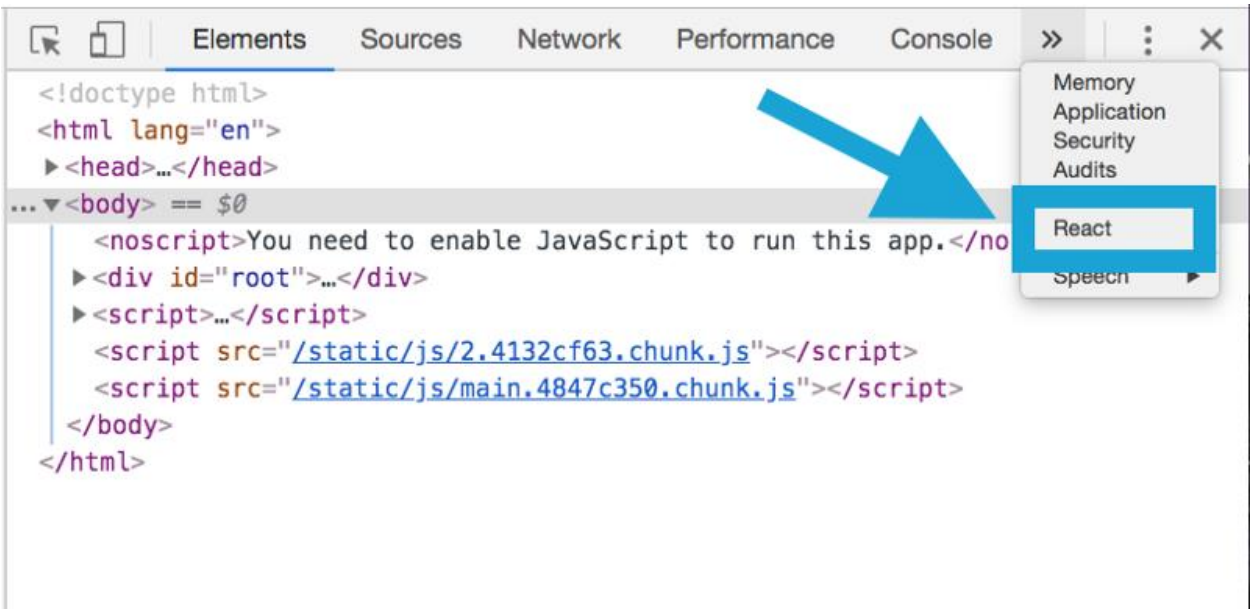

Рисунок 1.1 – Приклад HTML-коду React

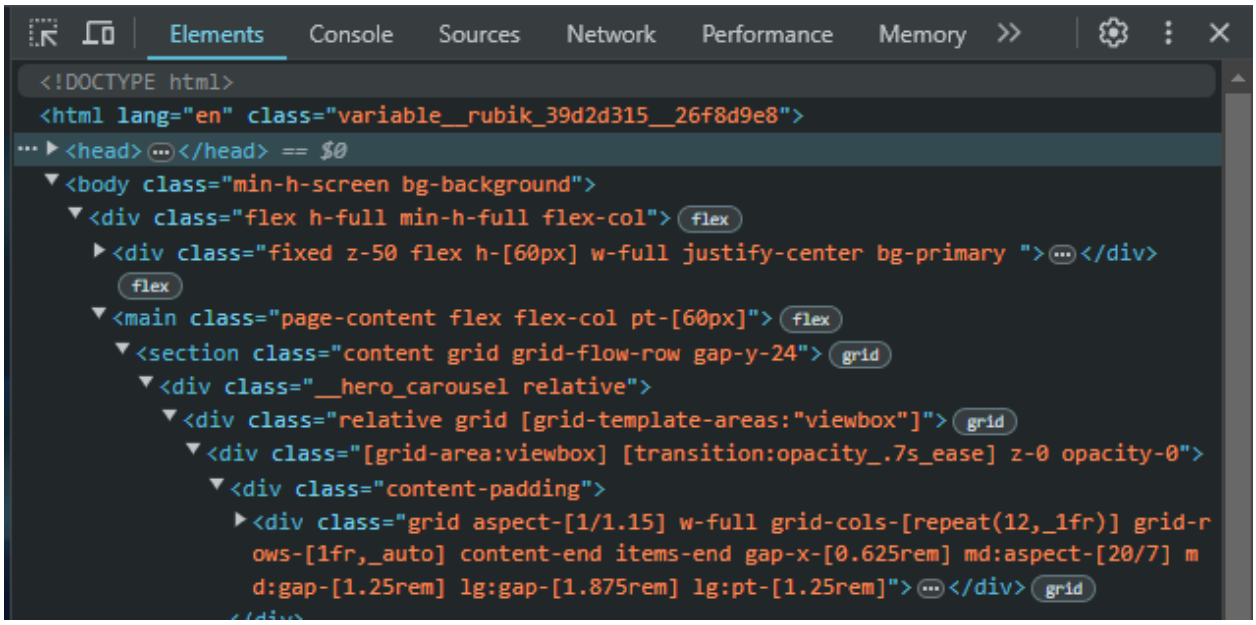

Рисунок 1.2 – Приклад HTML-коду Next.Js

На рис. 1.1 можна побачити один-єдиний div-тег який зазвичай називається “кореневим”, в межах якого буде керуватися React DOM (Document Object Model). В той час, як на рис. 1.2 бачимо повноцінний HTML-код з усіма класами і стилями.

Примітка: в режимі розробника "F12" можна вивчити всі наявні теги та вузли React DOM-дерева, проте вони не відтворюються у джерелі коду сторінки "Ctrl + U", що має значення для оптимізації веб-додатку з точки зору SEO.

Найважливішими плюсами SSR – є пошукова оптимізація (SEO) та користувацький досвід. SSR може покращити час завантаження сторінок для користувачів, оскільки вони отримують готовий HTML-код негайно, що є особливо важливим для перших відвідувачів, які можуть отримати загальний огляд контенту швидше.

SEO оптимізація є однією з найважливіших частин розробки бізнес проектів, робот індексатор читає вміст сторінки і шукає ключові слова, які відповідно будуть порівнюватись з запитами користувача, і в залежності від збіжностей надавати результат. У прикладі з React робот-індексатор нічого не буде бачити на сторінці, вона буде “пуста”, і щоб це виправити, розробник має

робити багато додаткової роботи, що не є не можливим, але є більш важким в плані реалізації. В цьому плані SSR дуже сильно виграє, пошуковий бот бачить весь скелет вашої сторінки, тим самим SEO оптимізація додатку стає зручніша, його використання допомагає забезпечити, що ваш вміст індексується швидше і краще.

Не менш важливим плюсом є **архітектура**, яка дозволяє обробляти частину логіки та генерувати сторінки на сервері, що може полегшити управління станом, роутінгом та іншими аспектами веб-розробки та спрощення процесу **кешування** статичних сторінок, що може поліпшити продуктивність веб-сайту.

Хоча SSR має свої переваги, він також може бути складним для впровадження в порівнянні з іншими підходами, такими як **Client-Side Rendering (CSR)**. SSR вимагає більшої обробки на сервері та зазвичай вимагає наявності серверної частини (наприклад, Node.js), що може бути витратним з точки зору ресурсів і складним для масштабування. Однак, залежно від вимог проекту, SSR може бути вигідним вибором для покращення продуктивності та SEO веб-сайту.

1.3 Огляд технології SSR

Server-Side Rendering (SSR) – це технологічний підхід, в якому веб-сторінка формується на сервері замість клієнтського браузера. Після повного відрендерення сторінки на сервері, вона передається клієнту, включаючи пакет JavaScript, який забезпечує інтерактивність за допомогою фреймворку для одно сторінкових додатків (**SPA**). Серверний візуалізаційний процес передбачає рендеринг вмісту сторінки безпосередньо на сервері. Це означає, що при переході на сторінку користувач отримує вже повністю сформовану HTML-структуру, яка включає весь необхідний вміст для перегляду веб-сайту. Такий підхід дозволяє уникнути затримок, пов'язаних з очікуванням завантаження JavaScript або CSS-файлів, що призводить до швидшого відображення сторінки користувачем.

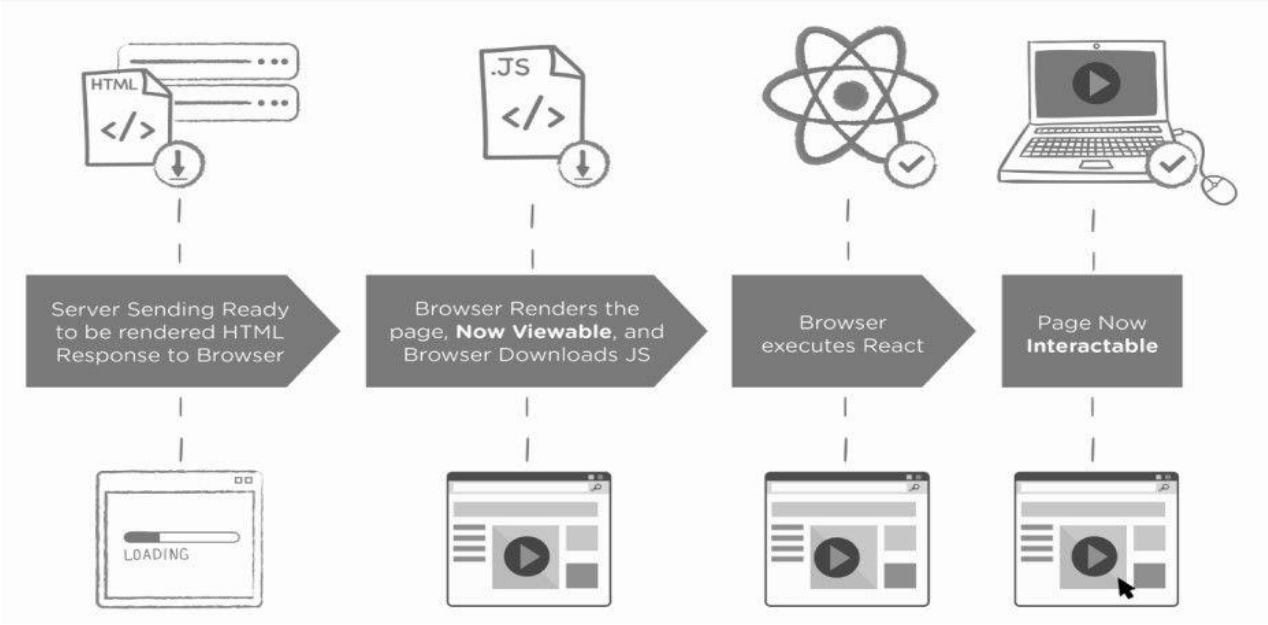

Рисунок 1.3 – Принцип роботи SSR

Правильне використання техніки відтворення на стороні сервера (**SSR**) може значно покращити ефективність веб-додатків, оскільки вона дозволяє генерувати контент перед його відправленням клієнту. Це сприяє скороченню часу завантаження сторінок і поліпшує їхню оптимізацію для пошукових систем, оскільки контент надсилається раніше, ніж у випадку, коли б він був згенерований після завантаження сторінки.

Технологія відтворення на стороні сервера широко використовується у відомих фреймворках JavaScript, таких як Angular і React, а також у менш відомих, але просунутих альтернативах, наприклад, Preact.

Веб-сайти, які використовують **SSR**, охоплюють різноманітні галузі, включаючи соціальні мережі, такі як Facebook і Twitter, а також платформи електронної комерції, такі як Walmart і Amazon.

Використання відтворення на стороні сервера (SSR) зазвичай сприяє покращенню **SEO**, оскільки пошукові системи можуть індексувати сторінки до їх завантаження у браузері. Це може позитивно вплинути на швидкість сканування та загальний рейтинг веб-сайту, що є дуже важливим для замовника. Перевагою **SSR** є також полегшення процесу індексації вмісту для

пошукових систем, що дозволяє їм ефективніше виявляти та аналізувати різноманітні показники, такі як відмови та час на сайті. Крім того ця технологія може бути корисною для веб-сайтів з динамічним вмістом, наприклад, блогів, або тих, які прагнуть зберегти деякі елементи прихованими до їх прокручування до певної частини сторінки, наприклад, галереї зображень.

Усі ці переваги роблять **SSR** важливим компонентом для розробки стрімінгових веб-додатків, забезпечуючи швидку, ефективну та оптимізовану роботу платформи.

1.4 Особливості використання Next.Js у розробці SSR

В якості інструменту розробки веб-додатку був вибран популярний фреймворк React з відкритим кодом, розроблений та підтримуваний компанією Vercel, який спрощує процес розробки – **Next.Js**. Його висока продуктивність, розширені можливості та зручний інтерфейс для розробників допомогли здобути велику популярність у веб-розробницькій спільноті. Основні можливості Next.js включають в себе рендеринг на серверній стороні (**SSR**), статичну генерацію сайту (**SSG**), автоматичний розподіл коду та легко зрозумілу систему маршрутизації, базовану на файловій структурі. Ці можливості дозволяють розробникам створювати веб-додатки, що працюють швидко, оптимізовані для пошукової оптимізації (**SEO**), з мінімально необхідними налаштуваннями.

Деякі ключові функції, які роблять Next.js кращим вибором для додатку:

- Серверний рендерінг (SSR): однією з ключових можливостей Next.js є SSR, що дозволяє відображати React компоненти на сервері, а не на клієнті. Це призводить до швидшого завантаження сторінок та покращує SEO, оскільки пошукові системи можуть індексувати повністю згенерований HTML.

- Статичне сторінкове генерування (SSG) : Next.js пропонує SSG, який попередньо генерує сторінки під час збірки. Цей підхід створює статичні HTML-файли для кожної сторінки, що забезпечує дуже швидке завантаження сторінок і спрощує розгортання на мережах доставки контенту (CDN).

- Клієнтська маршрутизація: має вбудовану систему клієнтської маршрутизації. Ви можете створювати динамічні односторінкові додатки (SPA), не налаштовуючи складну маршрутизацію.

- Автоматичне розділення коду: автоматично розділяє JavaScript код на менші частини, щоб користувачі завантажували лише необхідний код для поточної сторінки. Це оптимізує час завантаження та продуктивність.

- Файлова маршрутизація: Next.js спрощує маршрутизацію, дозволяючи розробникам створювати сторінки за допомогою файлової структури. Просто створюємо JavaScript файл у каталозі "pages", і Next.js автоматично керуватиме маршрутизацією за нас.

Цей фреймворк є потужним інструментом для розробки веб-додатків з підтримкою SSR. Його простота використання та переваги в швидкості та SEO роблять його привабливим вибором для багатьох розробників. Однак варто ретельно розглянути його недоліки, такі як складність динамічних сторінок, більший обсяг коду та вживати заходи для оптимізації продуктивності.

Висновки за розділом 1

Розділ надає докладний огляд глобального ринку стрімінгових сервісів, вказуючи на його значний ріст та прогнозовані перспективи розвитку. За наведеними даними, ринок стрімінгового відео наразі перебуває на стадії активного росту з потенціалом досягнення значної вартості у майбутньому. Такий аналіз відображає важливість цієї теми для подальших досліджень.

Була розглянута концепція веб-сайтів з використанням SSR (Server-Side Rendering) та її переваги. Зазначається, що SSR сприяє покращенню часу завантаження сторінок для користувачів і позитивно впливає на їхній користувацький досвід, зокрема швидкість першого завантаження сторінок.

Представлено технічний огляд технології SSR та особливостей використання Next.js у її реалізації. Цей огляд допомагає краще зрозуміти, як SSR працює на практиці та чому Next.js є привабливим вибором для веб-

розробників, особливо з урахуванням його продуктивності та оптимізації для SEO.

Розділ роботи надає важливі відомості щодо стану ринку стрімінгових сервісів, а також технічні аспекти використання SSR та Next.js, що є ключовими для подальшого розуміння та розвитку дослідження.

РОЗДІЛ 2
ПРОЕКТУВАННЯ АРХІТЕКТУРИ ДОДАТКУ

2.1 Вибір архітектурних паттернів

1) Паттерн “Мікросервісної архітектури” розбиває систему на невеликі, незалежні сервіси, кожен з яких відповідає за певну функціональність. Наприклад, один сервіс може бути відповідальний за каталог, інший - за акаунт користувача тощо. Це полегшує розгортання, масштабування та підтримку системи.

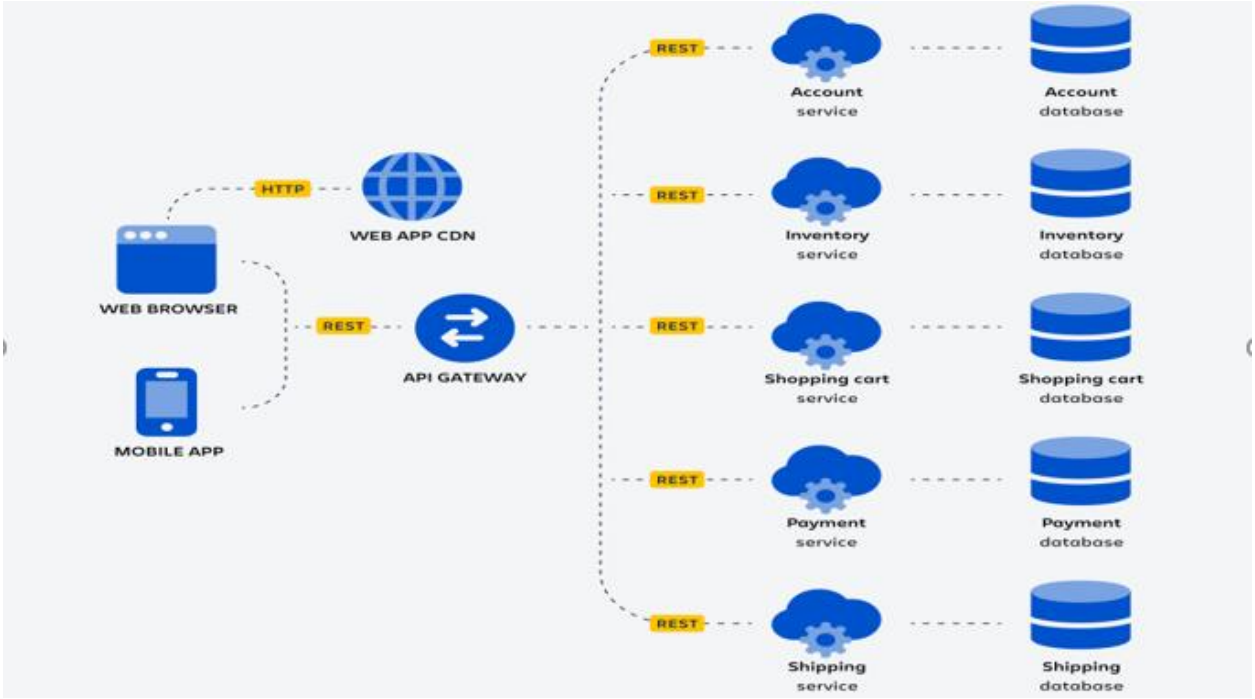

Рисунок 2.1 –Мікросервісна архітектура

Програми взаємодіють з мікросервісами через REST API, які надаються кожним з них. Додатки використовують API-шлюз для доступу до API мікросервісів, що дозволяє заміняти одні мікросервіси іншими, що мають той самий API.

Кожен мікросервіс складається з сервісу та бази даних. Сервіси взаємодіють з REST API, виконують бізнес-логіку та зберігають дані у відповідних базах.

2) Паттерн Кешування який може значно підвищити швидкодію та продуктивність стрімінгового сервісу, особливо при великому обсязі запитів на контент. Використання шаблону кешування дозволяє зменшити навантаження на сервери та забезпечити більш швидку відповідь на запити користувачів. У Next.js, кешування використовується для збереження результатів попередніх запитів на сервері чи в браузері. Основні види кешування в Next.js включають статичне кешування, серверне кешування та кешування на клієнтському боці.

Статичне кешування (Static Caching): використовується для статичних сторінок, які не змінюються між запитами. Під час першого запиту на таку сторінку Next.js генерує HTML-код та зберігає його для подальших запитів. Це дозволяє серверу відповідати на запити на такі сторінки без необхідності генерації HTML-коду заново.

