Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Комп'ютерних наук і технологій
(повне найменування факультету)

Комп'ютерні системи та мережі
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалаврський
(ступінь вищої освіти)

на тему: РОЗРОБКА ВЕБСИСТЕМИ ПРЕДСТАВЛЕННЯ АНІМЕ/МАНГИ ІЗ ВИКОРИСТАННЯМ ФРЕЙМВОРКУ NEXTJS

Виконав(ла): студент(ка) 4 курсу,
групи КНТ-512сп

Спеціальності
123 Комп'ютерна інженерія
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерна інженерія
(назва освітньої програми (спеціалізації)

ПАЛЄНОВ Д.А.
(ПРІЗВИЩЕ та ініціали)

Керівник КИРИЧЕК Г.Г.
(ПРІЗВИЩЕ та ініціали)
Рецензент КОЗІНА Г.Л.
(ПРІЗВИЩЕ та ініціали)

2025

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра комп'ютерних систем та мереж
Ступінь вищої освіти бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри КУДЕРМЕТОВ Р.К.
_____
«14» квітня 2025 року

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

ПАЛЄНОВА Данила Андрійовича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка вебсистеми представлення аніме/манги із використанням фреймворку NEXTJS.

керівник проєкту (роботи) к.т.н., доцент, КИРИЧЕК Г.Г.,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від « 08 » квітня 2025 року № 151
2. Строк подання студентом проєкту (роботи) 01.06.2025 р.
3. Вихідні дані до проєкту (роботи) дані ресурсів мережі інтернет, літературні та електронні джерела, перелік використовуваних програмних засобів: VS Code, Next.js, TypeScript, Firebase, Firestore, React, Cloudinary, API, Redux, JavaScript, Node.js

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Аналіз предметної області та визначення функцій системи;
2) Моделювання програмного забезпечення;
3) Проєктування вебсистеми;
4) Реалізація вебсистеми AniMore.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	КИРИЧЕК Г.Г.		
Нормоконтроль	ЩЕРБАК Н.В.		

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз предметної області	до 18.04.2025	
2	Визначення та аналіз вимог до програмного	до 22.04.2025	
	забезпечення		
3	Розробка специфікації вимог SRS	до 24.04.2025	
4	Розробка діаграми варіантів використання	до 28.04.2025	
5	Розробка діаграми послідовності	до 01.05.2025	
6	Розробка описів прецедентів	до 05.05.2025	
7	Проєктування архітектури програмного	до 12.05.2025	
	забезпечення		
8	Проєктування інтерфейсу користувача	до 22.05.2025	
9	Оформлення пояснювальної записки	до 25.05.2025	
10	Проходження нормоконтролю	до 01.06.2025	
11	Перевірка на наявність академічного плагіату	до 03.06.2025	
12	Проходження рецензування	до 10.06.2025	

Студент(ка) _____ Данило ПАЛЄНОВ
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Галина КИРИЧЕК
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 82 с., 3 табл., 23 рис., 5 дод., 24 джерел.

WEB-ПЛАТФОРМА, АВТОРИЗАЦІЯ, FIREBASE, CLOUDINARY, КОМЕНТАРІ, ВІДЕОПЛЕЄР, АДАПТИВНИЙ ІНТЕРФЕЙС, АНІМЕ, NEXTJS

Об’єкт розробки – веб-платформа для перегляду аніме та манги.

Мета роботи – розробка сучасної веб-платформи AniMore для автоматизації процесу пошуку, перегляду, оцінювання та обговорення аніме, а також для створення персоналізованого користувацького досвіду.

В ході розробки було проаналізовано технічне завдання. Проведено проєктування архітектури платформи, в результаті чого сформовано структуру бази даних, визначено основні алгоритми роботи сайту та побудовано макети ключових сторінок. Для зберігання даних обрано хмарну базу даних Firestore (Firebase), а для зберігання та оптимізації зображень – сервіс Cloudinary. Клієнтську частину розроблено з використанням бібліотеки React та фреймворку Next.js, що забезпечує швидке завантаження сторінок і SEO-оптимізацію. Серверна логіка реалізована через інтеграцію з Firebase та зовнішніми API.

В результаті розроблена веб-платформа дозволяє користувачам переглядати аніме та мангу, додавати їх до улюбленого, залишати коментарі, переглядати відео-епізоди, а також редагувати власний профіль із можливістю завантаження аватара. Інтерфейс платформи є адаптивним, зручним та сучасним, що забезпечує позитивний користувацький досвід на різних пристроях.

ABSTRACT

Explanatory note to the bachelor's thesis: 82 p., 3 tables, 23 figures, 5 app., 24 sources.

WEB-PLATFORM, AUTHORIZATION, FIREBASE, CLOUDINARY, COMMENTS, VIDEO PLAYER, ADAPTIVE INTERFACE, ANIME, NEXTJS

The object of development is a web platform for watching anime and manga.

The purpose of the work is to develop a modern web platform AniMore to automate the process of searching, viewing, rating, and discussing anime, as well as to create a personalized user experience.

During the development process, we analyzed the terms of reference. The platform architecture was designed, which resulted in the formation of the database structure, the main algorithms of the site, and the construction of key page layouts. We chose the Firestore cloud database for data storage, and the Cloudinary service for image storage and optimization. The client side was developed using the React library and the Next.js framework, which ensures fast page loading and SEO optimization. The server logic is implemented through integration with Firebase and external APIs.

As a result, the developed web platform allows users to watch anime and manga, add them to their favorites, leave comments, watch video episodes, and edit their own profile with the ability to upload an avatar. The platform's interface is responsive, user-friendly and modern, ensuring a positive user experience on various devices.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД	–	База даних
СКБД	–	Система керування базами даних
API	–	Application Programming Interface, інтерфейс прикладного програмування
GUI	–	Graphical User Interface, Графічний інтерфейс користувача
IDE	–	Integrated Development Environment, Інтегроване середовище розробки
UI	–	User Interface, Інтерфейс користувача
CRUD	–	Create, Read, Update, Delete (операції створення, читання, оновлення, видалення)
Anilibria	–	відкритий API для аніме-контенту
MangaDex	–	відкритий API для манга-контенту

ВСТУП

У сучасному цифровому середовищі розваги та дозвілля дедалі більше переходять в онлайн-формат. Особливо це стосується аніме та манги — культурних продуктів, що об’єднують мільйони фанатів у всьому світі. Швидкий доступ до відео й ілюстрованих томів, зручна навігація, приємний інтерфейс і соціальні функції стали ключовими чинниками успішного сервісу. У цьому контексті зростає потреба у комплексній платформі, яка забезпечить повний цикл фан-досвіду: від перегляду улюблених серіалів до зручного читання манги й взаємодії з іншими користувачами.

Дипломна робота присвячена розробці вебзастосунку “AniMore”**,** основним призначенням якого є створення єдиної платформи для перегляду аніме та читання манги. Цей застосунок поєднує високоякісний відеоплеєр із підтримкою адаптивного стримінгу (HLS/DASH) [1], автоплей, вибору якості, субтитрів і дубляжу, а також манґа-читалку з можливістю масштабування зображень, закладками, нічним режимом. Інтуїтивно зрозумілий інтерфейс, система каталогізації за жанрами й новинками, модуль рекомендацій і соціальні функції (рейтинги, коментарі) забезпечують повноцінну взаємодію між користувачами й підвищують їхню залученість.