Серверне кешування (Server-Side Caching): використовується для динамічних сторінок, які можуть змінюватися між запитами. Next.js може кешувати результати запитів на сервері протягом певного часу життя кешу, щоб уникнути повторної генерації відповідей на однакові запити.

Кешування на клієнтському боці (Client-Side Caching): кешування на клієнтському боці використовується для збереження даних на браузері клієнта. Це може бути корисним для кешування даних, які не змінюються часто, наприклад, результати попередніх запитів або стан додатка. Next.js може використовувати різні методи для збереження даних на клієнтському боці, такі як Local Storage, Session Storage або Cookies.

3) Ще один паттерн який варто розглянути – це “Шина подій” (Event Bus), він дозволяє різним компонентам системи спілкуватися між собою, надсилаючи події до спільної шини. Наприклад, коли новий трейлер серіалу додається до системи, він може надсилати подію до шини, яку слухають інші компоненти, такі як сервіс рекомендацій або сторінка з пошуком, щоб оновити свої дані.

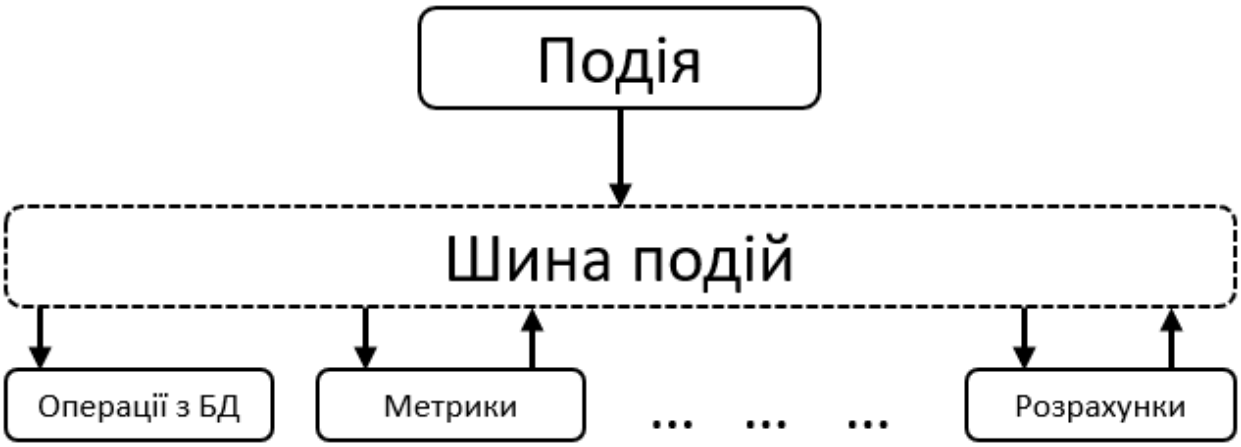

Рисунок 2.2 – Принцип роботи **Event Bus** паттерну

Було розглянуто кілька ключових архітектурних паттернів, які є важливими для веб-додатку, що розроблюється. Починаючи з "Мікросервісної архітектури", було наголошено на перевагах розбиття системи на невеликі, незалежні сервіси, які полегшують розгортання, масштабування та підтримку. Далі, шаблон кешування в Next.js визначається як інструмент для підвищення продуктивності та швидкодії за допомогою збереження результатів попередніх запитів. Нарешті, обговорюється "Шина подій", яка дозволяє різним компонентам системи спілкуватися між собою шляхом надсилання подій до спільної шини. Ці паттерни важливі для побудови високопродуктивного, масштабованого та зручного для підтримки веб-додатку. Застосування цих паттернів допоможе покращити якість та ефективність розроблених систем.

2.2 Структура додатку

Структуру клієнтської частини у проекті Next.Js має кілька основних папок і файлів, які відповідають за різні аспекти роботи додатку, для їх огляду дивитись рис. 2.3. Ця структура допомагає організувати код додатку в логічний та зрозумілий спосіб, полегшуючи розробку та підтримку проекту. Кожна папка та файл мають своє призначення, що допомагає зберігати порядок та уникати хаосу в коді.

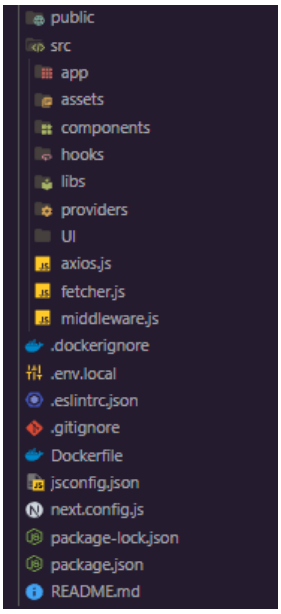

Рисунок 2.3 –Структура клієнтської частини веб-додатку

Папка «app» містить всі сторінки додатку. Next.js використовує файлову систему для створення маршрутизації, де кожен файл або папка автоматично стає маршрутом. Папка «public» містить статичні файли, які доступні за прямими URL. Наприклад, файл public/image.png буде доступний за URL «http://domain.com/image.png». Папка «assets» містить в собі допоміжні файли, які будуть використовуватись у додатку, наприклад svg картинки тощо. Папка components зазвичай містить повторно використовувані React-компоненти, які можуть бути використані на різних сторінках додатку. Папка «hooks» містить користувацькі React-хуки, які допомагають організувати та повторно використовувати логіку в компонентах. Папка «libs» (або її ще називають «utils») містить утиліти та бібліотеки, які використовуються у додатку. Це можуть бути функції для роботи з API, хелпери для обробки даних тощо. Папка «providers» використовується для зберігання компонентів провайдерів, які забезпечують контекстні дані або функціональність, доступні у всьому додатку. Це може включати провайдери для управління станом, аутентифікацією, темами та іншими глобальними аспектами додатку. Файл конфігурації «next.config.js» визначає налаштування різних параметри додатку, такі як налаштування Webpack, маршрутизація, середовище

виконання тощо. Файл «.env.local» використовується для зберігання конфіденційної інформації, такої як ключі API. Файл керування залежностями та скриптами проекту «package.json» містить інформацію про пакети, які використовуються у проекті, та скрипти для виконання різних задач, таких як запуск сервера розробки, збірка проекту тощо.

Структуру бекенду можна представити у вигляді діаграми, яка показує методи обробки даних які використовуються в контролерах. Кожен блок діаграми – відповідний метод на сервері.

Рисунок 2.4 – Бекенд структура веб-додатку

Робота з базою даних: ці методи включають запити на створення, читання, оновлення та видалення (CRUD) даних та оптимізацію запитів для підвищення продуктивності.

Авторизація та автентифікація: методи, що відповідають за перевірку особи користувача, управління сесіями, видачу та перевірку токенів, а також забезпечення безпеки даних користувачів.

Аналітика: методи збору та аналізу даних для отримання статистичної інформації, створення звітів, а також відстеження поведінки користувачів та продуктивності системи.

Операції з відео: методи, що займаються обробкою відеофайлів, такими як завантаження, зберігання, конвертація форматів, стрімінг та відтворення.

Контент: методи управління текстовим та мультимедійним контентом, включаючи створення, редагування, видалення та відображення контенту на веб-сторінках.

Система коментарів: методи, що дозволяють користувачам залишати коментарі, а також включають модерацію, сортування та відображення коментарів на відповідних сторінках.

Діаграма допомагає візуалізувати, як взаємодіють різні частини бекенд-системи, забезпечуючи ефективний та надійний обробіток даних для користувачів та адміністраторів системи.

2.3 Проектування бази даних

Проектування бази даних (БД) – це важливий етап розробки будь-якої веб платформи. Розглянемо ER-діаграму бази даних і деякі основні таблиці стрімінгового сервісу, що розроблюється.

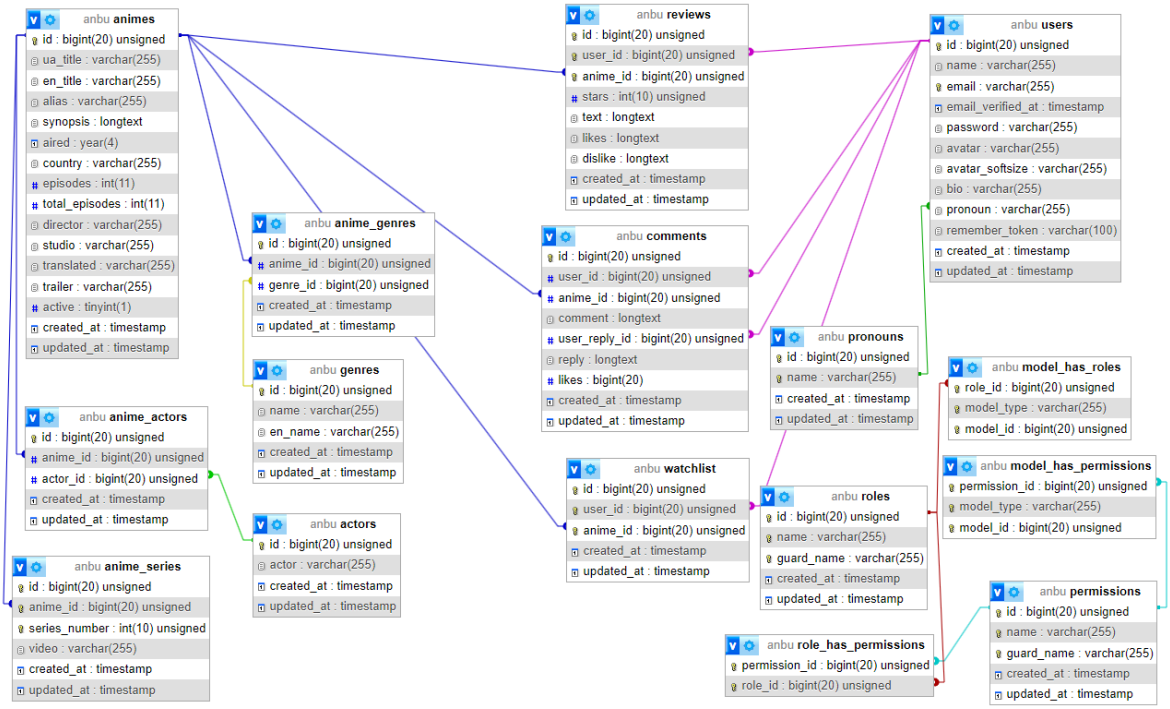

Рисунок 2.5 – ER-діаграма бази даних

Таблиця «animes» використовується для визначення основних даних про конкретний серіал. Структура таблиці:

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково	Дія
1	id	bigint(20)		UNSIGNED	Ні	*Немає*		AUTO_INCREMENT	Змінити Знищити Більше
2	ua_title	varchar(255)	utf8mb4_unicode_ci		Так	NULL			Змінити Знищити Більше
3	en_title	varchar(255)	utf8mb4_unicode_ci		Так	NULL			Змінити Знищити Більше
4	alias	varchar(255)	utf8mb4_unicode_ci		Так	NULL			Змінити Знищити Більше
5	synopsis	longtext	utf8mb4_unicode_ci		Так	NULL			Змінити Знищити Більше
6	aired	year(4)			Так	NULL			Змінити Знищити Більше
7	country	varchar(255)	utf8mb4_unicode_ci		Так	NULL			Змінити Знищити Більше
8	episodes	int(11)			Так	NULL			Змінити Знищити Більше
9	total_episodes	int(11)			Так	NULL			Змінити Знищити Більше
10	director	varchar(255)	utf8mb4_unicode_ci		Так	NULL			Змінити Знищити Більше
11	studio	varchar(255)	utf8mb4_unicode_ci		Так	NULL			Змінити Знищити Більше
12	translated	varchar(255)	utf8mb4_unicode_ci		Так	NULL			Змінити Знищити Більше
13	trailer	varchar(255)	utf8mb4_unicode_ci		Так	NULL			Змінити Знищити Більше
14	active	tinyint(1)			Ні	1			Змінити Знищити Більше
15	created_at	timestamp			Так	NULL			Змінити Знищити Більше
16	updated_at	timestamp			Так	NULL			Змінити Знищити Більше

Рисунок 2.6 – Структура таблиці «animes»

Таблиця «useres». Використовується для зберігання інформації про користувачів. У кожного користувача є своя роль – таблиця «model_has_roles» у кожної ролі є свої права - таблиця «roles_has_roles», також у кожного

користувача можуть бути свої права (наприклад: у кожного користувача є право залишати коментарі на сайті, адміністратор може забрати ці права і користувач отримує так званий статус “muted”.

	#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково	Дія
☐	1	id 🔑	bigint(20)		UNSIGNED	Ні	*Немає*		AUTO_INCREMENT	Змінити Знищити Більше
☐	2	name	varchar(255)	utf8mb4_unicode_ci		Ні	*Немає*			Змінити Знищити Більше
☐	3	email 🔑	varchar(255)	utf8mb4_unicode_ci		Ні	*Немає*			Змінити Знищити Більше
☐	4	email_verified_at	timestamp			Так	*NULL*			Змінити Знищити Більше
☐	5	password	varchar(255)	utf8mb4_unicode_ci		Ні	*Немає*			Змінити Знищити Більше
☐	6	avatar	varchar(255)	utf8mb4_unicode_ci		Так	*NULL*			Змінити Знищити Більше
☐	7	avatar_softsize	varchar(255)	utf8mb4_unicode_ci		Так	*NULL*			Змінити Знищити Більше
☐	8	bio	varchar(255)	utf8mb4_unicode_ci		Так	*NULL*			Змінити Знищити Більше
☐	9	pronoun 🔑	varchar(255)	utf8mb4_unicode_ci		Так	*NULL*			Змінити Знищити Більше
☐	10	remember_token	varchar(100)	utf8mb4_unicode_ci		Так	*NULL*			Змінити Знищити Більше
☐	11	created_at	timestamp			Так	*NULL*			Змінити Знищити Більше
☐	12	updated_at	timestamp			Так	*NULL*			Змінити Знищити Більше

Рисунок 2.7 – Структура таблиці «users»

Варто згадати про розділення можливостей між звичайним користувачем та адміністратором, які можна побачити на UML-діаграмі на рис. 2.5.

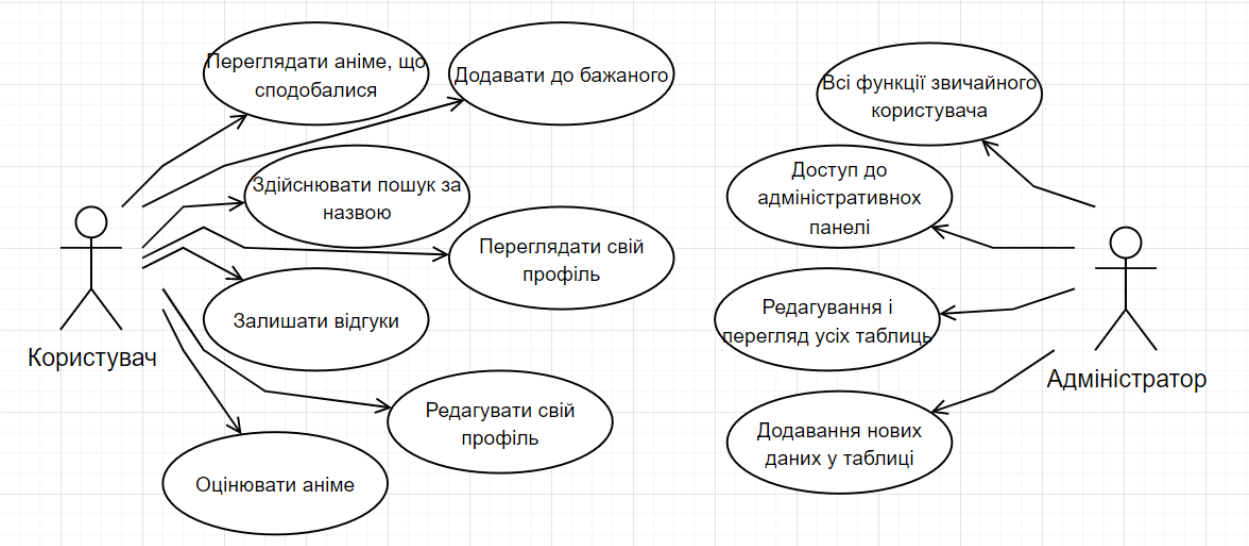

Рисунок 2.8 – UML-діаграма розподілу дозволів

На рис. 2.8 можна побачити розподіл дозволів між звичайним користувачем та адміністратором. Наприклад, користувач, на відміну від

адміністратора, не можу додавати новий контент до таблиць, окрім коментарів або інформації про його профіль.

Висновки за розділом 2

Розділ надає огляд ключових архітектурних паттернів для веб-додатку. Починаючи з Мікросервісної архітектури, відзначено переваги розбиття системи на невеликі, незалежні сервіси, що сприяє полегшенню розгортання, масштабування та підтримки. Далі, увага звернута на шаблон кешування в Next.js, який сприяє підвищенню продуктивності та швидкодії, зокрема за рахунок збереження результатів попередніх запитів. Нарешті, обговорюється Шина подій, яка дозволяє компонентам системи взаємодіяти, надсилаючи події до спільної шини. Використання цих паттернів сприяє побудові високопродуктивного, масштабованого та легко підтримуваного веб-додатку, що відповідає вимогам сучасних ринкових умов. Їх застосування може значно підвищити якість та ефективність розроблюваних систем.

Додаток на базі Next.js має структуру, що допомагає організувати код в логічний та зрозумілий спосіб. Розділення на основні папки та файли допомагає підтримувати порядок та уникати хаосу в коді, що робить розробку та підтримку проекту більш ефективною.

Проектування бази даних також відіграє ключову роль у розробці стрімінгового сервісу. ER-діаграма та UML-діаграма дозволяють чітко визначити структуру даних та розподіл функціональностей між звичайними користувачами та адміністраторами. Це допомагає забезпечити ефективне зберігання та обробку даних, а також забезпечує безпеку та доступність системи для користувачів з різними ролями.

У цілому, обрані архітектурні паттерни та структура додатку, разом з проектуванням бази даних, забезпечують ефективність, гнучкість та надійність стрімінгового сервісу. Ці аспекти роботи допомагають забезпечити задоволення потреб користувачів та успішну експлуатацію системи.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ДОДАТКУ З ВИКОРИСТАННЯМ NEXT.JS SSR

3.1 Налаштування середовища розробки

Підготовка середовища розробки відіграє важливу роль у забезпеченні продуктивної роботи над проектом. У випадку стрімінгового сервісу, де швидкість, ефективність та надійність грають критичну роль, важливо мати належно налаштоване середовище розробки. Використання Visual Studio Code (VS Code) який може значно полегшити і зробити робочий процес більш зручним завдяки своїм розширенням та налаштуванням.

Для ефективної роботи з JavaScript та React у Visual Studio Code, важливо використовувати різні інструменти та розширення, які допомагають підтримувати код у чистому, читабельному та оптимізованому стані. Серед таких інструментів можна виділити Prettier, ESLint та ES7+ React/Redux/React-Native snippets. Нижче наведено детальний опис кожного з цих розширень.

Prettier — це інструмент для автоматичного форматування коду. Він підтримує різні мови програмування, включаючи JavaScript, TypeScript, CSS, HTML та інші. Prettier забезпечує єдиний стиль коду у всьому проекті, що спрощує його читання та підтримку.

ESLint — це інструмент для аналізу статичного коду, який допомагає виявляти та виправляти проблеми у коді JavaScript. Він дозволяє налаштовувати правила перевірки коду відповідно до вимог проекту, що сприяє підтримці високої якості коду.

ES7+ React/Redux/React-Native snippets — це розширення для VSCode, яке надає набір корисних шорткатів для швидкого написання коду на React, Redux та React Native. Це розширення значно прискорює процес розробки, зменшуючи кількість рутинних дій та покращуючи продуктивність.

Використання цих розширень значно покращує процес розробки, роблячи його більш організованим, структурованим та ефективним. Ці інструменти забезпечують єдиний стиль коду, автоматичне виявлення та

виправлення помилок, а також прискорюють написання шаблонного коду, що робить їх незамінними для сучасного розробника.