Актуальність теми обумовлена стрімким зростанням популярності аніме та манги й відсутністю вітчизняних платформ із комплексним і простим у використанні функціоналом. Більшість існуючих сервісів або мають обмежену бібліотеку, або складний UX, що відлякує новачків, або позбавлені соціальних інструментів. Створення “Animore” є своєчасним та доцільним заходом для задоволення потреб сучасного цифрового суспільства та розвитку локальної фан-спільноти.

Мета роботи – спроєктувати та реалізувати вебзастосунок, який забезпечить комфортний перегляд аніме, зручне читання манги, інтуїтивну навігацію та соціальну взаємодію.

Для досягнення цієї мети необхідно:

- проаналізувати ринок онлайн-сервісів для аніме й манги;
- визначити функціональні й UX-вимоги до плеєра та читалки;
- обрати стек вебтехнологій;
- спроєктувати архітектуру системи та модель даних;
- реалізувати відеоплеєр і манґа-читалку;
- інтегрувати систему каталогізації, рекомендацій та соціальних функцій;
- провести функціональне та навантажувальне тестування.

Об’єкт дослідження – інформаційна система для доставки медіаконтенту (аніме та манга) й організації взаємодії користувачів. Предмет дослідження – технології та методи реалізації відеоплеєра, манґа-читалки й UX/UI-інтерфейсу в межах вебзастосунку [2].

Практичне значення “Animore” полягає в тому, що платформа спрощує доступ до аніме й манги, об’єднує функціонал перегляду та читання в одному місці, підвищує залученість користувачів і дозволяє контент-менеджерам ефективно керувати бібліотекою. Це сприятиме розвитку локальної спільноти фанатів і покращенню якості сервісу в ніші онлайн-розваг.

Структура роботи включає теоретичну частину (аналіз ринку, UX/UI-принципи), проєктну частину (архітектура, дизайн-макети, модель даних), практичну реалізацію (відеоплеєр, читалка, інтерфейс) та тестування (функціональне, навантажувальне) з оцінкою ефективності платформи “Animore”.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИЗНАЧЕННЯ ФУНКЦІЙ СИСТЕМИ

1.1 Огляд предметної області

Онлайн-платформи для перегляду аніме та читання манги поєднують у собі мультимедійні та соціальні сервіси, покликані надати фанатам повний спектр інтерактивного досвіду. До таких систем належать кілька взаємопов’язаних компонентів:

- відеоплеєр із адаптивним стримінгом (HLS/DASH) [3], що дозволяє обирати якість відтворення залежно від швидкості інтернет-з’єднання, підтримує автоплей наступних серій, субтитри та дубляж;
- манґа-читалка з можливістю масштабування зображень, прокрутки сторінок, закладками, нічним режимом та автоматичного збереження прогресу читання;
- система каталогізації та пошуку, що класифікує контент за жанрами, студіями, роком випуску, популярністю й новинками, а також дозволяє здійснювати текстовий і фільтровий пошук;
- модуль рекомендацій, який формує персоналізовані добірки на основі історії переглядів, оцінок і вподобань користувача;
- соціальні функції: рейтинги, коментарі, обговорення, можливість додавати контент до “улюбленого” та ділитися в соціальних мережах;
- адміністративна панель для контент-менеджерів, що включає завантаження нових серій і глав, модерацію коментарів, аналітику переглядів і читання, налаштування користувацьких ролей і прав доступу.

Ринок аніме-стримінгу та манга-читалок демонструє стрімке зростання: глобальна аудиторія налічує десятки мільйонів активних користувачів, а попит диктує високі вимоги до якості відтворення, швидкості завантаження та зручності інтерфейсу. Існуючі рішення (Anilibria, Crunchyroll, MangaDex) забезпечують широкий каталог, але водночас стикаються з низкою обмежень: фрагментарний

функціонал (відео й манга розділені на різних платформах), відсутність автоплей або адаптивного стримінгу, прості читалки без збереження прогресу та нічного режиму, слабка система рекомендацій і соціальних взаємодій.

Головною проблемою є відсутність єдиної, комплексної платформи, що закривала б усі потреби фанатів аніме й манги. Користувачі змушені перемикатися між різними сайтами, втрачати прогрес читання чи перегляду, миритися з буферизацією відео або обмеженими налаштуваннями плеєра. Для контент-менеджерів ці розпорошені сервіси не дають зручних аналітичних інструментів для оптимізації наповнення й маркетингових кампаній.

У світлі вищезазначеного виникає потреба в розробці вебзастосунку “Animore”, який інтегрує всі необхідні компоненти в єдиному інтерфейсі: високоякісний відеоплеєр із автоплей, вибором якості, субтитрами та дубляжем; просунуту манґа-читалку із закладками, масштабуванням, нічним режимом і збереженням прогресу; гнучку систему каталогізації, фільтрації та персональних рекомендацій; соціальні інструменти для обговорень, рейтингів і обміну контентом; а також аналітичну панель для ефективного управління бібліотекою та користувацькою активністю. Завдяки модульній архітектурі “Animore” легко масштабується й доповнюється новими функціями, відповідаючи на зміни ринку та потреби користувачів.

1.2 Дослідження аналогів вебсайтів для аніме та манги

У процесі вивчення наявних рішень на ринку вебзастосунків, призначених для перегляду аніме та читання манги, акцент робиться на кількох ключових складових. Ці складові дозволяють краще зрозуміти сучасні технології та підходи до реалізації таких сервісів, виявити поточні тренди й недоліки існуючих платформ, а також сформувати концепцію власного інноваційного рішення. До основних аспектів дослідження належать [13]:

- функціональні можливості – аналізуються типові функції сучасних сервісів, зокрема можливість вибору озвучки або субтитрів, перемикання якості відео, підтримка HLS-стрімів, перегляд у повноекранному режимі, система закладок, підтримка списків переглянутого. Для манги оцінюються такі функції, як гортання сторінок, нічний режим, можливість вибору режиму перегляду (по сторінках, вертикальний скрол тощо) [6];

- технологічна основа – досліджуються інструменти та технології, що використовуються конкурентами, включно з мовами програмування (наприклад, JavaScript, TypeScript), фреймворками (React, Vue), способами реалізації відеоплеєрів (наприклад, Video.js з підтримкою HLS), системами керування контентом, базами даних та архітектурними підходами (моноліт, мікросервіси). Це дозволяє визначити оптимальну технічну базу для створення швидкого, стабільного та масштабованого застосунку;

- досвід користувачів та дизайн (UX/UI) – аналізується структура інтерфейсу [12]: зручність пошуку аніме або манги, логіка навігації між серіями, епізодами, сторінками та розділами. Важливу роль відіграє адаптивність дизайну під мобільні пристрої, чіткість іконок, контрастність, мінімалістичність або навпаки — наявність інтерактивних елементів (рейтинги, коментарі, індикатори прогресу [5];

- цільовий сегмент ринку – визначення основної аудиторії, зокрема молодіжної групи 12 – 30 років, фанатів японської культури, поціновувачів високоякісного перекладу, субтитрів та зручного доступу до архівів. Це дозволяє зрозуміти, на які функції варто зробити акцент: легкий доступ до улюбленого контенту, швидке завантаження, можливість створення облікового запису, персоналізовані рекомендації;

- відгуки користувачів та репутація – аналіз думок аудиторії в коментарях, оглядах або на тематичних форумах дозволяє виявити найбільш часті скарги: поганий переклад, повільна робота плеєра, реклама, відсутність мобільної адаптації. У той же час позитивні відгуки дозволяють визначити сильні сторони, які варто взяти за основу при розробці власного продукту.

Такий комплексний підхід до вивчення тематики вебзастосунків для перегляду аніме й манги дозволяє створити сервіс, що максимально відповідає запитам цільової аудиторії, поєднуючи зручність, функціональність та естетичну привабливість. Урахування поточних технологічних рішень та досвіду користувачів забезпечить розробку сучасного продукту з високим рівнем задоволеності глядачів та читачів.

1.2.1 Crunchyroll

Crunchyroll — одна з найвідоміших міжнародних платформ для перегляду аніме в режимі онлайн[15]. Сервіс надає легальний доступ до тисяч серій аніме, а також до манґи, дорам та форумів спільноти. Платформа активно співпрацює з японськими студіями та володіє ліцензіями на показ контенту одразу після прем'єри в Японії.

Переваги Crunchyroll:

- ліцензований контент: користувачі можуть бути впевнені в легальності перегляду, а творці контенту — в отриманні прибутку;
- **s**imulcast та ексклюзиви: деякі серіали з’являються на Crunchyroll через кілька годин після ефіру в Японії;
- наявність субтитрів різними мовами: платформа підтримує десятки мов, що робить її доступною для глобальної аудиторії;
- мобільні застосунки та консольна підтримка: можна переглядати аніме з будь-якого пристрою.

Недоліки Crunchyroll:

- обмежений україномовний контент: українські субтитри відсутні, а переклад здебільшого англійський або російський;
- регіональні обмеження: деякі тайтли недоступні для перегляду в Україні через ліцензійні обмеження;
- платна підписка для HD-якості: безкоштовна версія має рекламу і обмежену якість відео.

Crunchyroll є прикладом успішної моделі ліцензованого поширення аніме з високим рівнем функціональності. Його досвід варто враховувати при створенні

українського продукту з підтримкою локалізованого контенту та без регіональних обмежень.

1.2.2 Anilibria

Anilibria — це популярний неофіційний український і російськомовний онлайн-сервіс, який надає доступ до озвученого аніме[16]. Платформа відома якісними озвученнями власного виробництва та зручною системою перегляду з вибором серії, озвучки та якості.

Переваги Anilibria:

- безкоштовний доступ: сайт не вимагає оплати для перегляду будь-якого контенту;

- кілька доріжок озвучки та якості: можна обрати мову та роздільну здатність потоку;

- зручний відеоплеєр: з підтримкою HLS, перемикання між серіями, швидкістю відтворення та темною темою;

- активна спільнота: користувачі можуть коментувати та обговорювати епізоди, залишати відгуки.

Недоліки Anilibria:

- відсутність ліцензій: правовий статус платформи викликає питання щодо легальності розміщеного контенту;

- залежність від реклами: хоча сайт безкоштовний, деякі елементи монетизації можуть впливати на досвід користувача;

- вузький фокус: платформа орієнтована винятково на аніме, без додаткових форматів, таких як манґа чи ранобе.

Anilibria є прикладом гнучкого технічного рішення з акцентом на зручність та локалізацію. Ідеї мультимовного вибору озвучки та плеєра високого рівня можуть бути використані при створенні власного легального продукту.

1.2.3 MangaDex

MangaDex — міжнародна платформа для читання манґи з підтримкою великої кількості мов[17]. Це спільнотний некомерційний проєкт, який надає

користувачам можливість самостійно завантажувати переклади, створювати команди перекладачів і слідкувати за оновленнями улюблених серій.

Переваги MangaDex:

- багатомовність: підтримка понад 30 мов, включно з українською (хоча в обмеженій кількості тайтлів);
- командна система перекладу: можливість створювати або приєднуватися до фанатських груп, які займаються перекладом;
- гнучка система пошуку й фільтрації: пошук за жанрами, демографією, статусом перекладу тощо;
- відсутність реклами: сайт працює без комерційної монетизації.

Недоліки MangaDex:

- відсутність офіційних ліцензій: весь контент розміщується фанатами без погодження з правовласниками;
- текстовий інтерфейс: інтерфейс менш сучасний, орієнтований на функціональність, а не дизайн;
- проблеми з доступністю: у минулому платформа тимчасово припиняла роботу через атаки або зміну хостингу.

MangaDex є джерелом натхнення для створення гнучкого, децентралізованого інструменту для читання манґи з активною спільнотою. Досвід платформи доводить ефективність моделей із залученням користувачів до створення контенту.

1.2.4 YummiAnime

YummiAnime — неофіційна онлайн-платформа для безкоштовного перегляду аніме з підтримкою декількох озвучок та субтитрів, що пропонує весь каталог серій і сезонів підряд без реєстрації[18]. Контент викладається в хорошій якості (720p/1080p) із можливістю обрати оригінальну або фанатську озвучку.

Переваги YummiAnime:

- безкоштовний доступ без підписки та плати, що залучає широку аудиторію користувачів;

- кілька доріжок озвучки та субтитрів, зокрема російська, англійська та фанатські варіанти, що підвищує гнучкість перегляду;

- якісні відеопотоки без буферизації: сайт використовує адаптивну доставку контенту через HLS/DASH, забезпечуючи плавне відтворення навіть при змінній швидкості інтернету;

- активна спільнота на Пікабу, де публікуються офіційні новини та анонси сервісу Пикабу;

- офіційний канал у Telegram, група в Discord для швидкого реагування на технічні питання та інформування про оновлення;

- простий інтерфейс з каталогом серій, фільмів та OVA, що дозволяє легко знаходити нові релізи й класику;

- висока репутація серед користувачів: сайт потрапив до топ-5 кращих ресурсів для перегляду аніме в 2024 році на Пікабу.

Недоліки YummiAnime:

- відсутність офіційних ліцензій: весь контент розповсюджується без згоди правовласників, що створює правові ризики;

- інтрузивна реклама у вбудованому плеєрі, яка інколи загружається перед кожною серією та може заважати перегляду;

- блокування домену в ряді країн (зокрема РФ), що вимагає використання VPN або дзеркал для доступу;

- відсутність манґа-читалки, оскільки платформа орієнтована виключно на відеоконтент.

YummiAnime демонструє успіх неофіційних рішень із фокусом на зручність користувачів і широку локалізацію, але потребує уваги до легальності контенту та нав’язливої реклами.

Також створив таблицю для порівняння функціональності між моїм проектом та іншими вебзастосунками, зображено у таблиці 1.1.

Таблиця 1.1 – Таблиця порівняння функціональності

Функція / Платформа	Animore (мій проект)	Anilibria	YummyAnime	Crunchyroll	MangaDex
Адаптивний HLS/DASH-плеєр	Повноцінний HLS/DASH-плеєр з автоплей, вибором якості, субтитрами й дубляжем	1080p та HEVC (x265/10-bit), але без HLS/DASH	HLS-стримінг, але без автоплей та субтитрів	Автоматичний підбір якості та ручний підбір	Немає(орієнтовано лише на мангу)
Рідер для манги	Масштабування, закладки, нічним режимом та автозбереженням прогресу	Немає	Немає	із вертикальним/горизонтальним скролом, базове збереження прогресу	Масштабування, закладки, нічний режим, збереження прогресу читання
Персональні рекомендації	Рекомендації за підписками на тайтли та тегами	Немає	Немає	Алгоритми на основі історії переглядів і вподобань	Рекомендації за підписками на тайтли та тегами
Соціальні функції	Коментарі, рейтинги, обговорення	Немає	Коментарі	Коментарі з “Spoiler” тегами та рейтинги	Форумні обговорення, коментарі, групові чати,
Модерація контенту	Черга модерації, фільтри порушень, логування дій	Модерація відгуків	Немає	Модерація відгуків і батьківський контроль	Групова модерація, фільтрація за тегами, лог дій модераторів

1.3 Специфікація вимог до ПЗ

Розроблюване програмне забезпечення є вебзастосунком, призначеним для перегляду аніме та читання манґи в різних форматах — від коротких OVA та фільмів до повноцінних серіалів і багатотомних томів манґи. Основною метою створення цієї системи є забезпечення інтуїтивно зрозумілого й зручного способу

пошуку, перегляду та читання контенту, а також управління персональними списками і налаштуваннями сповіщень про нові епізоди чи глави.