3.2 Реалізація серверної частини за допомогою Laravel

Laravel — це популярний PHP-фреймворк, який використовується для створення веб-додатків. Він надає елегантний синтаксис та інструменти, що сприяють швидкому та ефективному розробленню серверної частини додатків. В цьому розділі буде розглянуто основні етапи створення серверної частини за допомогою Laravel.

Встановлення Laravel. Для початку роботи з Laravel необхідно встановити Composer — менеджер пакетів для PHP, після чого створити новий проект Laravel.

1) Встановлення Composer:

```
    php -r "copy('https://getcomposer.org/installer',
'composer-setup.php');"
    php -r "if (hash_file('sha384', 'composer-setup.php')
===
'55ce33a94b98f0b90044f4b00c1a3075d3ea9b07ff50d38c1351f3f24e2
0d775e9f78e10b7d98cb4e74b3c9c90f8e1d0') { echo 'Installer
verified'; } else { echo 'Installer corrupt';
unlink('composer-setup.php'); } echo PHP_EOL;"
    php composer-setup.php
    php -r "unlink('composer-setup.php');"
```

Рисунок 3.1 – Команда встановлення Composer

2) Встановлення Laravel через Composer:

```
composer global require laravel/installer
```

Рисунок 3.2 – Команда встановлення Laravel через Composer

3) Створення нового проекту Laravel:

```
laravel new project-name
```

Рисунок 3.3 – Команда створення нового проекту Laravel

Налаштування середовища. Після створення проекту необхідно налаштувати файл конфігурації .env, який містить основні параметри середовища розробки.

```
APP_NAME=Dokidoki
APP_ENV=local
APP_KEY=
APP_DEBUG=true
APP_URL=http://localhost
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=
```

Рисунок 3.4 – Код налаштування .env файлу

Значення «APP_KEY» в файлі у використовується для забезпечення безпеки додатку. «DB_CONNECTION» дає зрозуміти фреймворку, яка саме БД буде використовуватись. «DB_USERNAME» та «DB_PASSWORD» це секретні дані, які використовуються для доступу до БД.

Створення контролерів. Для роботи з моделями необхідно створити відповідні контролери, в яких буде відбуватися робота з SQL запитами до бази даних, та логіка різних можливостей веб-додатку, пов’язана з відеоконтентом та користувачами.

```
php artisan make:controller Api/AnimeController
php artisan make:controller Api/UsersController
```

Рисунок 3.5 – Команда створення контролерів

Робота з API. Для роботи з API на основі Laravel необхідно створити відповідні маршрути доступу до даних за шляхом «routes/api.php», на які буде приходити запит з клієнтської частини додатку, який буде оброблятися у відповідних контролерах (див. Рис 3.5).

```
        Route::controller(AnimeController::class)-
>group(function () {
    Route::get('api/anime', 'index')->name('api.anime.all');
    Route::get('api/anime/{id}', 'show')-
>name('api.anime.byId');
    Route::get('api/series/anime/get/{animeId}',
'getSeries')->name('api.anime.get');
});
```

Рисунок 3.6 – Код маршрутизації доступу даних

Цей код визначає групу маршрутів, які використовують контролер «AnimeController». За допомогою методу «Route::controller» групуються маршрути, які будуть оброблятись методами цього контролера.

Варто розглянути, що саме робить кожен маршрут. «Route::controller(AnimeController::class)» вказує, що всі маршрути всередині групи будуть використовувати методи контролера «AnimeController->group(function () { ... })», що дозволяє об'єднати кілька маршрутів у групу, що спрощує їх керування та читабельність. Маршрути всередині групи обробляють GET-запити на зазначені URL, та викликають необхідні методи для обробки даних. Кожному маршруту присвоюється ім’я «…->name(‘ІМ’Я’)», яке можна використовувати для генерації URL та посилання на цей маршрут у коді.

3.3 Робота з базою даних

Eloquent ORM є потужним інструментом Laravel для роботи з базами даних, який дозволяє взаємодіяти з базою даних за допомогою об'єктно-орієнтованого синтаксису. Замість написання SQL-запитів, можна

використовувати прості та інтуїтивно зрозумілі методи Eloquent для виконання CRUD-операцій (створення, читання, оновлення та видалення).

Для початку роботи з базою даних, необхідно створити «міграції», які дозволяють визначати структуру БД за допомогою PHP-коду, а не SQL-запитів. Міграції дають змогу створювати, змінювати та видаляти таблиці й колонки в базі даних у спосіб, що є контрольованим та відтворюваним.

В даному проекті будуть використовуватися дві основні міграції, для створення моделі відеоконтенту та користувача.

```
public function up()
    {
        Schema::create('animes', function (Blueprint $table)
{
            $table->bigIncrements('id')->from(1001);
            $table->string('ua_title')->nullable();
            $table->string('en_title')->nullable();
            $table->string('alias')->nullable();
            $table->longText('synopsis')->nullable();
            $table->year('aired')->nullable();
            $table->string('country')->nullable();
            $table->integer('episodes')->nullable();
            $table->integer('total_episodes')->nullable();
            $table->string('director')->nullable();
            $table->string('studio')->nullable();
            $table->string('translated')->nullable();
            $table->time('duration')->nullable();
            $table->string('trailer')->nullable();
            $table->boolean('active')->default(1);
            $table->timestamps();
        });
    }
```

Рисунок 3.7 – Метод створення моделі відеоконтенту

```
public function up()
    {
        Schema::create('users', function (Blueprint $table)
{
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')-
>nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }
```

Рисунок 3.8 – Метод створення моделі користувача

Для того, щоб взаємодіяти з таблицями БД у коді, необхідно створити «моделі», які представляють собою ці таблиці. Наприклад, для таблиці «users», можна створити модель «Users». Для створення моделі використовується команда:

```
php artisan make:model User
```

Рисунок 3.9 – Команда створення моделі

Це створить файл моделі у каталозі «app/Models», після чого можна здійснювати різні дії, такі як визначення відношення між моделями, оголошення змінних, відповідно до офіційної документації Laravel [2].

Розглянемо роботу з контролерами і моделями на прикладі методу реєстрації користувача.

```
public function registered(Request $request)
    {
        $request->validate([
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email',
'max:255', 'unique:' . User::class],
            'password' => ['required',
Rules\Password::defaults()],
        ]);

        $user = User::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make($request->password),
        ]);

        $user->assignRole('Visitor');

        $token = $user->createToken('main')->plainTextToken;

        event(new Registered($user));

        Auth::login($user);

        return response([
            'user' => $user,
            'token' => $token,

        ]);
    }
```

Рисунок 3.10 – Метод реєстрації нового користувача

Метод «validate» перевіряє вхідні дані запиту, щоб переконатися, що вони відповідають визначеним правилам:

- Поле «name» має бути обов'язковим (required), рядковим (string) і не перевищувати 255 символів (max:255).

- Поле email має бути обов'язковим, рядковим, правильно форматованим як адреса електронної пошти, не перевищувати 255 символів і бути унікальним у таблиці «users».

- Поле password має бути обов'язковим та відповідати стандартним правилам валідації пароля (визначені в «Rules\Password::defaults()»).

Метод «User::create» створює новий запис у БД, з використанням даних, які були валідовані. Пароль хешується за допомогою «Hash::make» для збереження у зашифрованому вигляді, після чого, щойно створеному користувачу призначається роль «Visitor», яка обмежує функціональність, згідно з визначеними дозволами.

Для нового користувача створюється токен доступу. Це може бути використано для аутентифікації API-запитів. Метод «createToken» створює новий API токен, а «plainTextToken» повертає цей токен у вигляді звичайного тексту.

Подія «Registered» відбувається після створення нового користувача. Це може бути корисно для відправки вітальних листів або інших дій, які повинні відбуватися після реєстрації.

Метод «Auth::login($user)» визначає, що користувач, який щойно зареєструвався, автоматично увійде в систему.

Метод «return» повертає відповідь, яка містить інформацію про новоствореного користувача та його токен доступу, якою можна маніпулювати на стороні клієнту (відображення ім’я користувача у коментарях).

Цей підхід забезпечує безпечний та ефективний процес реєстрації з мінімальною кількістю коду.

3.4 Реалізація клієнтської частини

Для початку роботи з клієнтською частиною на Next.Js необхідно створити новий проект, за допомогою команд:

```
npx create-next-app my-next-app
cd my-next-app
```

Рисунок 3.11 – Команди створення нового Next.Js проекту

Це створить нову директорію з назвою «my-next-app», в яку буде включена базова структура додатку, встановлена командою розробки фреймворку - Vercel.

Розробка додатку починається з хедеру, на якому буде знаходитися логотип веб-сайту, посилання на різні сторінки, пошук та система авторизації. Головної сторінки, на якій будуть знаходитися каруселі рекомендації контенту для користувача, такі як нові серіали, найпопулярніші, найкращі тощо., рекламні банери, та карусель з рекомендаціями новинок.

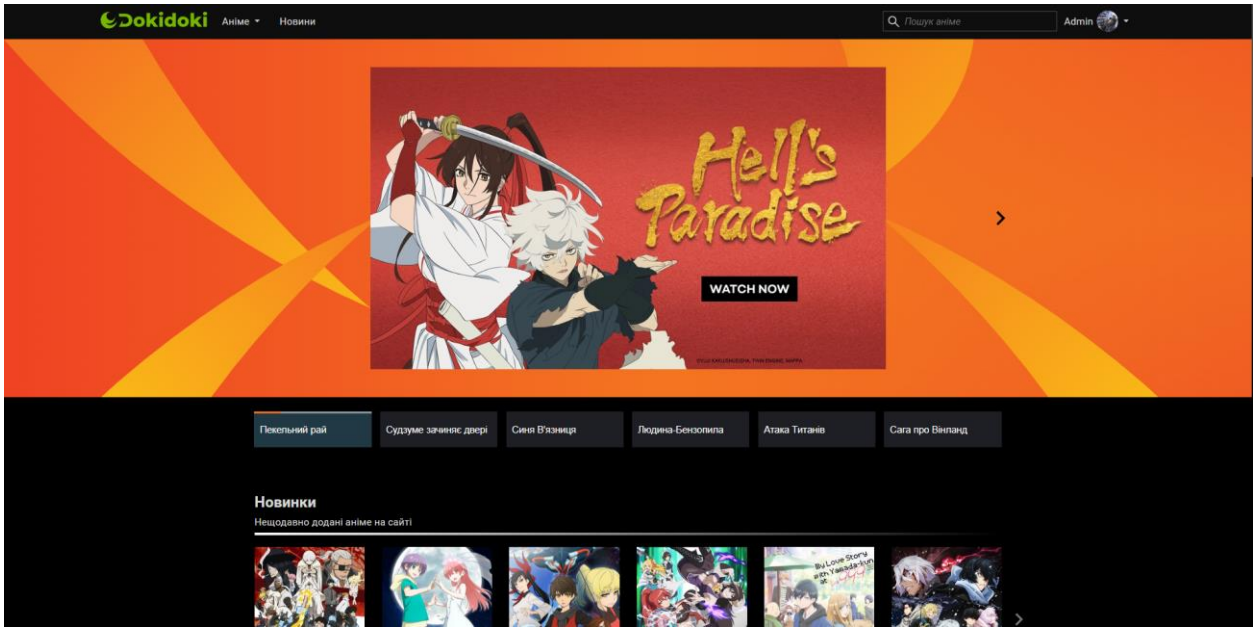

Рисунок 3.12 – Головна сторінка додатку

Список серіалів та картинки, які отримуються з серверної частини, повинні кешуватися на сервері, для забезпечення більш швидкого завантаження сторінок, та відображення контенту на них. Для реалізації цього функціоналу, необхідно використовувати SSR компоненти. В стандартному React проекті, всі запити до серверу на отримання контенту відбуваються безпосередньо у клієнтських компонентах, які не дозволяють кешувати дані на сервері. На момент написання даної дипломної роботи, Next.Js не дозволяє використовувати стандартні клієнтські запити у серверних компонентах, але

для досягнення поставленої задачі існує спосіб розбиття компонентів на окремі частини.

Для пояснення роботи SSR компонентів розглянемо структуру (див. рис. 3.14) і код сторінки будь-якого серіалу (див. рис. 3.13), на якій будуть відображатися: інформація про серіал, відеоплеєр, коментарі користувачів.

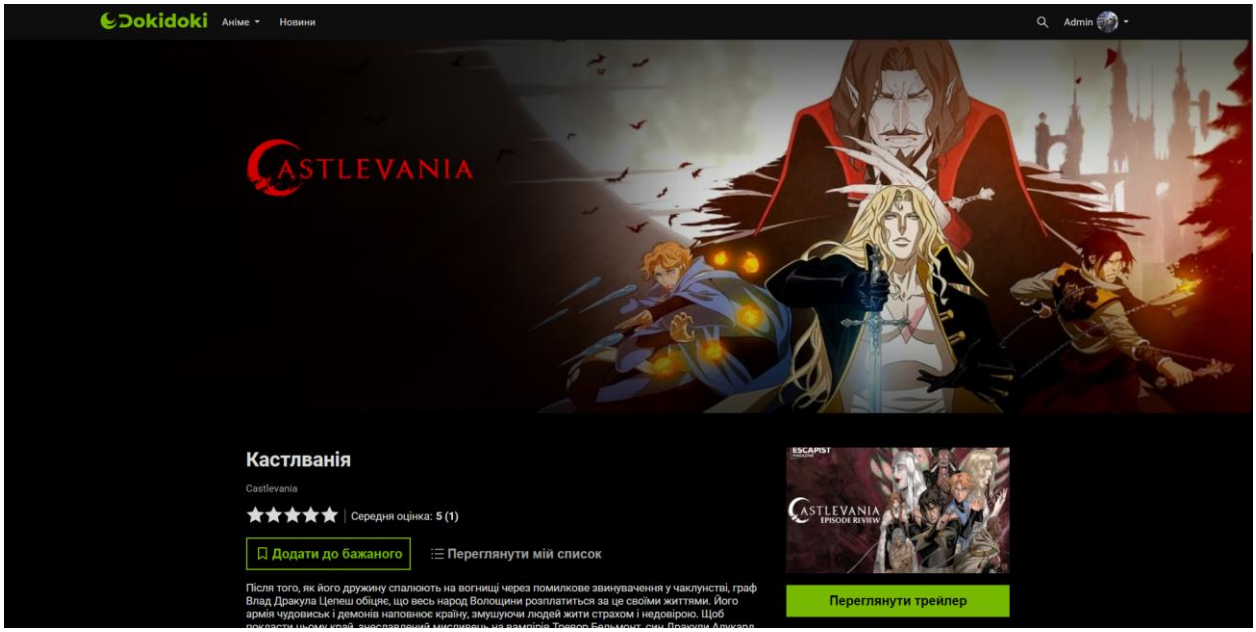

Рисунок 3.13 – Сторінка перегляду серіалу

Рисунок 3.14 – Структура сторінки перегляду серіалу

Шлях до сторінки у браузері буде виглядати наступним чином «http://localhost:5173/anime/series/1106/castlevania», в якому «localhost» - це доменне ім’я, «5173» - порт до якого звертаємось. Можна побачити, що всі наступні сегменти шляху виглядають як послідовність назв папок з рис. 3.14, це базовий спосіб маршрутизації Next.Js. За шляхом «/anime/series/[oid]»

знаходиться «page.jsx», це файл який включає в собі структуру компонентів, які будуть відображатися користувачу.

```
const AnimeSeriesPage = async ({ params }) => {
    const animeData = await fetcher(`/animes/${params.oid}`,
{
        cache: 'no-store',
    })

    return (
        <div className={styles.animePage}>
            <Header animeData={animeData} />
            <Info animeData={animeData}/>
        </div>
    )
}

export default AnimeSeriesPage
```

Рисунок 3.15 – Компонент-сторінка перегляду серіалу

Це серверний компонент, який включає в себе два клієнтських компонента: Header, який відображає велику картинку як задній фон та логотип серіалу, та компонент Info, в якому знаходиться інформація (назва, опис тощо) про серіал. Щоб передати якісь дані в ці два компоненти, необхідно отримати їх з серверної частини, для цього використовується кастомний метод «fetcher», який дозволяє отримувати дані за допомогою стандартного JavaScript запиту:

```
export async function fetcher(endpoint, options = {}) {
    const url =
`${process.env.NEXT_PUBLIC_API_URL}${endpoint}`

    try {
        const response = await fetch(url, {
            ...options,
        })

        if (!response.ok) {
            throw new Error(`HTTP error! Status:
${response.status}`)
        }
        return await response.json()
    } catch (error) {
        throw new Error(`Fetch error: ${error.message}`)
    }
}
```

Рисунок 3.16 – Метод «fetcher» для надсилання запитів на сервер

Він приймає в себе «endpoint» як кінцеву точку API, до якої треба звернутися за даними, та «options» як масив налаштувань цього запиту, таких як вид кешування.

Після відправки запиту і отримання даних, вони записуються у змінну «animeData» (див. рис. 3.15), після чого, цю змінну можна передати вниз по DOM дереву, наприклад у компонент Info, всередині якого буде можливе подальше використання будь-яких даних, що знаходяться в «animeData».

Опираючись на даний приклад, будуть розроблятися подальші компоненти додатку.

Висновки за розділом 3

Досліджено та розглянуто кілька ключових аспектів розробки програмного забезпечення для стрімінгового сервісу. Налаштування середовища розробки визначено як критично важливий етап для забезпечення продуктивності та ефективності роботи над проектом. Використання Visual

Studio Code було обрано як основне середовище розробки, завдяки його зручності та можливостям розширення.

Розглянуто ключові етапи створення веб-додатків з використанням цього фреймворку Laravel, який був визначений як інструмент, який сприяє швидкому та ефективному розробленню серверної частини.

Була описана робота з базою даних за допомогою Eloquent ORM, що дозволяє взаємодіяти з базою даних за допомогою об'єктно-орієнтованого синтаксису, спрощуючи CRUD-операції та забезпечуючи контрольований та відтворюваний підхід до роботи з даними.

Завершальний розділ присвячено реалізації клієнтської частини з використанням Next.Js, де розглядається створення нового проекту, основні кроки початку роботи, та написання SSR компонентів.

РОЗДІЛ 4
АПРОБАЦІЯ ТА РОЗГОРАННЯ ДОДАТКУ

4.1 План та види тестування

Тестування є критично важливим етапом розробки веб-додатку стрімінгового сервісу. Воно забезпечує високу якість продукту, мінімізує кількість помилок та покращує користувацький досвід. Розглянемо план, який визначає стратегію тестування веб-додатку. Основні етапи плану включають визначення мети тестування, обсягу тестування, вибір методів та інструментів, планування тестових сценаріїв, виконання тестів, аналіз результатів та підготовку звіту про тестування.

Перш ніж розпочати процес тестування, необхідно чітко визначити його мету. В цьому випадку, мета тестування полягає у забезпеченні високої якості та надійності веб-додатка, а також у перевірці відповідності вимогам та очікуванням користувачів.

Один із ключових етапів - це визначення обсягу тестування. Ми розглядаємо такі аспекти, як відображення сторінок, робота відеоплеєра, коректне відображення інформації на сторінках, система автентифікації, система коментарів, пошук по сайту та профіль користувача.

Для досягнення мети тестування будуть використовуватися ручні тести.

Після завершення тестування ми проведемо аналіз отриманих результатів. Будуть ідентифіковані виявлені дефекти, оцінено рівень відповідності вимогам та розроблені подальші кроки для виправлення виявлених проблем.

Тестуванню будуть підлягати наступні аспекти веб-додатку:

- Тестування відображення сторінок: Перевірка коректності відображення і розміщення елементів на сторінках.
- Тестування роботи відеоплеєра: Перевірка функціональності відеоплеєра, включаючи відтворення, паузу, перемотку та інші функції.

- Тестування коректного відображення інформації на сторінках: Перевірка правильності відображення тексту, зображень, таблиць, форм та іншої інформації на веб-сторінках.

- Тестування системи автентифікації: Перевірка процесу входу в систему з використанням різних облікових записів і методів автентифікації та перевірка можливостей редагування та управління профілем користувача

- Тестування системи коментарів: Перевірка можливості додавання коментарів та обмеження без необхідних прав.