Програмний продукт буде орієнтований на максимальну інтеграцію з сучасними вебтехнологіями, без підтримки десктопних клієнтів. Це дозволить зосередитися на універсальності доступу через веббраузер, адаптивності інтерфейсу та зменшенні загальної складності розгортання.

В межах проекту розробка обмежується створенням веборієнтованого застосунку, що працює в браузері з адаптивним дизайном для десктопів і мобільних пристроїв. У межах даного проєкту в майбутньому передбачається розробка нативних мобільних додатків (iOS/Android). Основна увага зосереджена на доступності через веббраузер, швидкодії програвання відео та зручності читання манґи.

Система призначена для поціновувачів аніме та манґи і використовується для:

- перегляду аніме (серіали, фільми, OVA) з підтримкою перемикання якості (480p, 720p, 1080p), озвучки та субтитрів;
- читання манґи у режимі посторінкового гортання або вертикального скролу з автозбереженням прогресу;
- управління особистими списками й історією споживання контенту;
- отримання рекомендацій на основі жанрів, рейтингу та активності користувача;
- спілкування в спільноті: оцінка контенту, коментарі, обмін посиланнями на улюблені серії або томи.

1.3.1 Категорії користувачів та їх потреби

У межах системи передбачено три основні типи користувачів, кожен з яких має власні функціональні потреби та модель взаємодії з вебзастосунком:

Звичайні користувачі (гості / зареєстровані) - це люди, які заходять на сайт щоб знайти, переглянути або прочитати аніме/манґу.

Ключові потреби:

- зручний каталожний перелік із постерами, жанрами та рейтингами;

- швидкий пошук і фільтрація за назвою, жанром, статусом (онгоінг, завершено), мовою озвучки чи перекладу;
- інтуїтивний відеоплеєр (перемикання якості, субтитрів/доріжок) та рідер манґи (горизонтальний/вертикальний режими, нічний режим);
- персональні списки, історія перегляду/читання;
- можливість швидко залишити відгук: оцінка контенту, коментарі.

Модерація сайту - це роль відповідальна за перевірку та управління всім користувацьким і завантаженим контентом перед його публікацією, а також за підтримку порядку в коментарях і на форумах.

Ключові потреби:

- черга модерації: інтерфейс зі списком нових завантажень (епізоди, глави манґи, субтитри), коментарів та відгуків, що очікують на перевірку;
- інструменти прийняття рішень: можливість схвалити, відхилити або повернути контент на доопрацювання з коментарем;
- фільтрація порушень: автоматизовані сповіщення про потенційні порушення (спам, нецензурщина, порушення авторських прав) із можливістю швидкого огляду та блокування;
- управління користувачами: функції попередження, тимчасового або постійного блокування порушників, перегляд історії їхніх дій;
- логування активності: збереження записів усіх модераторських дій для аудиту та аналітики;
- аналітика роботи модерації: статистика часу обробки запитів, кількості схвалених/відхилених матеріалів, навантаження на модераторів.

Адміністратори системи - це фахівці, відповідальні за безпеку, стабільність та розвиток платформи в цілому.

Ключові потреби:

- управління ролями та правами (гості, зареєстровані, куратори, модератори);
- модерація контенту та коментарів: блокування спаму, нецензурщини, плагіату;

- моніторинг працездатності сервісу: логи помилок, навантаження на сервери, використання CDN;
- аналітика користувацької активності (кількість переглядів, залученість у коментарі, конверсія гостей у зареєстровані акаунти);
- налаштування системних параметрів: ліміти на завантаження, параметри кешування, шаблони email-сповіщень.

1.3.2 Основні функції системи

У межах розроблюваного вебзастосунку передбачено реалізацію ключових функціональних можливостей, які забезпечують повний цикл взаємодії користувача з платформою.

Функція авторизації дозволяє зареєстрованим користувачам безпечно входити в систему для перегляду аніме та читання манґи. Процес та компоненти проходження авторизації:

- вхідні дані**:** електронна пошта або нікнейм, пароль.

Вихідні дані:

- успішний вхід із видачею JWT-токена або встановленням сесії;
- повідомлення про помилку: «неправильний логін/пароль».

Функціональні вимоги:

- перевірка коректності формату email/нікнейму та складності паролю);
- безпечне зберігання паролів у хешованому вигляді;
- оповіщення користувача на email про спробу входу з нового пристрою.

Функція реєстрації створює обліковий запис для доступу до персональних списків і сповіщень.

Вхідні дані:

- обов’язково: email, пароль;
- рекомендовано: нікнейм, вибір улюблених жанрів (для першочергових рекомендацій).

Вихідні дані:

- підтвердження успішної реєстрації з подальшим переходом на сторінку підтвердження email;
- повідомлення про помилки: «email уже зареєстровано», «пропущене обов’язкове поле».

Функціональні вимоги:

- валідація формату email та складності паролю (мінімум 8 символів, цифри + букви);
- перевірка унікальності email/нікнейму в БД;
- відправка листа з підтвердженням (одноразовий посилання-логін на 24 години);
- збереження вибору жанрів для персоналізації рекомендацій із першого входу.

Особистий кабінет дає зареєстрованому користувачеві зручний доступ до всіх персональних налаштувань і даних про взаємодію з контентом.

Вхідні дані: ідентифікатор користувача (через сесію або JWT-токен).

Вихідні дані:

- списки контенту: для аніме та манґи;
- рекомендації: персоналізовані пропозиції на основі жанрів і оцінок;
- налаштування сповіщень: підписки на прем’єри нових епізодів, оновлення глав манґи, системні повідомлення.

Функціональні вимоги:

- відображення лише актуальних даних, синхронізованих із сервером у режимі реального часу;
- можливість одразу перемістити епізод/главу між списками;
- швидке видалення або скасування позначки в будь-якому списку;
- гарантія конфіденційності: інші користувачі не бачать приватні списки та налаштування.

Управління персональними даними - цей модуль дозволяє користувачеві переглядати й змінювати власну інформацію, що зберігається в профілі.

Вхідні дані: нові значення полів (ім’я, email, нікнейм, аватар, улюблені жанри).

Вихідні дані: повідомлення про успішне оновлення або список помилок (некоректний формат, email вже використовується тощо).

Функціональні вимоги:

- валідація: перевірка формату email, розміру/формату файлу аватара, обов’язковості полів;
- безпека: зміни зберігаються тільки після повторного введення паролю або підтвердження через email;
- захист даних: особиста інформація шифрується в базі даних і передається тільки через HTTPS;
- миттєве оновлення: після збереження даних інтерфейс кабінету оновлюється без перезавантаження сторінки.

Перегляд списку доступних аніме/манґи - надає всім відвідувачам доступ до каталогів із постерами та базовими метаданими.

Вхідні дані: жодних (гість або зареєстрований користувач).

Вихідні дані: перелік тайтлів із назвою, постером, жанрами, рейтингом та статусом (ongoing/finished).

Функціональні вимоги:

- динамічне підвантаження (infinite scroll або пагінація);
- синхронізація з актуальною БД;
- відображення лише доступного контенту (фільтрація прибраних модераторами).

Пошук тайтлів за заданими критеріями
Забезпечує користувачам точний пошук і фільтрацію каталогу.

- Вхідні дані: ключові слова (назва), жанри, статус, мінімальний рейтинг, мова озвучки/перекладу.
- Вихідні дані: відфільтрований перелік тайтлів з елементами каталогу.

Функціональні вимоги:

- автозаповнення під час введення (type-ahead);

- комбінована фільтрація за декількома параметрами;
- відображення кількості знайдених результатів.

1.3.3 Функціональність вебзастосунку для модерації

Створення та публікація тайтлів (для модераторів). Дозволяє модераторам додавати нові аніме/манґа в систему.

Вхідні дані: файли відео або скановані зображення, метадані (назва, оригінальна назва, жанри, опис, студія/автор, статус).

Вихідні дані: новий запис у каталозі зі статусом «на модерації» або «опубліковано».

Функціональні вимоги:

- валідація розмірів і форматів файлів (mp4/HLS для відео, JPEG/PNG для зображень);
- обов’язкові поля: назва, жанри, опис;
- можливість попереднього перегляду запису перед публікацією.

Управління створеними тайтлами (для модераторів) - надає інструменти редагування вже опублікованих записів.

Вхідні дані: ідентифікатор тайтлу та нові значення полів (опис, постер, статус тощо).

Вихідні дані: оновлений запис із підтвердженням збереження.

Функціональні вимоги:

- історія змін (логування попередніх версій);
- можливість відкликати тайтл (зняти з публікації);
- валідація оновлених даних.

Також розробив графічну модель вебзастосунку, зображено на рис.1.1

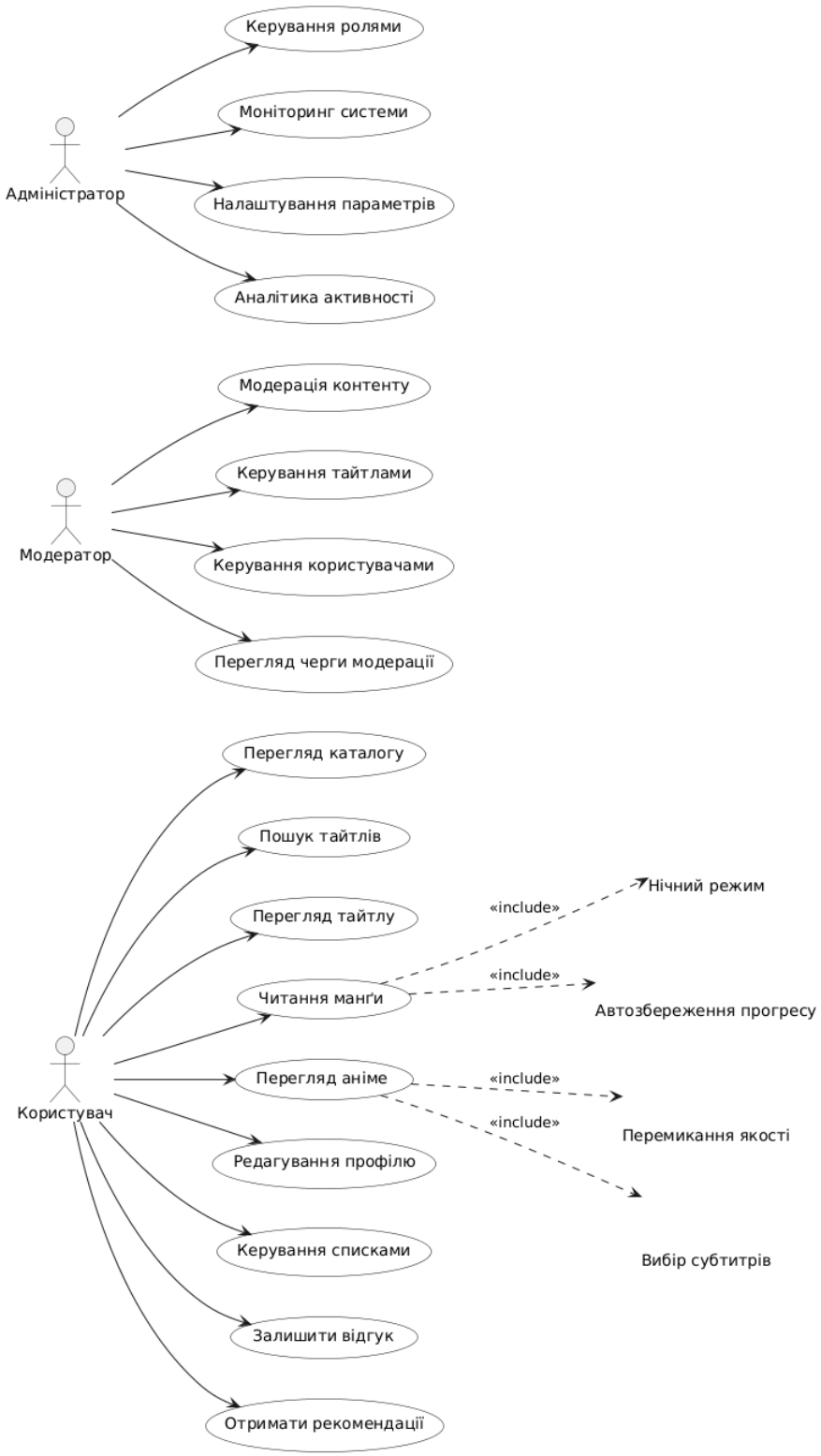


Рисунок 1.1 – Графічна модель системи

1.3.4 Мобільна версія сайту

Платформа має адаптивну версію для мобільних пристроїв, яка забезпечує комфортне використання функціоналу сайту зі смартфонів та планшетів.

Вхідні дані: розмір екрану пристрою.

Результат: інтерфейс, оптимізований під мобільні екрани.

Ключові вимоги: реалізація адаптивного дизайну, збереження повної функціональності основного сайту, забезпечення функціональності всіх основних функцій сайту в мобільній версії, оптимізація швидкодії та продуктивності для мобільних пристроїв.

1.3.5 Архітектура та розробка системи

Система побудована за принципом монолітної архітектури, де вебзастосунок складається з взаємопов’язаних сервісів, що спільно використовують єдину базу даних. Така структура спрощує розгортання та супровід, особливо на етапі ранньої розробки.

Для розробки використовуються такі середовища:

- Visual Studio Code – для стеку та роботи з інтерфейсами;
- “Firebase” - для роботи з базами даних.

Комунікація між компонентами системи забезпечується через сучасні мережні протоколи, зокрема HTTPS, що гарантує безпечний обмін даними.

Основним інструментом адміністрування та роботи з базами даних є - Firebase. Він забезпечує: зручне створення та редагування структур бази даних [9], управління безпекою доступу, модифікацію, налаштування параметрів серверів, графічний інтерфейс для роботи з даними.

Для написання цього вебзастосунку використовувались наступні технології: JavaScript, Next.js, HTML5 (Video API для HLS/DASH), Video.js. Для бази даних: Firebase.