- Тестування пошуку по сайту: Перевірка функціональності пошукового механізму, включаючи пошук за ключовими словами, фільтрацію та сортування результатів.

4.2 Валідація функціональності та швидкості

Тестування додатку почнеться за першим пунктом плану з розділу 4.1 Перевірка коректності відображення і розміщення елементів на сторінках. Елементи сторінки повинні відображатись з дуже маленькою затримкою, не перетинатись один з одним та відповідати зазначеним стилям.

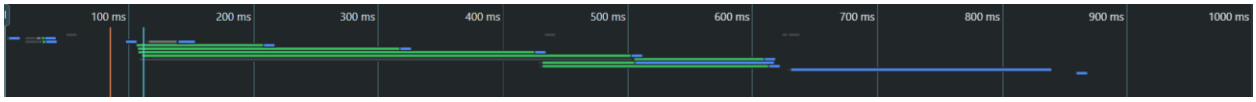

Рисунок 4.1 – Структура сторінки перегляду серіалу

За допомогою влаштованих в браузер засобів тестування, можна побачити шкалу завантаження елементів додатку, час завантаження дорівнює 0.63 мс, що є задовільним показником.

Перевірка функціональності відеоплеєра в більшості залежить від вибраного CDN, що дозволяє швидше завантажувати відео. Тестування буде проводитись на локальному сервері, відео буде завантажуватись з локального сховища, що не надає можливості коректно оцінити швидкість роботи плеєра.

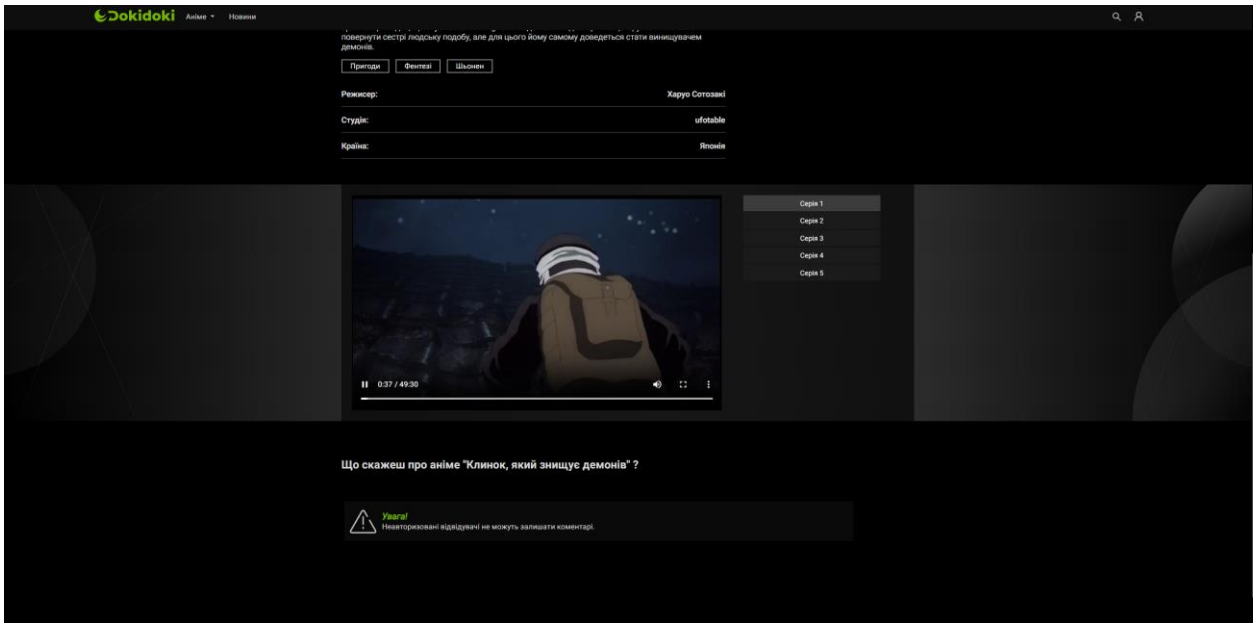

Рисунок 4.2 – Відео в режимі перегляду

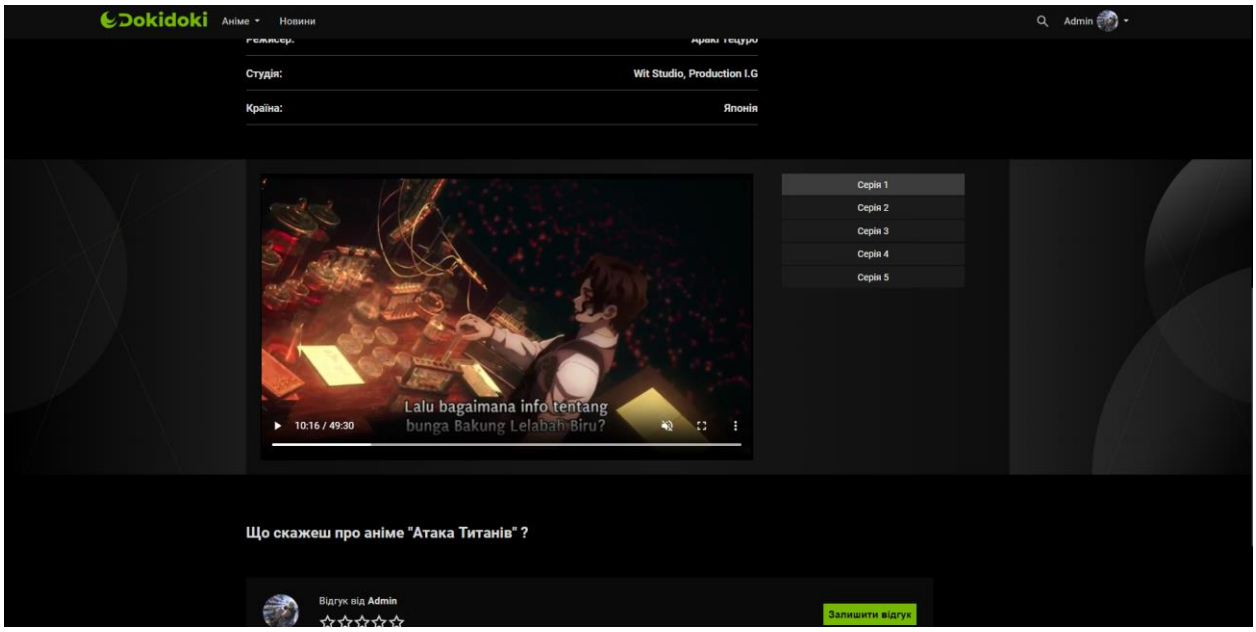

Рисунок 4.3 – Відео на паузі

На рис. 4.2 і 4.3 можна побачити, що «грати», «пауза» та перемотка відео працюють коректно.

Перевірка правильності відображення тексту, зображень, таблиць, форм та іншої інформації на веб-сторінках залежить від JSX та CSS коду. Перевірку будуть проходити такі сторінки, як «Головна сторінка», «Профіль користувача» та «Панель Адміністратора».

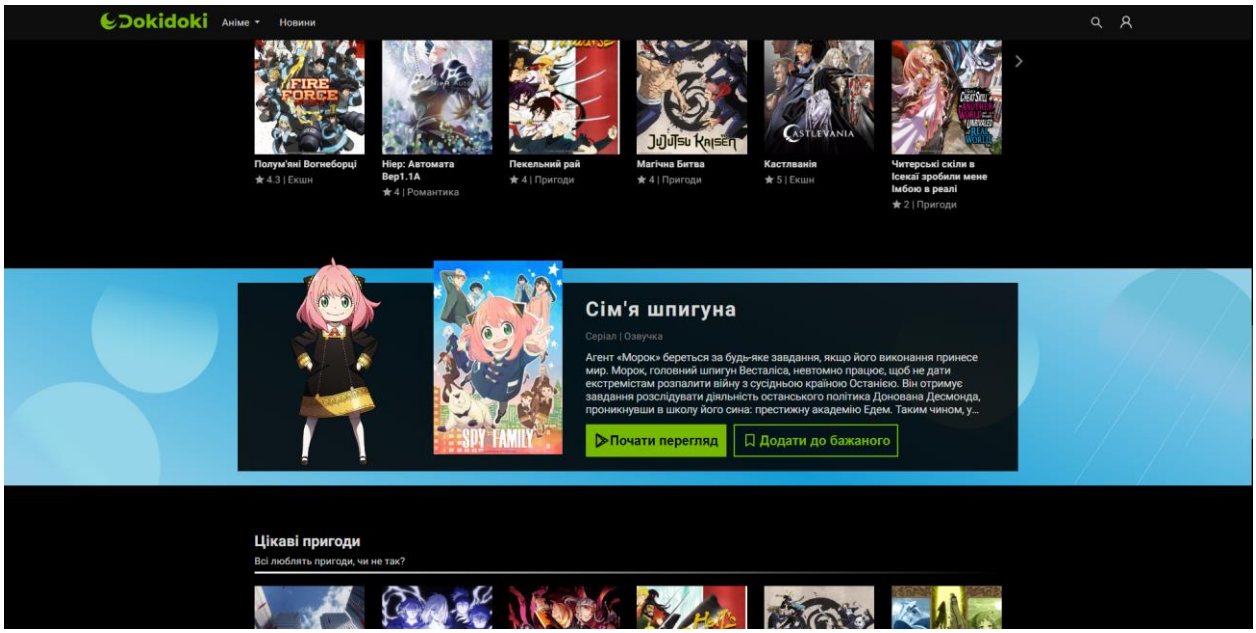

Рисунок 4.4 – Головна сторінка

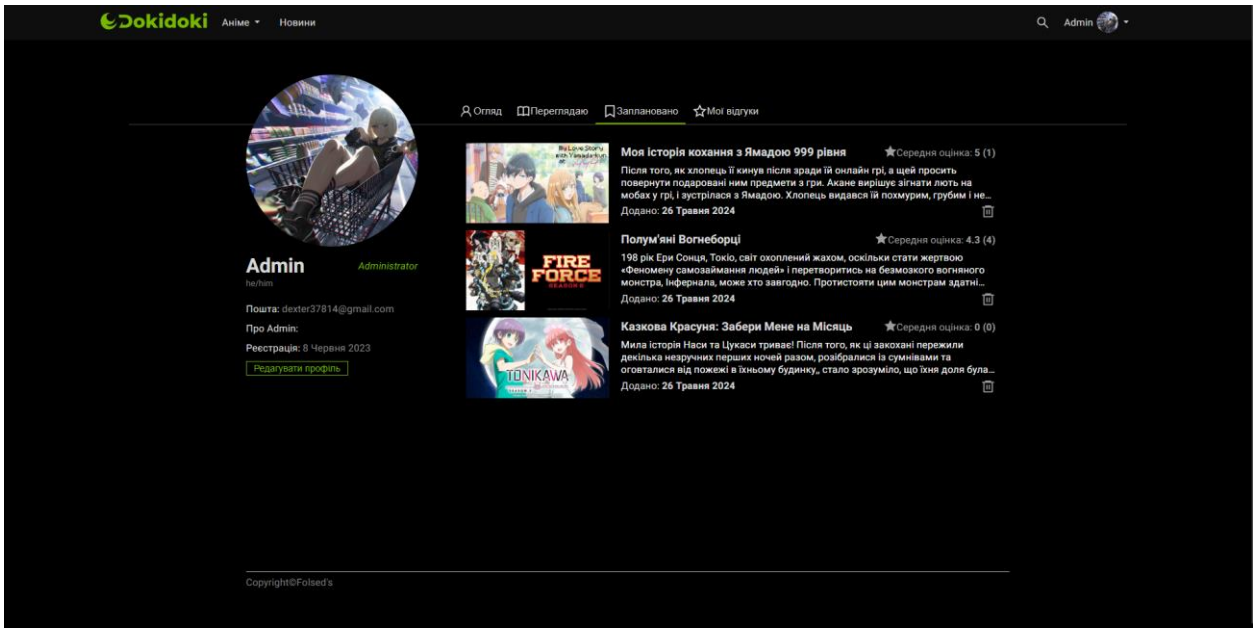

Рисунок 4.5 – Профіль користувача

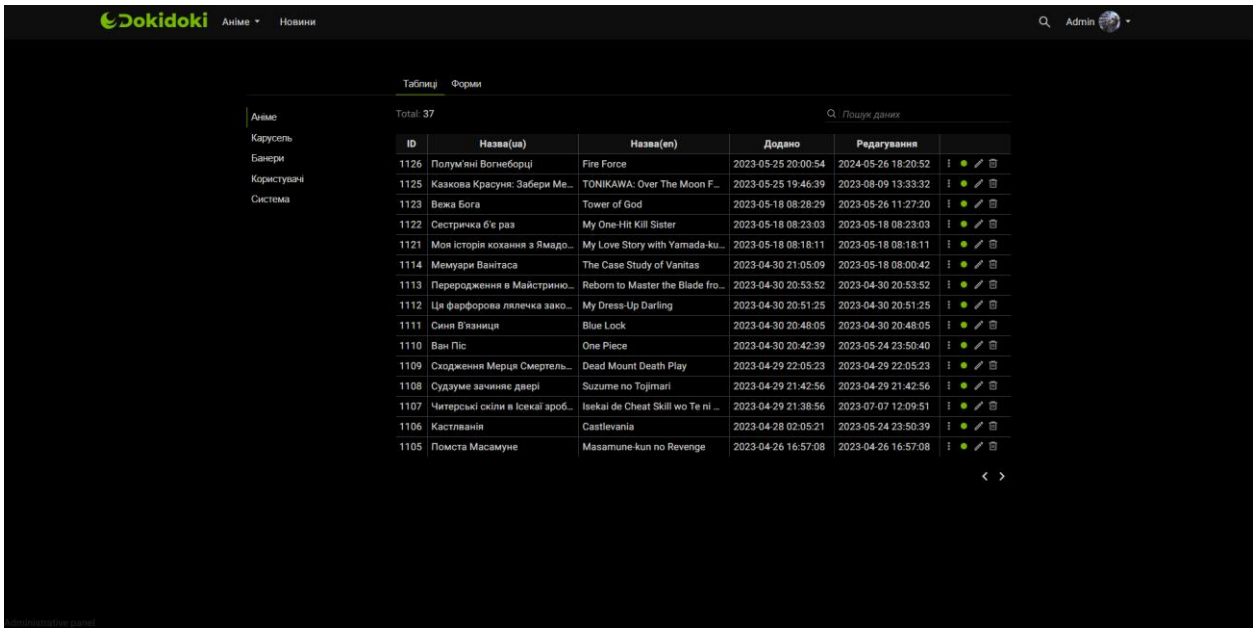

Рисунок 4.6 – Панель адміністратора

Кожен блок контенту відповідає написаному коду, елементи такі, як карусель з серіалами та банери відображаються у відповідних місцях, таблиці в адміністраційній панелі відображаються коректно, включаючи структуру та форматування.

Для тестування процесу входу в систему будуть використовуватись нові дані, яких немає в базі даних. Зареєструємо нового користувача з ім’ям «Kyrylo», електроною поштою «kyrylopost@gmail.com» та паролем «123123123». Після реєстрації перейдемо до профілю.

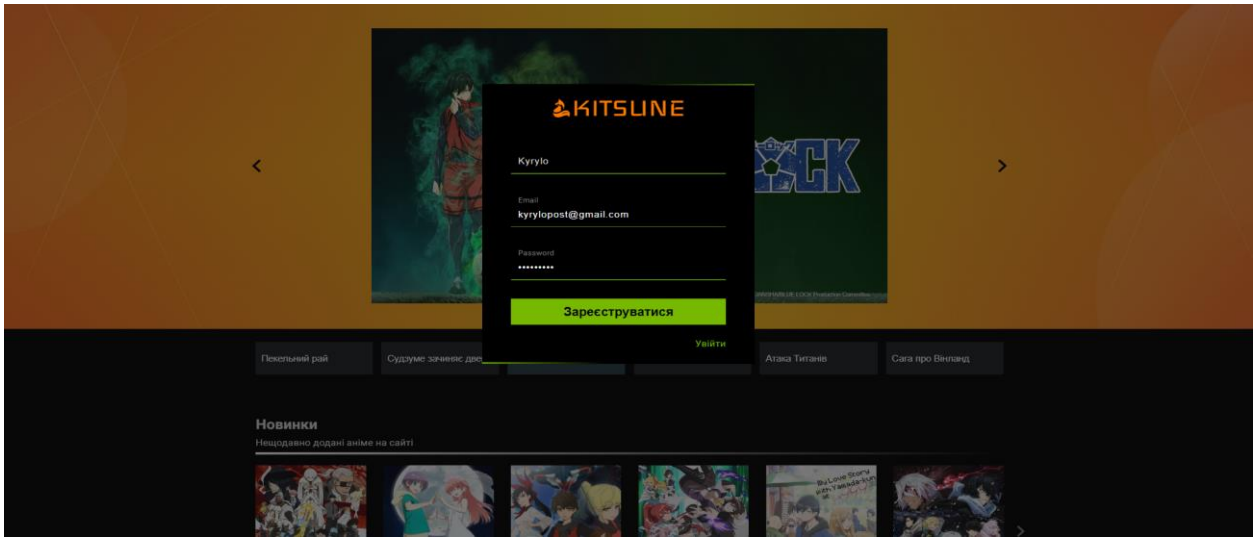

Рисунок 4.7 – Реєстрація нового користувача

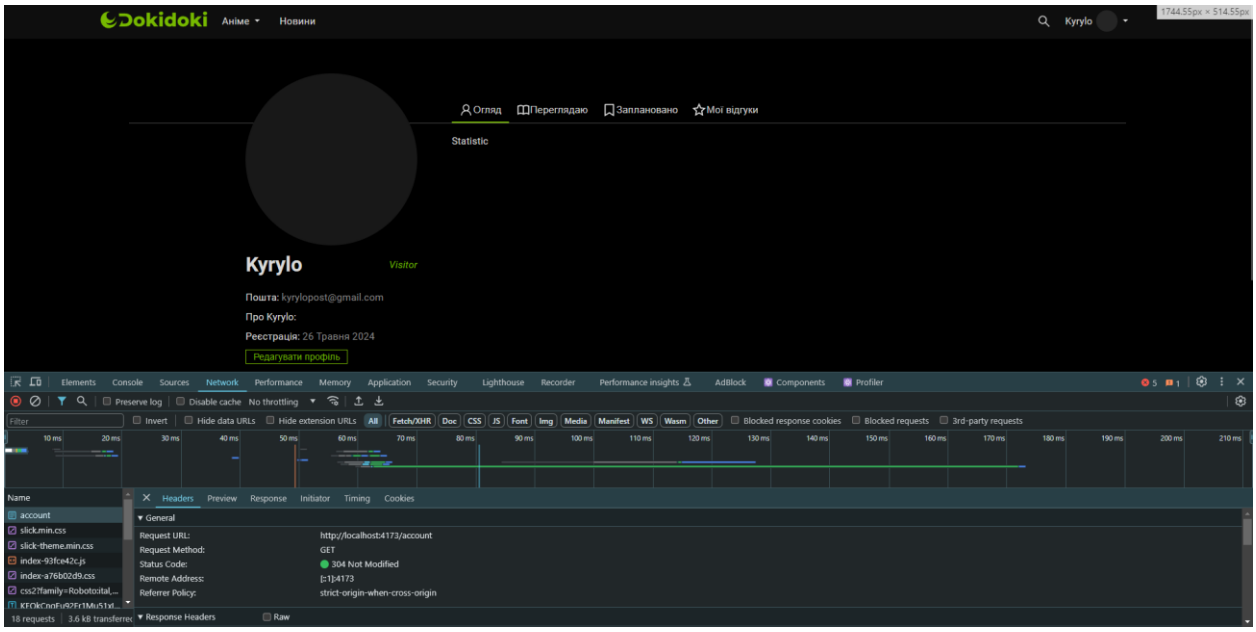

Рисунок 4.8 – Профіль користувача після реєстрації

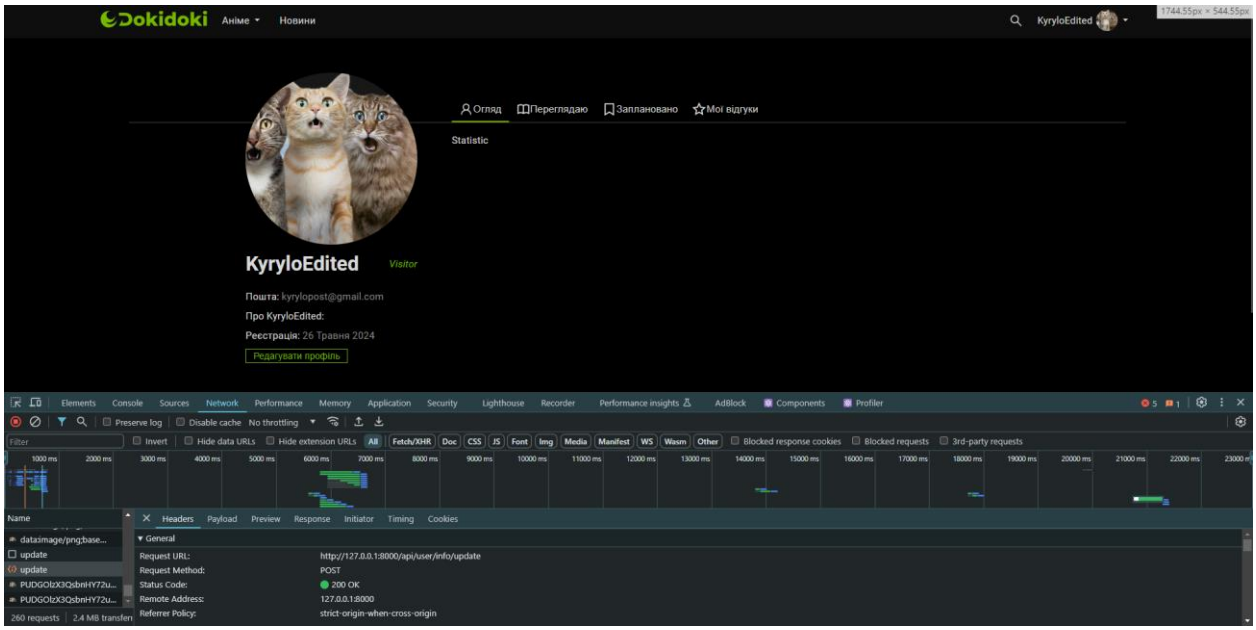

Рисунок 4.9 – Профіль користувача після редагування

На рис. 4.8 можна побачити, що користувач був зареєстрований, та його дані були надіслані з сервера, для відображення на сторінці, а на рис. 4.9, що дані для редагування були успішно відправлені на сервер після чого автоматично змінилися на сторінці.