В основі фронтенду – фреймворк Next.js. Next.js — сучасний фреймворк на базі React, який поєднує переваги класичного React із додатковими можливостями серверного рендерингу, статичної генерації сторінок та зручної маршрутизації. Він дозволяє створювати не просто односторінкові застосунки, а повноцінні багатосторінкові платформи з високою продуктивністю та чудовою SEO-оптимізацією. Крім того, Next.js підтримує серверний рендеринг (SSR) та статичну генерацію (SSG), що дозволяє швидко завантажувати сторінки та робити

їх доступними для пошукових систем. Ще однією важливою перевагою Next.js є можливість створювати API-роути безпосередньо у проєкті, що спрощує інтеграцію з бекендом або зовнішніми сервісами. Також фреймворк має вбудовану підтримку TypeScript, CSS/SCSS-модулів, оптимізації зображень та інших сучасних інструментів для розробки.

Для зберігання та обробки даних у проєкті використовується хмарна платформа Firebase. Вона забезпечує аутентифікацію користувачів (через Firebase Authentication), а також зберігання інформації про профілі, списки аніме, коментарі та інші дані у Firestore — сучасній NoSQL базі даних. Це дозволяє уникнути розгортання власного серверного застосунку, спростити налаштування безпеки та гарантувати високу доступність даних.

1.3.6 Нефункціональні вимоги

Інтерфейс має бути максимально зручним та інтуїтивно зрозумілим для користувачів з різним рівнем технічної підготовки. Застосування сучасних принципів UI/UX дизайну забезпечує легку навігацію, доступ до основних функцій і приємний користувацький досвід.

Система повинна бути постійно доступною — 24 години на добу, 7 днів на тиждень, без тривалих збоїв чи технічних перерв, забезпечуючи надійний сервіс для всіх користувачів.

Програмне забезпечення має бути легко підтримуваним: із добре структурованим кодом, повною технічною документацією та налагодженим механізмом зворотного зв’язку для швидкого реагування на помилки та пропозиції користувачів.

Архітектура системи повинна дозволяти її розгортання на різних платформах і в різних середовищах з мінімальними змінами. Цього можна досягти шляхом компонентного підходу для фронтенду.

Система повинна швидко обробляти великі обсяги інформації та одночасні запити, забезпечуючи час відгуку не більше ніж 3 секунди навіть при високому навантаженні. Відеопотоки доставляються з підтримкою HLS/DASH із автоматичним підлаштуванням якості.

Функціонування системи має бути стабільним і безперервним. Необхідно мінімізувати кількість збоїв, помилок і відмов, особливо в критичних компонентах, що впливають на роботу користувача.

Система повинна гарантувати високий рівень захисту персональних даних. Передбачено використання сучасних методів шифрування, безпечної автентифікації на основі токенів, які зберігаються в середовищах, недоступних для клієнтської сторони.

Для більш точного уявлення, розробив dfd-діаграму(Data Flow Diagram) – діаграма, яка показує процеси, зовнішні сутності та потоки даних між ними, зображена на рис.1.2.

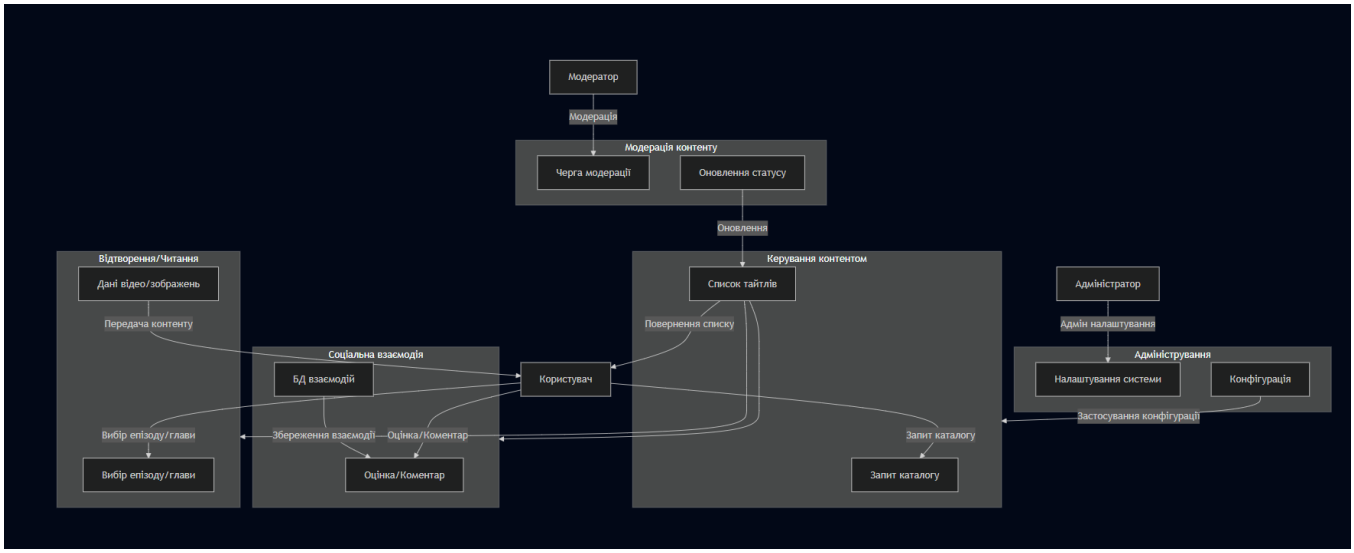


Рисунок 1.2 – DFD-діаграма “Animore”

1.4 Постановка завдань

У рамках практичної частини проєкту “Animore” передбачається поетапна реалізація основних функціональних блоків мультимедійної платформи та рідера манґи. Такий підхід дозволить забезпечити масштабованість, зручність використання й стабільну роботу сервісу на всіх етапах розвитку. Нижче наведено

ключові завдання, які повинні бути вирішені для успішного впровадження всіх компонентів проєкту:

- створення мультимедійної платформи – кастомного HTML5-відеоплеєра на базі Video.js із підтримкою адаптивного HLS/DASH, автоплеєм та вибором якості для забезпечення стабільного відтворення аніме-контенту за будь-яких параметрів мережі;
- впровадження рідера манґи з компонентами для посторінкового гортання й вертикального скролу, автозбереженням прогресу читання та нічним режимом;
- проєктування API для завантаження зображень із CDN і синхронізації стану рідера з бекендом (Firebase);
- розробка макета сайту у Figma з визначенням розміщення головного меню, каталогу, персонального кабінету та елементів керування плеєром і рідером;
- верстка сторінок із використанням CSS Grid та Flexbox на основі затвердженого макета, інтеграція відеоплеєра й рідера в єдину шаблонну структуру з коректним відображенням постерів, списків і форм;
- адаптація інтерфейсу шляхом налаштування відображення контролів відеоплеєра для мобільної та десктопної версій і визначення адаптивних точок-зламу для зручного читання манґи на будь-якому екрані;
- синхронізація стану між компонентами відеоплеєра й рідера для плавного та безперебійного переходу при зміні якості відео або сторінки манґи.

Планована реалізація всіх зазначених етапів гарантує створення єдиного, інтуїтивно зрозумілого та надійного рішення для фанатів аніме й манґи. Поетапна робота над кожним блоком дозволить своєчасно виявляти й усувати потенційні проблеми, а злагоджена інтеграція компонентів забезпечить високий рівень задоволеності користувачів і масштабованість платформи в майбутньому.

2 МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Опис ключових прецедентів взаємодії користувача з платформою

У роботі розглянуто два основні прецеденти взаємодії користувача з платформою AniMore: оновлення персональних даних у профілі та перегляд переліку улюблених аніме, зображено на рисунку 2.1. Обидва сценарії вимагають обов’язкової авторизації, що гарантує індивідуальний доступ до даних і персоналізований досвід користування системою.