Перевірка можливості додавання коментарів та обмеження без необхідних прав буде відбуватися з щойно створеного акаунту. Якщо користувач увійшов в систему, у нього повинна бути можливість додавати

коментарів, якщо користувач не авторизований – тільки перегляд вже існуючих, без можливості додати новий.

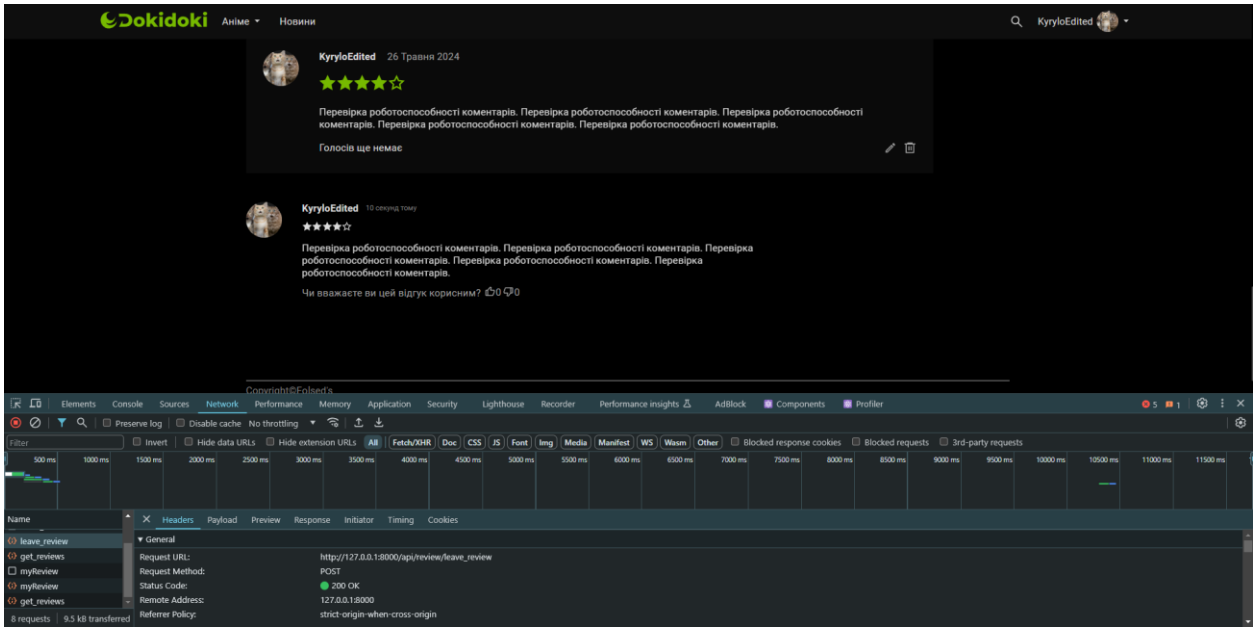

Рисунок 4.10 – Можливість додавати коментар

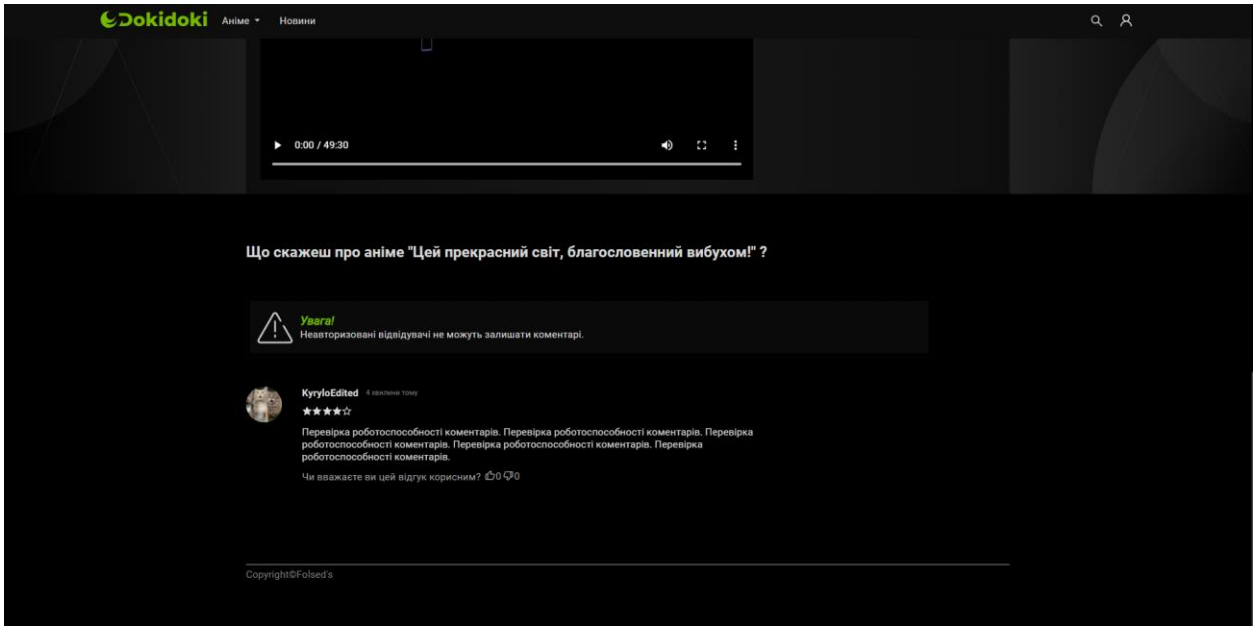

Рисунок 4.11 – Перевірка ролі користувача

На рис. 4.10 і 4.11 можна побачити, що все працює коректно, користувач може додати новий коментар тільки в тому випадку, якщо він увійшов в систему.

Перевірка функціональності пошукового механізму буде відбуватись по ключовому слову або частині слова за допомогою строки пошуку з лівого боку хедера. Фільтрація та сортування винесені в окреме меню, для швидкого доступу, меню можна відкрити натиснувши на кнопку «Аніме» в лівій частині хедеру. В пошуковій строчці було введено «Ця фарф», відповідно до цього набору символів повинно бути виведено всі схожі результати. По натисканню на кнопки жанрів, або списків відсортованого контенту, повинно бути виведено всі серіали з відповідними перевірками.

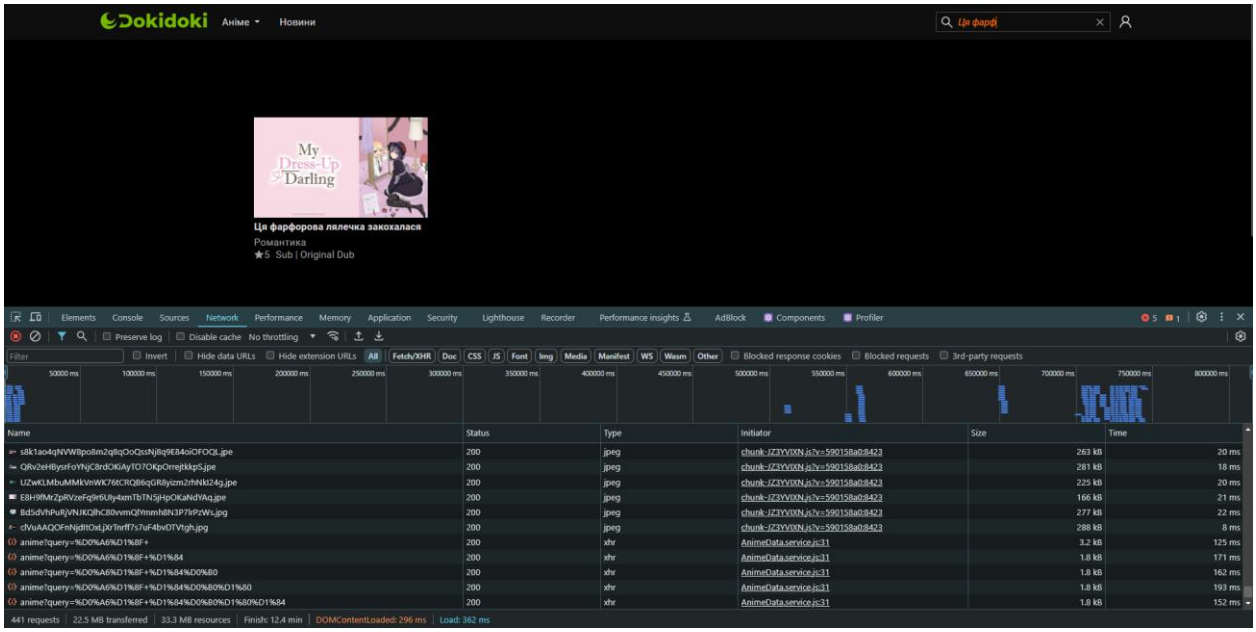

Рисунок 4.12 – Перевірка пошуку

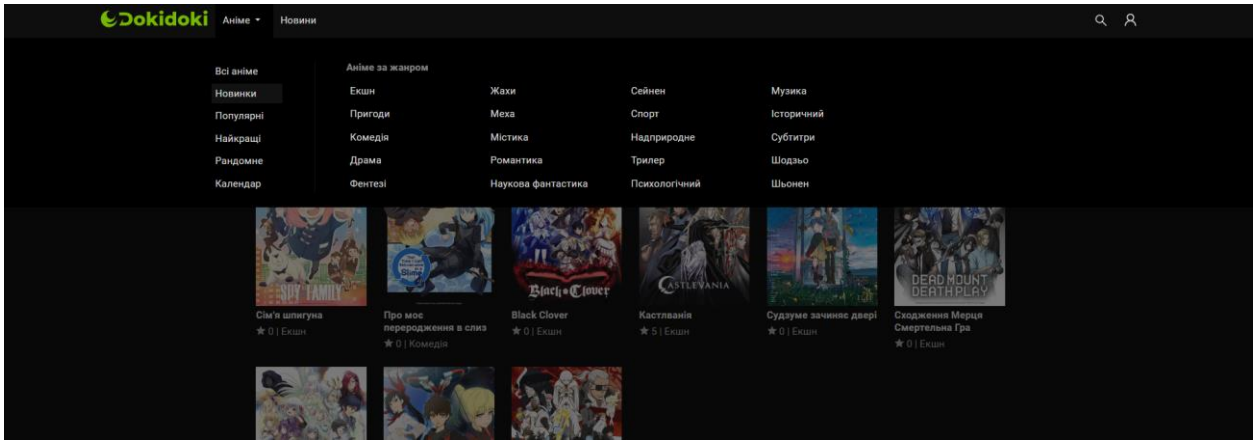

Рисунок 4.13 – Меню сортування серіалів

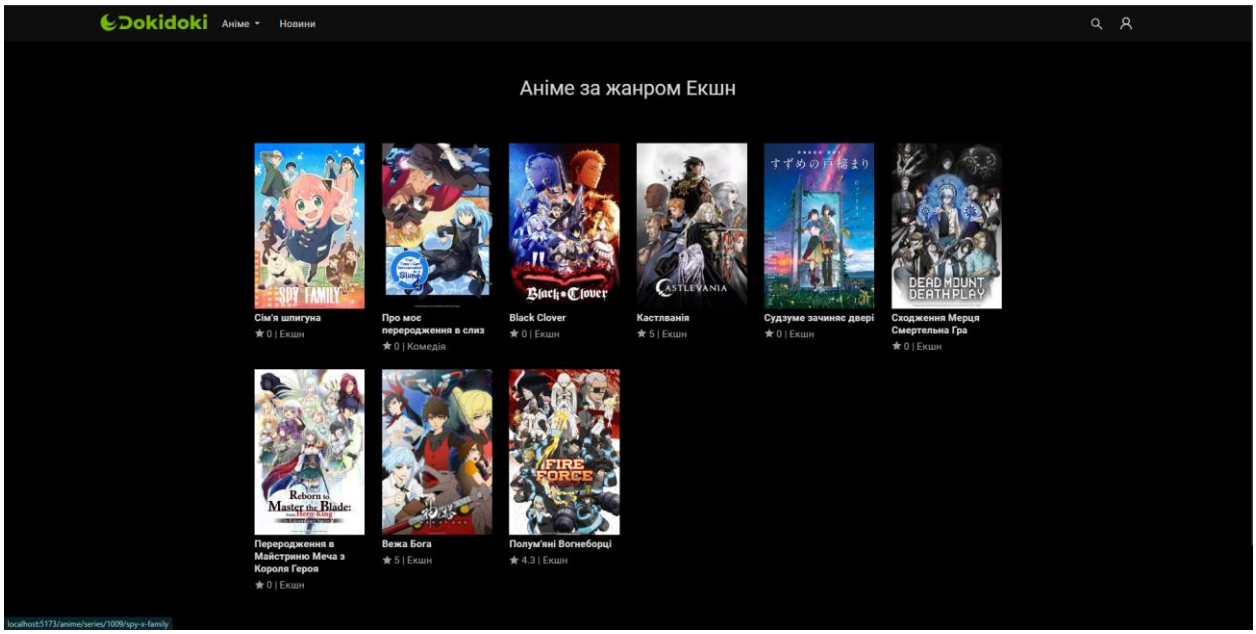

Рисунок 4.14 – Всі серіали по вибраному сортуванню

На рис. 4.12 можна побачити, що через маленьку кількість даних в базі даних, список серіалів зі схожими символами складається лише з одного об’єкту, але пошук працює коректно. рис. 4.13 показує список сортування серіалів за різними критеріями, а рис. 4.14 надає докази, що сортування працює коректно.

4.3 Вибір хостинг-провайдера та розгляд критеріїв для вибору

При розробці та розгортанні веб-додатку, правильний вибір хостинг-провайдера є критичним для забезпечення високої продуктивності, безпеки та доступності сервісу. Для цього проекту було обрано два хостинг-провайдери: Vercel для клієнтської частини та Render для серверної. Розглянемо детальніше, чому саме ці платформи були обрані, а також ключові критерії, які вплинули на цей вибір.

Vercel є популярною платформою для розгортання фронтенд-додатків. Основні переваги цього хостинг-провайдера включають:

- Безкоштовний план: Vercel пропонує безкоштовний тарифний план, що включає більшість необхідних функцій для невеликих та середніх проектів, включаючи необмежену кількість розгортань та 100 ГБ трафіку на місяць.

- Простота використання: інтуїтивно зрозумілий інтерфейс та автоматична інтеграція з популярними системами контролю версій, такими як GitHub, дозволяють швидко та легко розгорнути додаток.

- Глобальна CDN: Vercel використовує глобальну мережу доставки контенту (CDN), що забезпечує високу швидкість завантаження сторінок для користувачів з різних куточків світу.

- Автоматичне масштабування: платформа автоматично масштабує ресурси залежно від навантаження, що гарантує стабільну роботу додатку навіть під час пікових навантажень.

- Безперервна інтеграція та доставка (CI/CD): Vercel підтримує безперервну інтеграцію та доставку, що дозволяє автоматизувати процес розгортання та забезпечувати швидкі оновлення додатку.

Render є потужною платформою для хостингу бекенд-сервісів. Основні переваги цього хостинг-провайдера включають:

- Безкоштовний план: Render також пропонує безкоштовний тарифний план для хостингу веб-сервісів, що є ідеальним для тестування та розгортання проектів.

- Простота налаштування: Платформа підтримує безліч технологій та фреймворків, включаючи Node.js, Python, Ruby та інші, що дозволяє швидко налаштувати середовище для розробки.

- Автоматичне розгортання з Git: Render інтегрується з Git, що дозволяє автоматично розгортати нові версії додатку при кожному пуші до репозиторію.

- Моніторинг та логування: Платформа надає зручні інструменти для моніторингу та логування, що допомагає швидко виявляти та виправляти помилки.

- Безпека: Render забезпечує високий рівень безпеки завдяки використанню сучасних технологій та регулярному оновленню систем.

Обидва провайдери, Vercel та Render, були обрані з урахуванням вищезазначених критеріїв, що забезпечує надійну та ефективну роботу фронтенд та бекенд частин додатку. Завдяки їх безкоштовним тарифним планам, простоті використання та потужним функціональним можливостям.

Для початку тестування додатку, необхідно завантажити файли бекенд частини на хостинг.

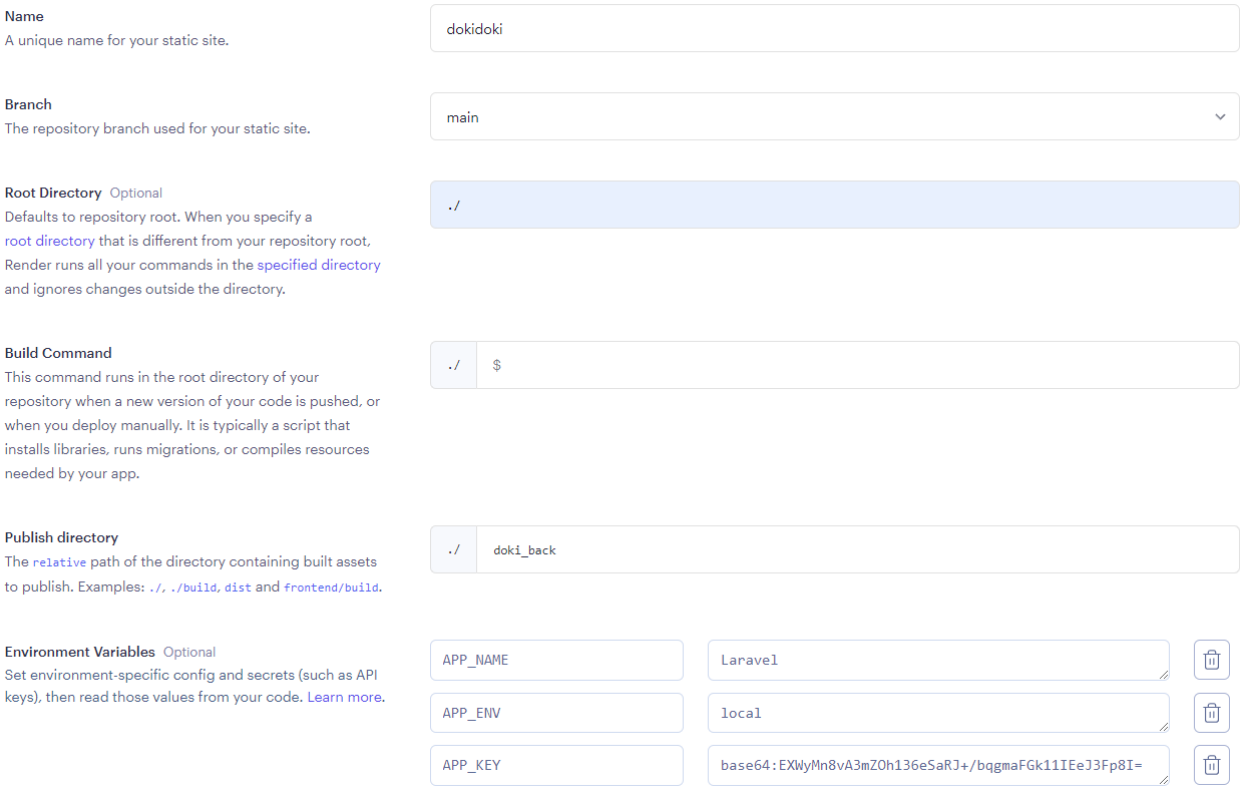

Рисунок 4.15 – Інтерфейс хостингу Render – завантаження на хостинг

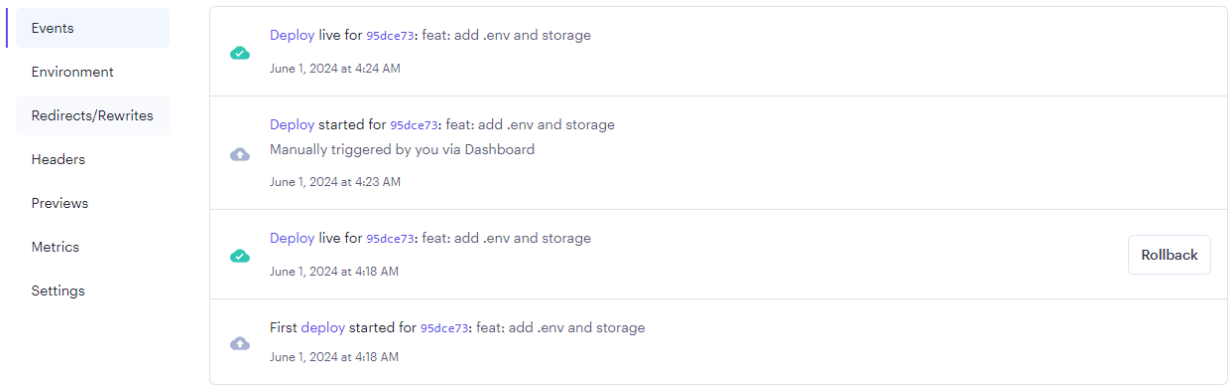

Рисунок 4.16 – Панель управління Render

На рис. 4.15 можна побачити, що були вказані необхідні поля, такі як назва, шлях до директорії на GitHub, та змінні середовища, для коректного функціонування додатку. Після чого, на рис. 4.16 можна переконатися, що завантаження пройшло успішно.

Після завантаження бекенд частини додатку, необхідно зробити те ж саме для фронтенду. Vercel має аналогічне меню, в якому необхідно заповнити такі ж поля, як і у Render.

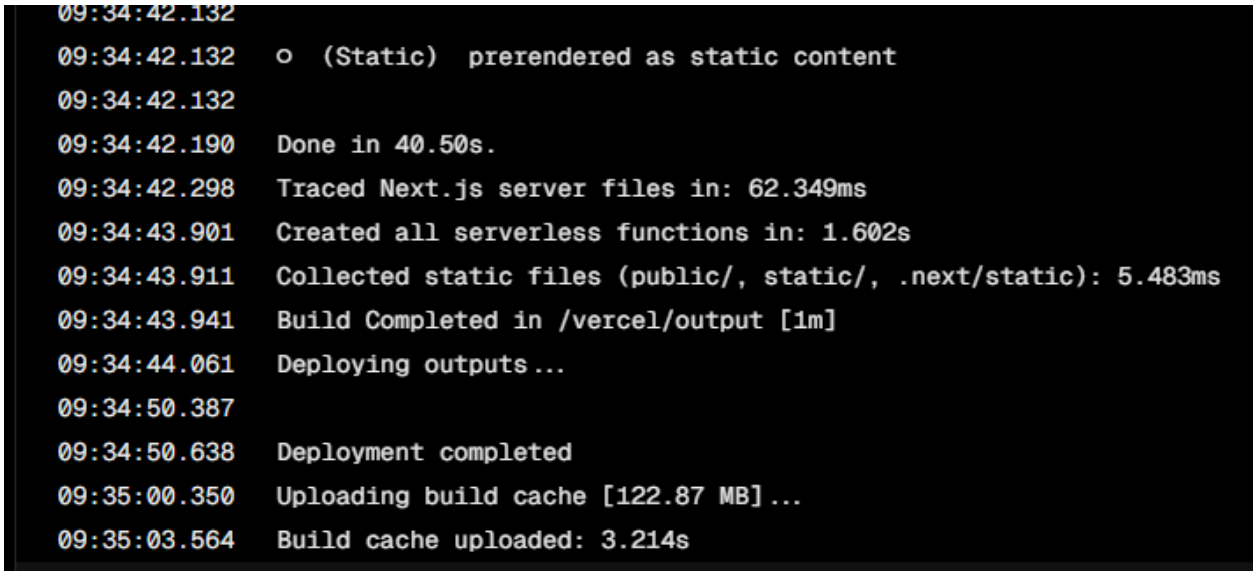

Рисунок 4.17 – Консоль Vercel

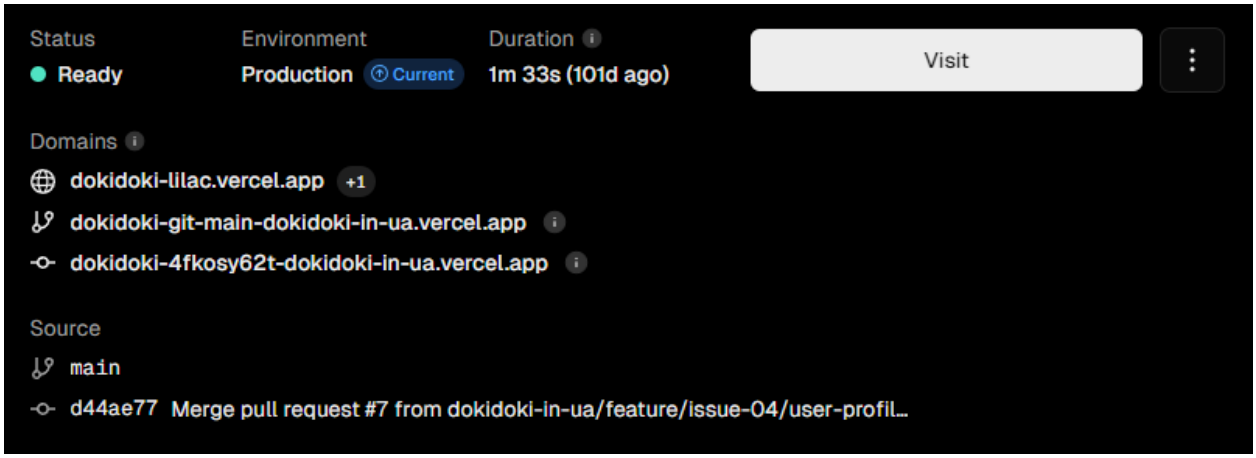

Рисунок 4.18 – Панель загальної інформації Vercel

На рис. 4.17 можна побачити, що завантаження фронтенд частини було проведено успішно, і додаток функціонує коректно, про що свідчить рис 4.18.

Після проведення всіх налаштувань, можна перевірити працездатність додатку та швидкість завантаження даних, вже безпосередньо на хостингу. Швидкість завантаження сторінки не повинна перевищувати одну секунду.

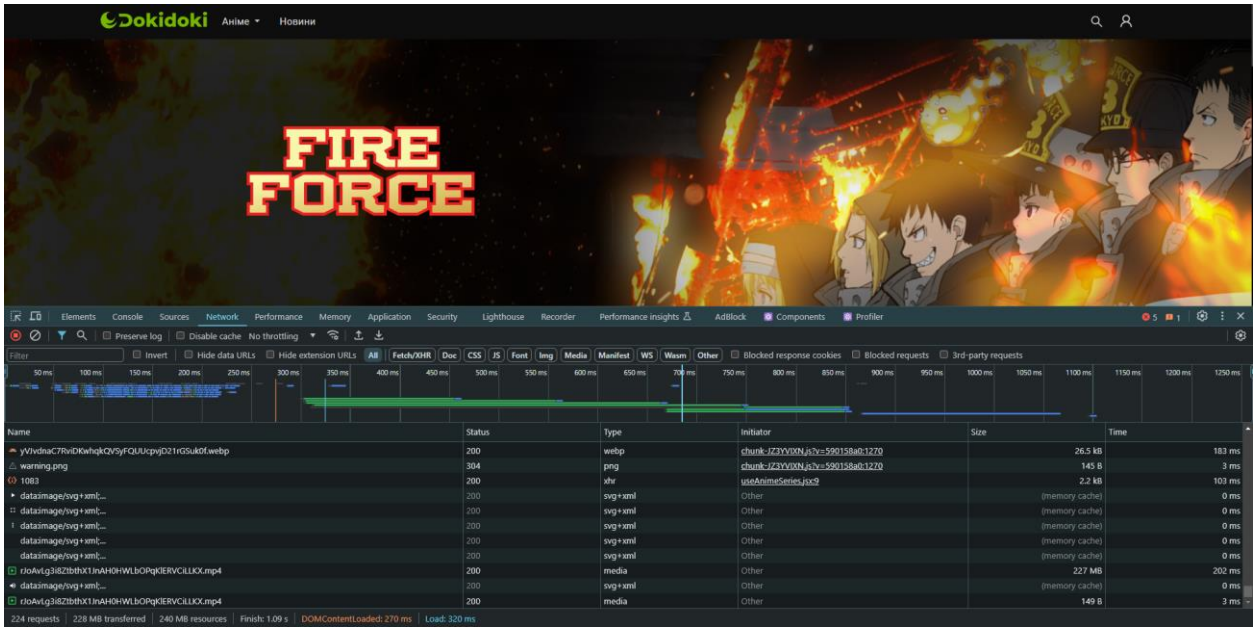

Рисунок 4.19 – Завантаження сторінки на хостингу

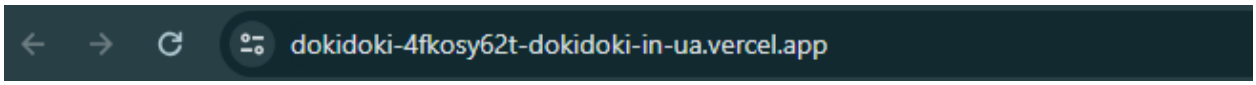

Рисунок 4.20 – URL веб-додатку

На даних рисунках можна побачити, що швидкість завантаження сторінки на хостингу стала ненабагато більша, ніж на локальному сервері, це добрий показник, який відповідає зазначеному критерію.

Висновки за розділом 4

Завершальним етапом розробки веб-додатку стрімінгового сервісу є всебічне тестування, яке включає як функціональне тестування, так і валідацію швидкості та коректності роботи всіх елементів додатку. Основні етапи тестування включають визначення мети, планування тестових сценаріїв, виконання тестів, аналіз результатів та підготовку звіту. Зокрема, тестування

охоплює перевірку відображення сторінок, роботи відеоплеєра, коректного відображення інформації, функціонування систем автентифікації, коментарів та пошуку по сайту.

Обрані хостинг-провайдери, Vercel та Render, підтвердили свою надійність, забезпечивши необхідну швидкість завантаження сторінок, що не перевищує одну секунду. Це забезпечує високу якість та надійність веб-додатку, відповідність вимогам та очікуванням користувачів, що є ключовою метою тестування.

ВИСНОВКИ

У даній дипломній роботі було проведено детальний огляд ринку стрімінгових сервісів, визначено основні тенденції та технологічні інновації, які впливають на розвиток цього сегменту. Досліджено концепцію SSR та її переваги, такі як поліпшення продуктивності та SEO. Особливу увагу приділено огляду технології SSR та аналізу можливостей Next.js, як одного з провідних фреймворків для реалізації цього підходу.

Було розглянуто проектування архітектури веб-додатку. Обрано відповідні архітектурні паттерни, описано структуру додатку, а також спроектовано базу даних для забезпечення ефективного зберігання та обробки даних.

Описано процес реалізації додатку з використанням Next.js та Laravel. Проведено налаштування середовища розробки, реалізовано серверну частину за допомогою Laravel, організовано роботу з базою даних та створено клієнтську частину. Висвітлено ключові аспекти інтеграції компонентів та оптимізації продуктивності.

Проведена апробація та розгортанню додатку. Розроблено план тестування, здійснено валідацію функціональності та швидкодії додатку. Розглянуто критерії вибору хостинг-провайдера та визначено оптимальні варіанти для розгортання розробленої системи.

Таким чином, у дипломній роботі розглянуто повний цикл розробки веб-додатку на основі SSR: від аналізу ринку та теоретичних аспектів до проектування, реалізації та апробації готового продукту. Робота демонструє практичну цінність та потенціал використання сучасних технологій для створення ефективних веб-рішень у сфері стрімінгових сервісів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Video Streaming Market Size, Share & Industry Analysis [Електронний ресурс]. – режим доступу: URL: https://www.fortunebusinessinsights.com/video-streaming-market-103057 (дата звернення – 29.12.2023)

2. Документація. Laravel Documentation [Електронний ресурс]. – режим доступу: URL: https://laravel.com/docs/11.x (дата звернення – 02.01.2024)

3. Tarun, Dr. Vishal Shrivastava, Dr. Akhil Pandey / International Journal of Research Publication and Reviews, Vol 5, no 3, pp 1601-1604 March 2024 / [Електронний ресурс]. – режим доступу: URL: https://ijrpr.com/uploads/V5ISSUE3/IJRPR23506.pdf (дата звернення – 23.12.2023)

4. Evan Burchard «Refactoring JavaScript: Turning Bad Code Into Good Code», O'Reilly, 2017.

5. Документація. Next.Js Documentation [Електронний ресурс]. – режим доступу: URL: https://nextjs.org/docs

6. Документація. Vercel провайдер [Електронний ресурс]. – режим доступу: URL: https://vercel.com/docs (дата звернення – 25.03.2024)

7. Документація. Render провайдер [Електронний ресурс]. – режим доступу: URL: https://docs.render.com (дата звернення – 25.03.2024)

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **бакалавр**
Галузь знань: **12 – Інформаційні технології**
Спеціальність: **123 – Комп'ютерна інженерія.**

ЗАТВЕРДЖУЮ
Завідувач кафедри
теоретичної та прикладної системотехніки
_____д.т.н., проф. Шматков С. І.
«21» грудня
2023 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

КОЛГАНОВА КИРИЛА ЄВГЕНОВИЧА

_____
(прізвище, ім'я, по батькові студента)

1. Тема роботи «МОДЕЛЬ ОНЛАЙН КІНОТЕАТРУ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ SSR КОМПОНЕНТІВ»

керівник роботи к.т.н., доцент кафедри математичного моделювання та штучного інтелекту **КОРОБЧИНСЬКИЙ Кирил Петрович**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "03" травня 2024року №4101-5/909

2.Строк подання студентом роботи 31 травня 2024року

3. Перелік питань, які потрібно розробити

1. Побудувати модель веб сайту
2. Побудувати модель бази даних
3. Розробити веб додаток моделі онлайн кінотеатру
4. Застосувати SSR компоненти у проекті

5. Підготувати наукову статтю за темою роботи

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Аналіз наукової літератури	21.12.2023 – 25.01.2024
2	Аналіз конкурентів і пошук статей відповідно до вибраної теми	19.12.2023 – 2.01.2024
3	Побудова моделі веб сайту і бази даних	2.01.2024 – 2.02.2024
4	Розробка веб сайту і бази даних на базі моделі	2.01.2024 – 2.02.2024
5	Використання SSR компонентів в проекті	2.02.2024 – 10.02.2024
6	Тестування та апробація проекту	11.02.2024 – 30.03.2024
7	Підготовка наукової статті	3.03.2024 – 30.04.2024
8	Розробка пояснювальної записки.	31.03.2024 – 27.05.2024

5. Дата видачі завдання 21.12.2023

Студент К. Є. Колганов _____
ініціали, прізвище підпис

Керівник роботи К. П. Коробчинський __ _____
ініціали, прізвище

Додаток Б

Затверджую

_____

«_____»_____2024 р.

Технічне завдання
на розробку програмного виробу «МОДЕЛЬ ОНЛАЙН КІНОТЕАТРУ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ SSR КОМПОНЕНТІВ»

1.	Введення	1.1. Назва: Метод аналізу інформативності змінних стану при діагностиці систем з використанням інформаційних критеріїв. 1.2. Галузь застосування: інформаційні технології.
2.	Підстава для розробки	2.1. Навчальний план за спеціальністю 123 – Комп'ютерна інженерія. 2.2. Завдання на кваліфікаційну роботу бакалавра № 4101-5/909 від «03» травня 2024 (представити як Додаток А до пояснювальної записки до кваліфікаційної роботи).
3.	Призначення розробки	3.1. Мета розробки: забезпечення ефективного використання SSR для покращення швидкості завантаження сторінок та забезпечення користувачам зручного і швидкого доступу до контенту. 3.2. Призначення розробки надає можливість створити високоякісний та швидкий відеосервіс, який зможе конкурувати з іншими популярними платформами такими як Netflix, YouTube або Hulu. 3.3. Вихідні дані розробки: статистика використання, аналітика швидкості завантаження, дані щодо контенту, дані щодо SEO
4.	Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик: відсутні. 4.2. Вимоги до надійності: забезпечувати точність та достовірність визначення інформативності параметрів стану не гіршу за існуючі методи та програмне забезпечення. 4.3. Вимоги до умов експлуатації: відсутні. 4.4. Вимоги до складу і параметрів технічних засобів: звичайне обчислювальне обладнання, ПК, тощо.

<table>
<tr><td></td><td></td><td colspan="2">4.5. Вимоги до інформаційної та програмної сумісності: відсутні.
4.6. Вимоги до маркування та упаковки: відсутні.
4.7. Вимоги до транспортування і зберігання: на звичайних носіях інформації.
4.8. Спеціальні вимоги: відсутні.</td></tr>
<tr><td>5.</td><td>Вимоги до програмної документації</td><td colspan="2">Програмною документацією до виробу «Модель онлайн кінотеатру з використанням технології SSR компонентів» вважати:
1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи).
2) Методику розробки одно-сторінкового веб-додатку (у вигляді розділу 3 пояснювальної записки до кваліфікаційної роботи).
3) Опис виробу (представлений в розділі 1 пояснювальної записки до кваліфікаційної роботи)</td></tr>
<tr><td>6.</td><td>Вимоги до техніко-економічних показників</td><td colspan="2">Програмною документацією до виробу «Модель онлайн кінотеатру з використанням технології SSR компонентів» вважати:
1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи).
2) Методику розробки одно-сторінкового веб-додатку (у вигляді розділу 3 пояснювальної записки до кваліфікаційної роботи).
3) Опис виробу (представлений в розділі 1 пояснювальної записки до кваліфікаційної роботи)</td></tr>
<tr><td>7.</td><td>Стадії і етапи розробки</td><td>Дата</td><td>Назва етапу</td></tr>
<tr><td></td><td></td><td>від 21 грудня 2023 до 25 січня 2024

від 19 грудня 2023 до 2 січня 2024</td><td>Аналіз наукової літератури.