Рисунок 2.1 – Діаграма взаємодії користувача з вебсистемою

Діаграма прецеденту для сценарію "Перегляд списку улюблених аніме" ілюструє взаємодію користувача з платформою AniMore під час отримання доступу до власного переліку улюблених аніме. Основним актором виступає користувач, який спочатку проходить процедуру авторизації в системі. Авторизація є обов’язковою умовою для персоналізованого доступу до даних, оскільки лише після підтвердження особи користувача система може ідентифікувати його та надати доступ до індивідуального списку улюблених аніме.

Після успішної авторизації користувач отримує можливість скористатися функцією перегляду списку улюблених аніме. Цей прецедент передбачає, що система, використовуючи ідентифікатор авторизованого користувача, здійснює запит до бази даних і повертає актуальний перелік аніме, які були додані до улюбленого. Таким чином, діаграма відображає залежність між прецедентами:

"Перегляд списку улюблених аніме" включає виконання прецеденту "Авторизація", що забезпечує захист персональних даних та індивідуальний досвід користування платформою.

2.1.1 Прецендент «Перегляд аніме»

Прецедент "Перегляд аніме" дозволяє користувачу ознайомитися з повним переліком аніме, доступних на платформі AniMore, діаграму прецендента зображено на рисунку. Після переходу на відповідну сторінку користувач отримує доступ до інтерфейсу, де реалізовано функціонал пошуку, фільтрації та сортування аніме за різними критеріями (жанр, рік випуску, рейтинг тощо). При завантаженні сторінки фронтенд-частина ініціює запит до бази даних Firestore для отримання актуального списку аніме. Система повертає дані, які обробляються на клієнті та відображаються у вигляді карток із зображенням, назвою, коротким описом та іншою релевантною інформацією.

У разі виникнення помилки під час завантаження даних (наприклад, через проблеми з мережею або недоступність бази даних), інтерфейс повідомляє користувача про невдачу та пропонує повторити спробу. Якщо ж база даних не містить жодного аніме, система повертає порожній список, а інтерфейс інформує користувача про відсутність результатів. Такий підхід забезпечує зручність навігації, швидкий доступ до інформації та позитивний користувацький досвід.

2.2 Діаграма послідовності

Діаграма послідовності ілюструє процес оновлення профілю користувача на платформі AniMore, діаграма зображена на рисунку 2.2. Основними учасниками взаємодії є: Користувач, компонент ProfileInfo (відповідає за інтерфейс профілю), FirebaseSDK (логіка взаємодії з Firebase) та Firestore (база даних).

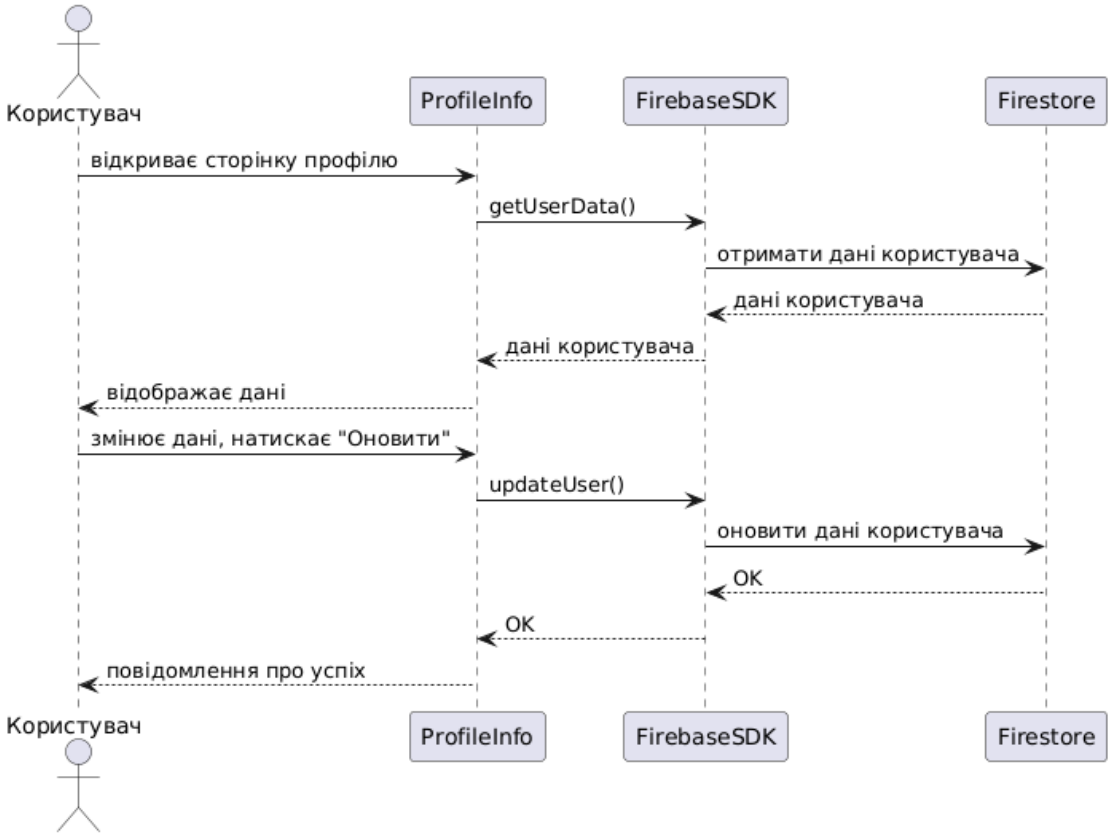


Рисунок 2.2 – Діаграма послідовності для прецеденту «Оновлення даних»

Користувач відкриває сторінку профілю. Компонент ProfileInfo ініціює запит до FirebaseSDK методом getUserData(), щоб отримати поточні дані користувача. FirebaseSDK надсилає запит до Firestore для отримання даних користувача. Firestore повертає відповідну інформацію, яка через FirebaseSDK надходить назад у компонент ProfileInfo. ProfileInfo відображає отримані дані користувачу у вигляді форми. Коли користувач змінює дані та натискає кнопку "Оновити", ProfileInfo викликає метод updateUser() у FirebaseSDK. FirebaseSDK надсилає запит на оновлення даних у Firestore. Після успішного оновлення Firestore повертає підтвердження (OK). FirebaseSDK передає підтвердження компоненту ProfileInfo, який повідомляє користувача про успішне оновлення профілю.

2.2.1 Діаграма класів

Діаграма класів відображає основну структуру та взаємозв’язки між ключовими компонентами клієнтської частини платформи AniMore, діаграма зображена на рисунку 2.3.