Аналіз конкурентів і пошук статей відповідно до вибраної теми.</td></tr>
</table>

		від 2 січня 2024 до 2 лютого 2024 від 2 січня 2024 до 2 лютого 2024 від 2 лютого 2024 10 лютого 2024 від 11 лютого 2024 до 30 березня 2024 від 3 березня 2024 до 30 квітня 2024 від 31 березня 2024 до 27 травня 2024	Побудова моделі веб сайту і бази даних. Розробка веб сайту і бази даних на базі моделі. Використання SSR компонентів в проекті. Тестування та апробація проекту. Підготовка наукової статті Розробка пояснювальної записки.
8.	Порядок контролю і приймання програмного продукту (моделі)	1. Перевірку ходу розробки програми виконувати раз в 3 тижні. 2. Захист розробленої моделі провести на засіданні Атестаційної комісії. 3. Пояснювальну записку подати в електронному вигляді в 1 примірнику та роздрукованому вигляді.	

Виконавець
студент групи КІ- 41
КОЛГАНОВ К. Є.

Замовник
к. т. н., доцент
КОРОБЧИНСЬКИЙ К. П.

Додаток В

Програма і методика випробувань програмного виробу

«МОДЕЛЬ ОНЛАЙН КІНОТЕАТРУ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ SSR КОМПОНЕНТІВ»

1. Об'єкт випробувань

1. Назва програмного виробу : «Модель онлайн кінотеатру з використанням технології SSR компонентів»
2. Галузь застосування : Розваги та освіта
3. Перераховані відомості запозичуються з відповідних розділів Технічного завдання.

2. Мета випробувань

Перевірка відповідності функціональності програмної реалізації системи заявленим функціональним можливостям в технічному завданні (Додаток Б до пояснювальної записки до кваліфікаційної роботи).

3. Загальні положення

1. **Підстави для проведення випробувань**
   Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

2. **Місце і тривалість випробувань**
   Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу кафедри в період роботи атестаційної комісії.
3. **Обсяг випробувань**
   Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї програми і методики випробувань.
4. **Організації, які беруть участь у випробуваннях**
   Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.
5. **Вимоги до програми або програмного виробу**

Модель повинна задовольняти наступним вимогам:

1. працювати на основних операційних системах: Windows, Linux, MacOS;

2. вимоги до надійності;
3. передбачити захист від некоректних дій користувача;
4. сумісність з іншими програмними продуктами;
5. зменшити об’єм програмного коду необхідного для створення веб-додатків;
6. бути легко розширюваною;
7. елементи програми повинні бути ізольовані одне від одного для зменшення їх впливу на роботи програми під час редагування програмного коду;
8. вимоги до складу і параметрів технічних засобів;
9. вимоги до маркування та упаковки (не висуваються);
10. вимоги до транспортування і зберігання (не висуваються).

Спеціальні вимоги (не пред’являються).

5. Вимоги до програмної документації

Програмною документацією до виробу «Модель онлайн кінотеатру з використанням технології SSR компонентів» вважати:

1. Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи).
2. Методику розробки одно-сторінкового веб-додатку (у вигляді розділу 3 пояснювальної записки до кваліфікаційної роботи).
3. Опис виробу (представлений в розділі 1 пояснювальної записки до кваліфікаційної роботи)

6. Засоби і порядок випробувань

6.1 Засоби випробувань

Для досягнення мети тестування будуть використовуватися ручні тести.

Після завершення тестування ми проведемо аналіз отриманих результатів. Будуть ідентифіковані виявлені дефекти, оцінено рівень відповідності вимогам та розроблені подальші кроки для виправлення виявлених проблем.

Тестуванню будуть підлягати наступні аспекти веб-додатку:

- Тестування відображення сторінок: Перевірка коректності відображення і розміщення елементів на сторінках.

- Тестування роботи відеоплеєра: Перевірка функціональності відеоплеєра, включаючи відтворення, паузу, перемотку та інші функції.

- Тестування коректного відображення інформації на сторінках: Перевірка правильності відображення тексту, зображень, таблиць, форм та іншої інформації на веб-сторінках.

- Тестування системи автентифікації: Перевірка процесу входу в систему з використанням різних облікових записів і методів автентифікації та перевірка можливостей редагування та управління профілем користувача

- Тестування системи коментарів: Перевірка можливості додавання коментарів та обмеження без необхідних прав.

- Тестування пошуку по сайту: Перевірка функціональності пошукового механізму, включаючи пошук за ключовими словами, фільтрацію та сортування результатів.

6.2 Порядок проведення випробувань

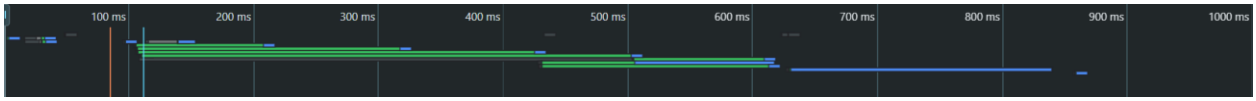

Рисунок В.1 – Структура сторінки перегляду серіалу

За допомогою влаштованих в браузер засобів тестування, можна побачити шкалу завантаження елементів додатку, час завантаження дорівнює 0.63 мс, що є задовільним показником.

Перевірка функціональності відеоплеєра в більшості залежить від вибраного CDN, що дозволяє швидше завантажувати відео. Тестування буде проводитись на локальному сервері, відео буде завантажуватись з локального сховища, що не надає можливості коректно оцінити швидкість роботи плеєра.

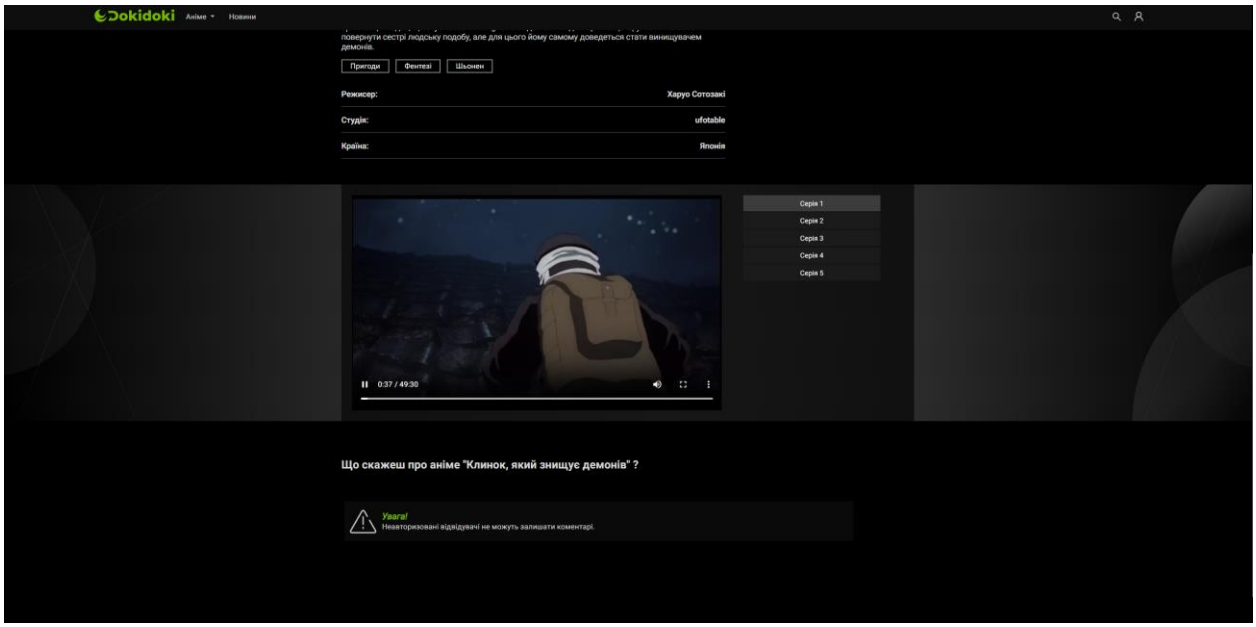

Рисунок В.2 – Відео в режимі перегляду

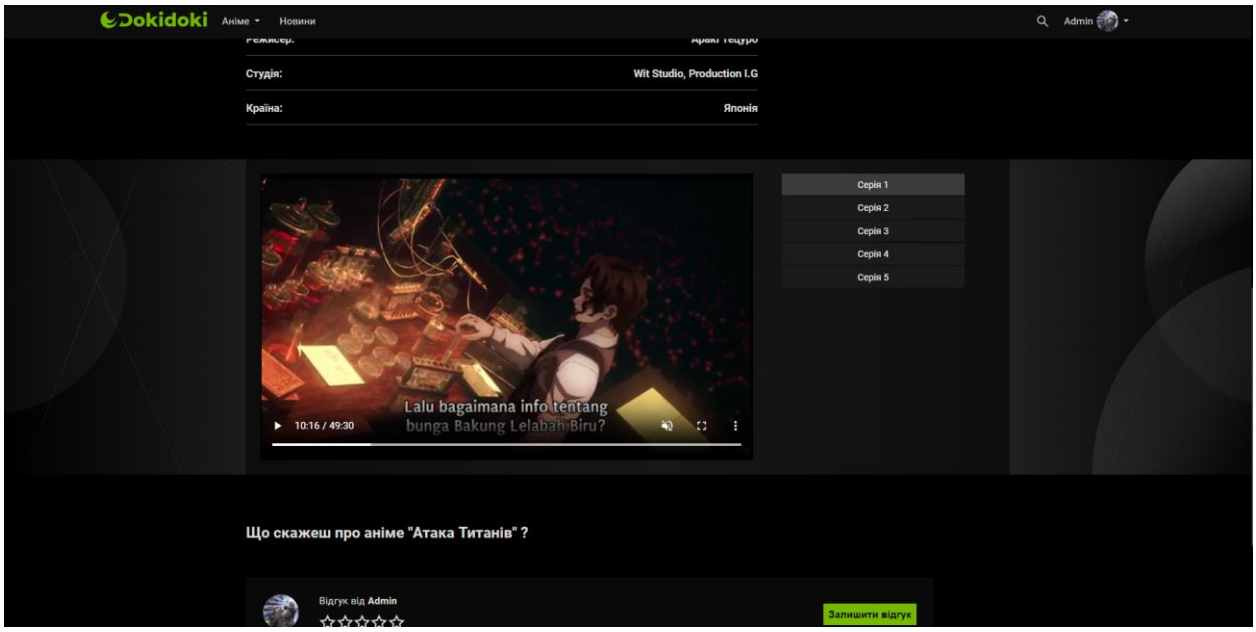

Рисунок В.3 – Відео на паузі

На рис. В.2 і В.3 можна побачити, що «грати», «пауза» та перемотка відео працюють коректно.

Перевірка правильності відображення тексту, зображень, таблиць, форм та іншої інформації на веб-сторінках залежить від JSX та CSS коду. Перевірку будуть проходити такі сторінки, як «Головна сторінка», «Профіль користувача» та «Панель Адміністратора».

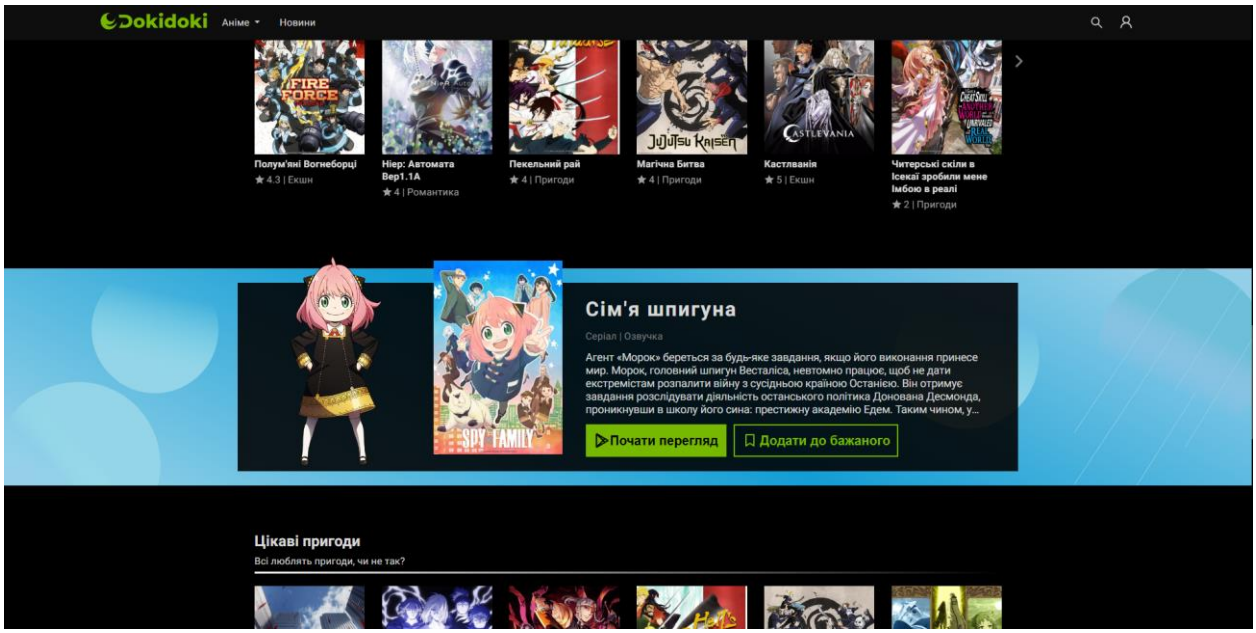

Рисунок В.4 – Головна сторінка

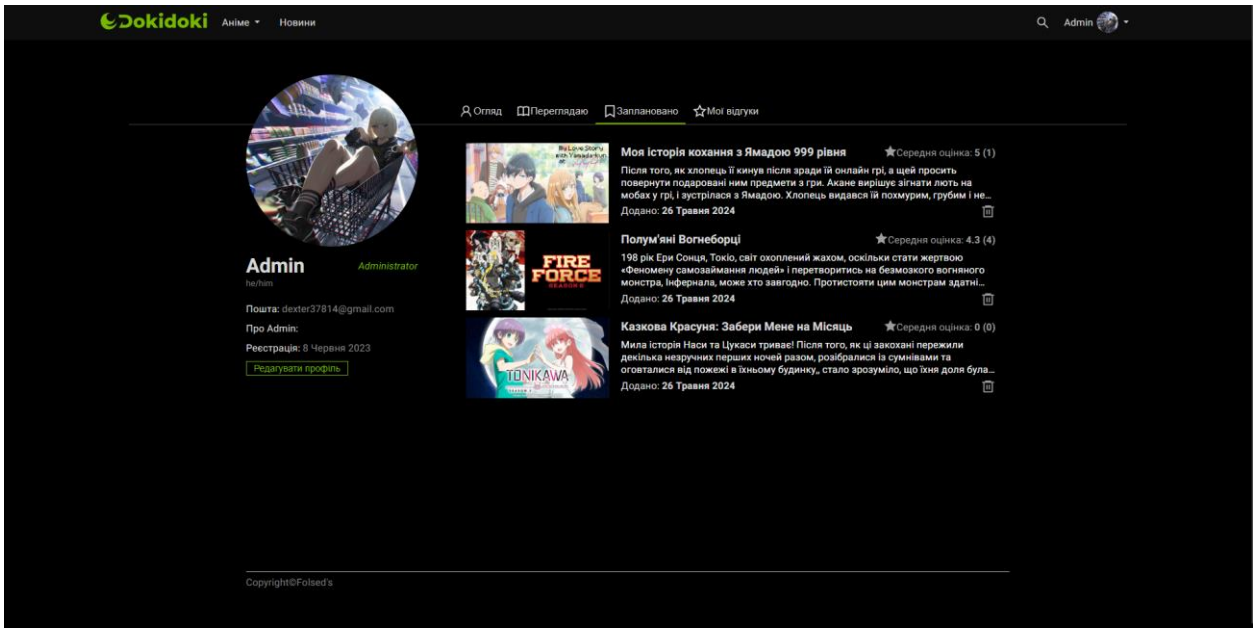

Рисунок В.5 – Профіль користувача

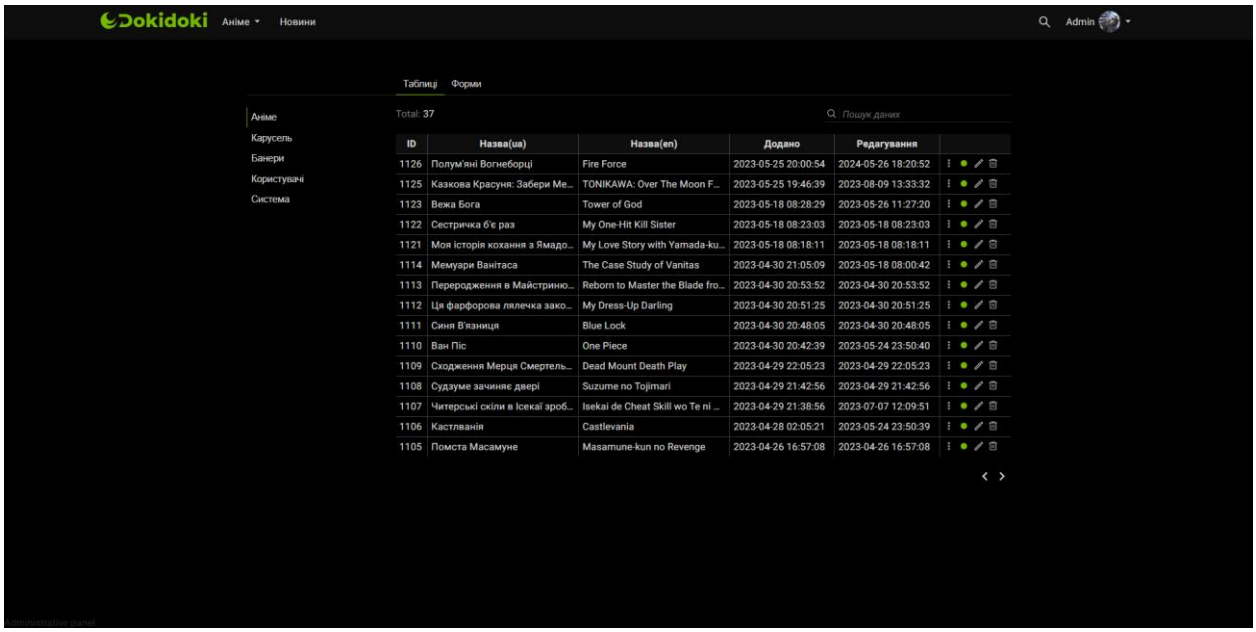

Рисунок В.6 – Панель адміністратора

Кожен блок контенту відповідає написаному коду, елементи такі, як карусель з серіалами та банери відображаються у відповідних місцях, таблиці в адміністраційній панелі відображаються коректно, включаючи структуру та форматування.

Для тестування процесу входу в систему будуть використовуватись нові дані, яких немає в базі даних. Зареєструємо нового користувача з ім’ям «Kyrylo», електроною поштою «kyrylopost@gmail.com» та паролем «123123123». Після реєстрації перейдемо до профілю.

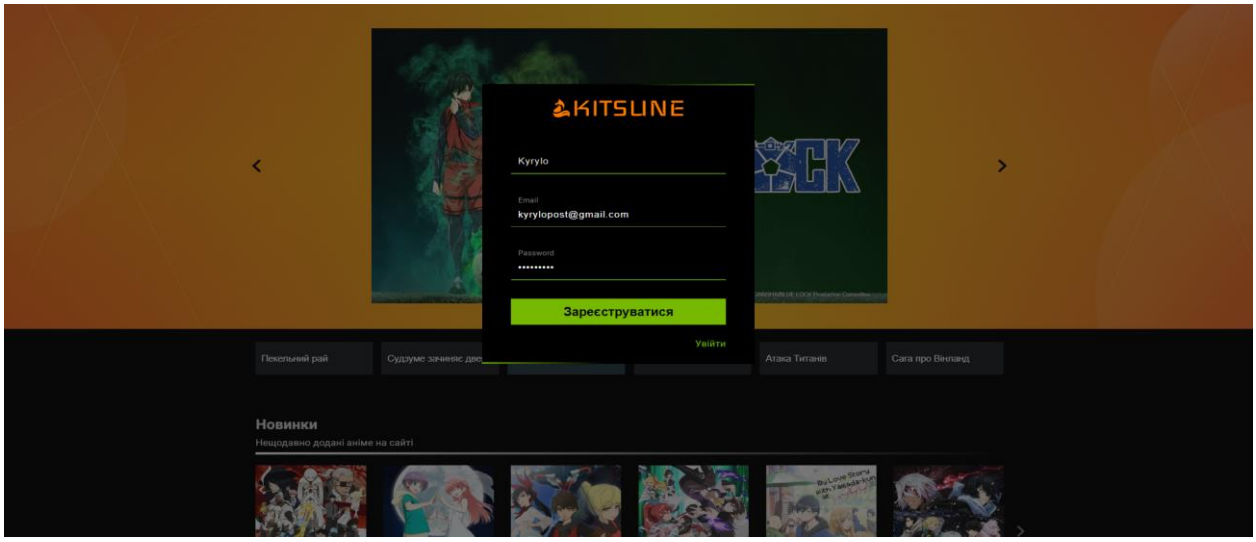

Рисунок В.7 – Реєстрація нового користувача

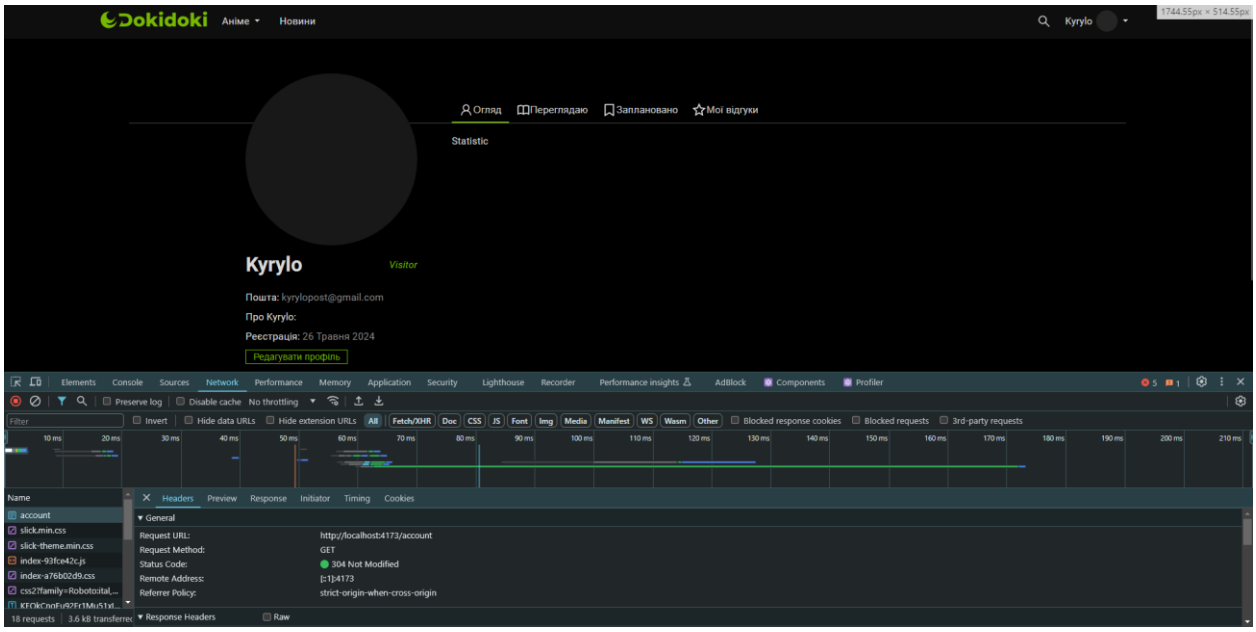

Рисунок В.8 – Профіль користувача після реєстрації

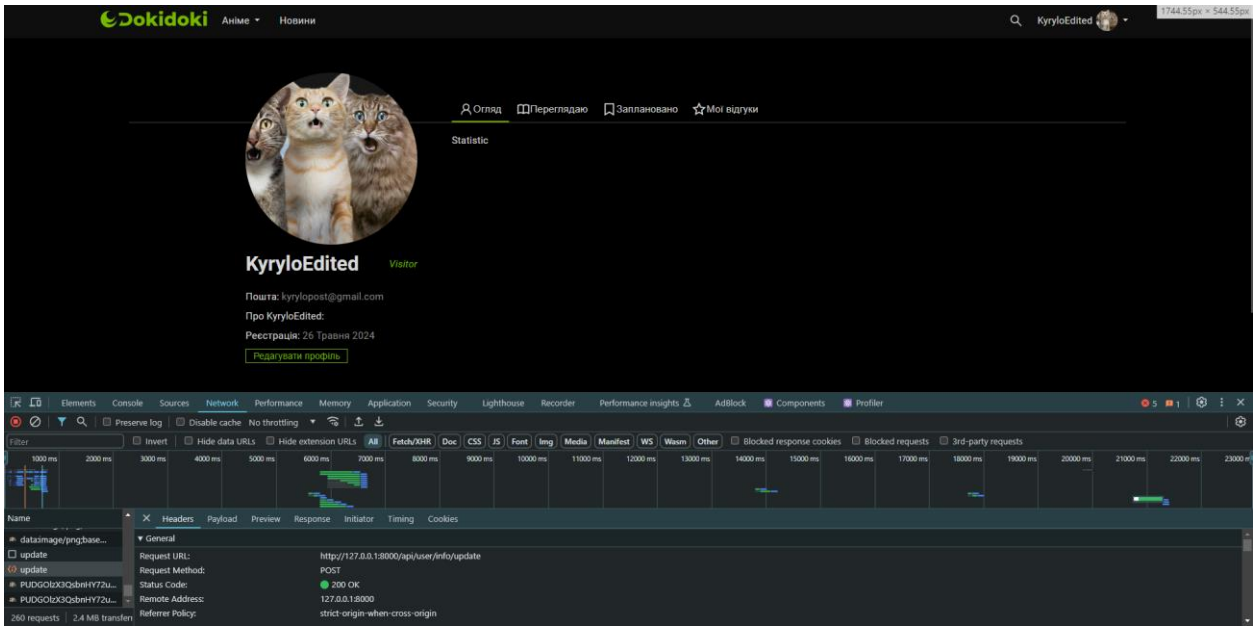

Рисунок В.9 – Профіль користувача після редагування

На рис. В.8 можна побачити, що користувач був зареєстрований, та його дані були надіслані з сервера, для відображення на сторінці, а на рис. В.9, що дані для редагування були успішно відправлені на сервер після чого автоматично змінилися на сторінці.

Перевірка можливості додавання коментарів та обмеження без необхідних прав буде відбуватися з щойно створеного акаунту. Якщо користувач увійшов в систему, у нього повинна бути можливість додавати

коментарів, якщо користувач не авторизований – тільки перегляд вже існуючих, без можливості додати новий.

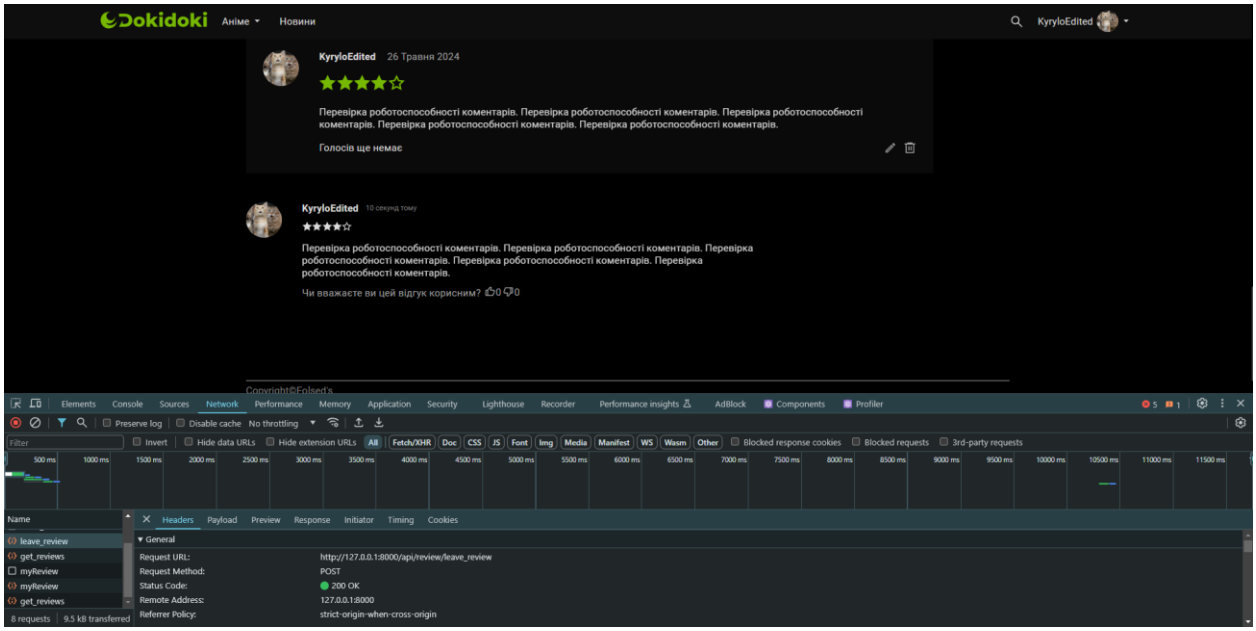

Рисунок В.10 – Можливість додавати коментар

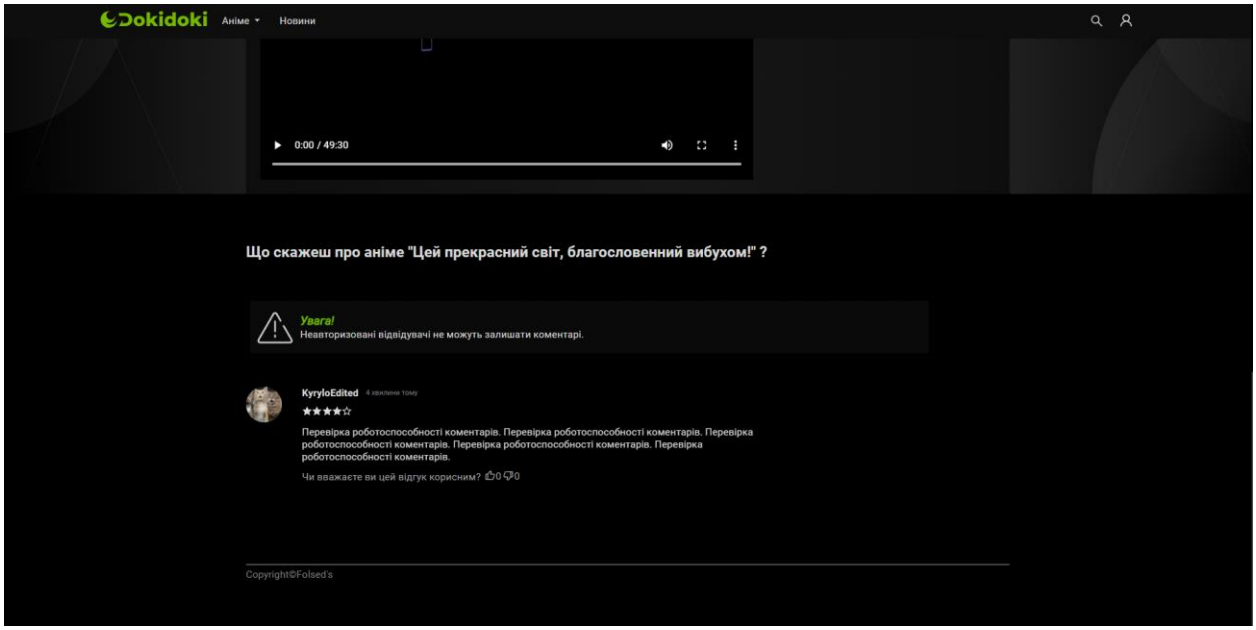

Рисунок В.11 – Перевірка ролі користувача

На рис. В.10 і В.11 можна побачити, що все працює коректно, користувач може додати новий коментар тільки в тому випадку, якщо він увійшов в систему.

Перевірка функціональності пошукового механізму буде відбуватись по ключовому слову або частині слова за допомогою строки пошуку з лівого боку хедера. Фільтрація та сортування винесені в окреме меню, для швидкого доступу, меню можна відкрити натиснувши на кнопку «Аніме» в лівій частині хедеру. В пошуковій строчці було введено «Ця фарф», відповідно до цього набору символів повинно бути виведено всі схожі результати. По натисканню на кнопки жанрів, або списків відсортованого контенту, повинно бути виведено всі серіали з відповідними перевірками.

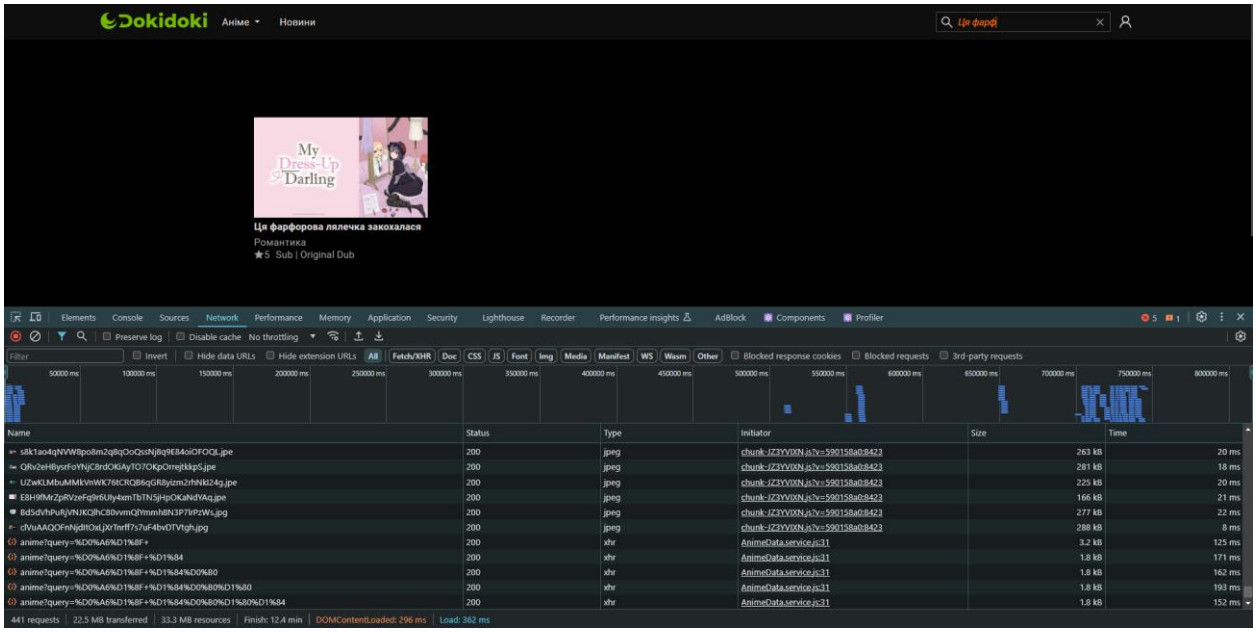

Рисунок В.12 – Перевірка пошуку

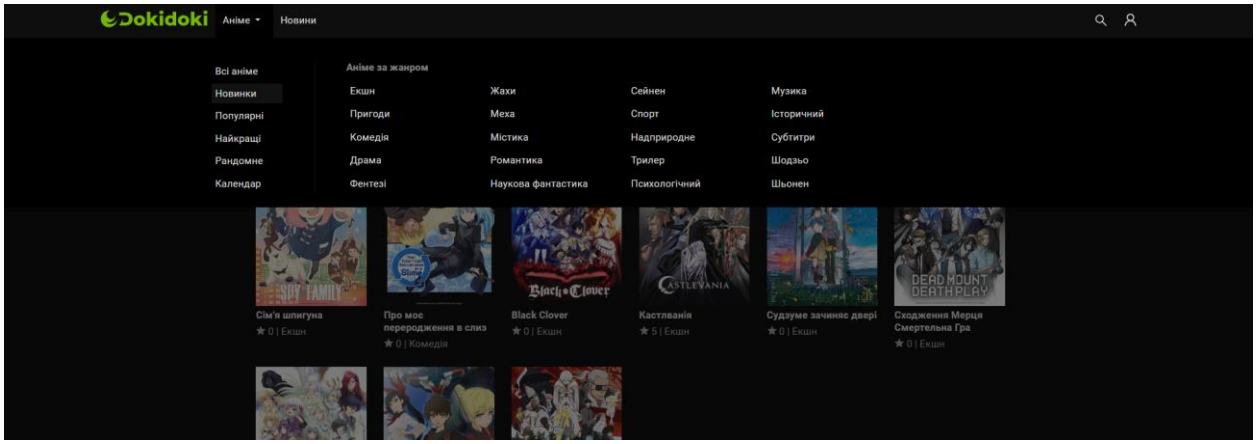

Рисунок В.13 – Меню сортування серіалів

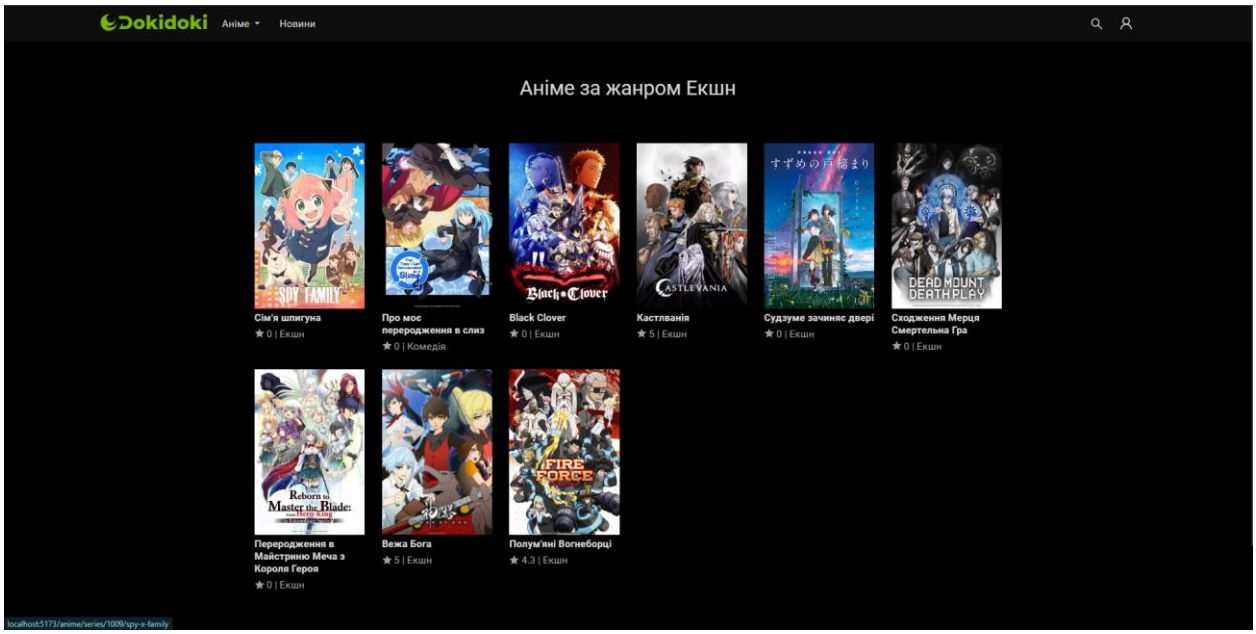

Рисунок В.14 – Всі серіали по вибраному сортуванню

На рис. В.12 можна побачити, що через маленьку кількість даних в базі даних, список серіалів зі схожими символами складається лише з одного об’єкту, але пошук працює коректно. рис. В.13 показує список сортування серіалів за різними критеріями, а рис. В.14 надає докази, що сортування працює коректно.

Виконавець: студент групи КІ-41, Колганов К.Є.