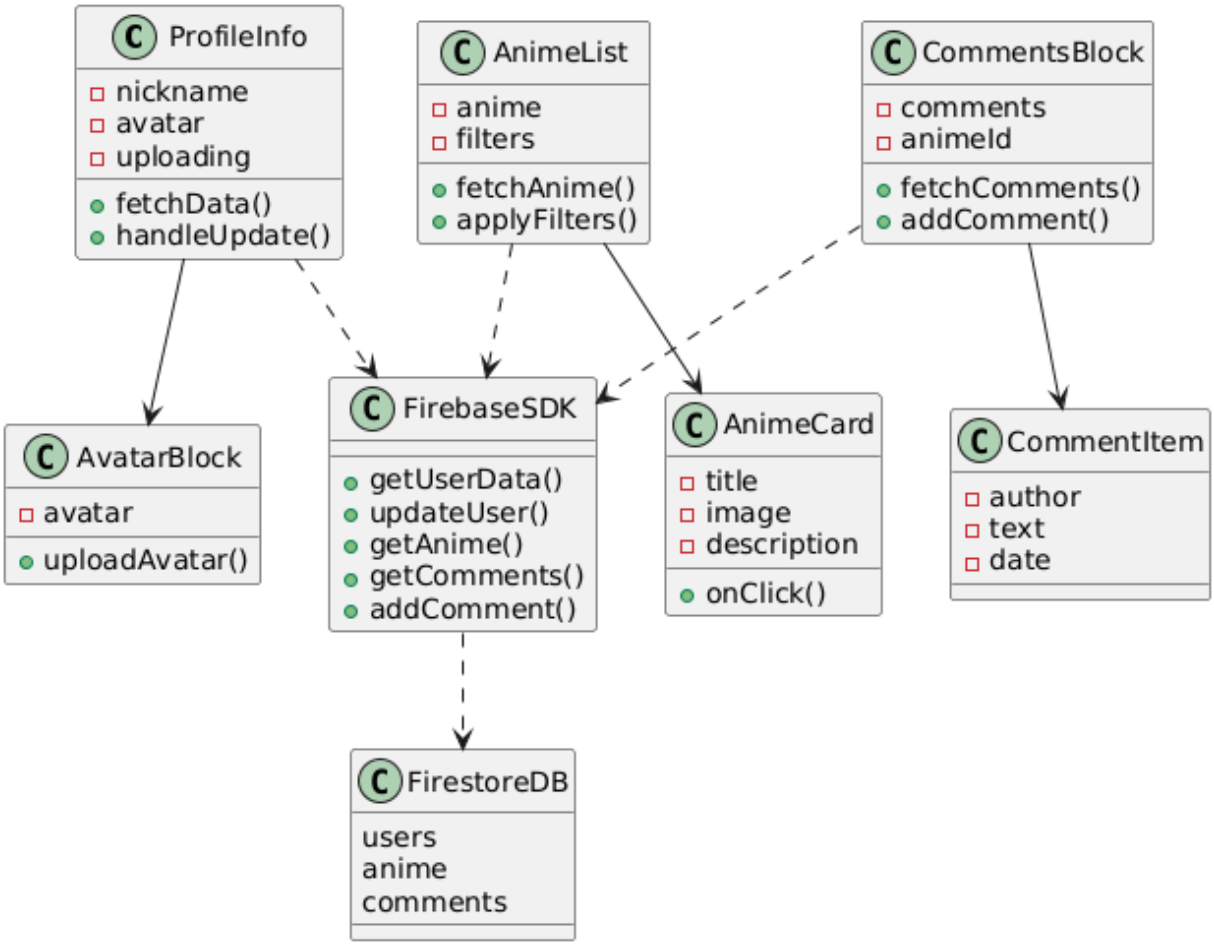


Рисунок 2.3 – Діаграма класів

У центрі діаграми знаходиться клас FirebaseSDK, який виступає посередником між клієнтськими компонентами та базою даних FirestoreDB. Саме через FirebaseSDK усі основні компоненти — такі як ProfileInfo, AnimeList та CommentsBlock - виконують запити для отримання або оновлення даних користувача, списків аніме чи коментарів.

Компонент profileInfo відповідає за відображення та редагування профілю користувача. Він містить властивості для зберігання нікнейму, аватара та стану завантаження, а також методи для отримання й оновлення даних. Для роботи з аватаром profileInfo використовує окремий компонент AvatarBlock, який реалізує логіку завантаження зображення.

Компонент animeList реалізує функціонал перегляду та фільтрації списку аніме. Він містить властивості для зберігання списку аніме та фільтрів, а також

методи для отримання даних і застосування фільтрів. Для відображення окремих аніме використовується компонент AnimeCard, який містить інформацію про конкретне аніме та метод для обробки кліків.

Компонент commentsBlock відповідає за відображення та додавання коментарів до аніме. Він містить властивості для зберігання списку коментарів та ідентифікатора аніме, а також методи для отримання та додавання коментарів. Для відображення окремого коментаря використовується компонент CommentItem, який містить автора, текст і дату коментаря.

Усі ці компоненти взаємодіють із FirebaseSDK, який надає методи для роботи з користувачами, аніме та коментарями. Сам FirebaseSDK звертається до FirestoreDB, де зберігаються відповідні колекції: users, anime, comments.

Діаграма демонструє модульну структуру застосунку, чітке розділення відповідальності між компонентами та централізовану роботу з даними через FirebaseSDK, що забезпечує зручність підтримки та масштабування системи.

2.2.2 Діаграма компонентів

Діаграма компонентів ілюструє основну структуру клієнтської частини застосунку AniMore. Вона показує, як різні компоненти взаємодіють між собою для забезпечення роботи основних функцій платформи. Діаграма зображена на рисунку 2.4.

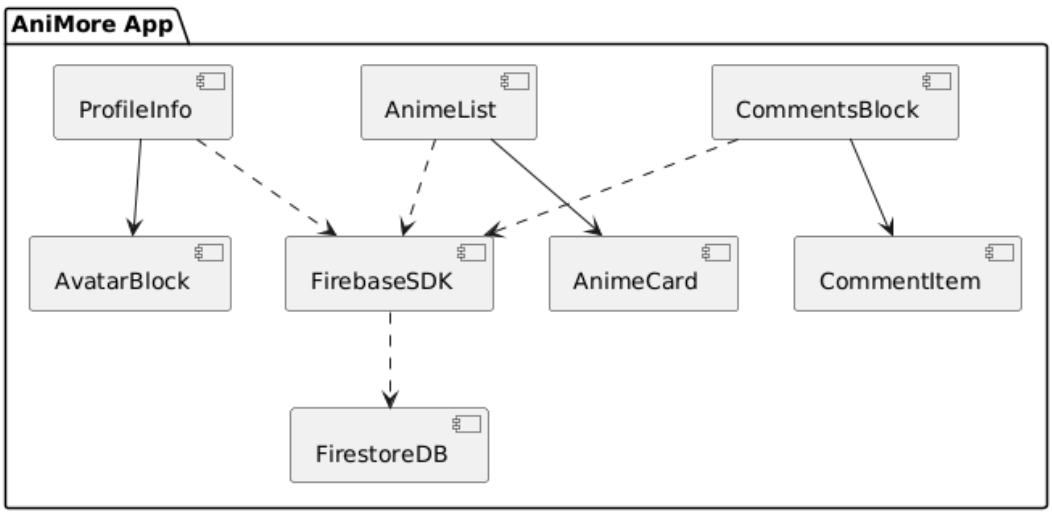


Рисунок 2.4 – Діаграма компонентів

Усі ключові сторінкові компоненти ProfileInfo, AnimeList та CommentsBlock використовують допоміжні підкомпоненти для реалізації окремих функцій. Зокрема, ProfileInfo взаємодіє з AvatarBlock для роботи з аватаром користувача, AnimeList використовує AnimeCard для відображення інформації про кожне аніме, а CommentsBlock — CommentItem для відображення окремих коментарів.

Центральним елементом взаємодії є FirebaseSDK, через який усі основні компоненти звертаються до бази даних FirestoreDB для отримання, оновлення або зберігання інформації про користувачів, аніме та коментарі. Така структура забезпечує чітке розділення відповідальності, модульність і зручність масштабування застосунку.

2.2.3 Діаграма діяльності

Діаграма діяльності відображає послідовність дій під час редагування профілю користувача на платформі AniMore, зображена на рисунку 2.5.

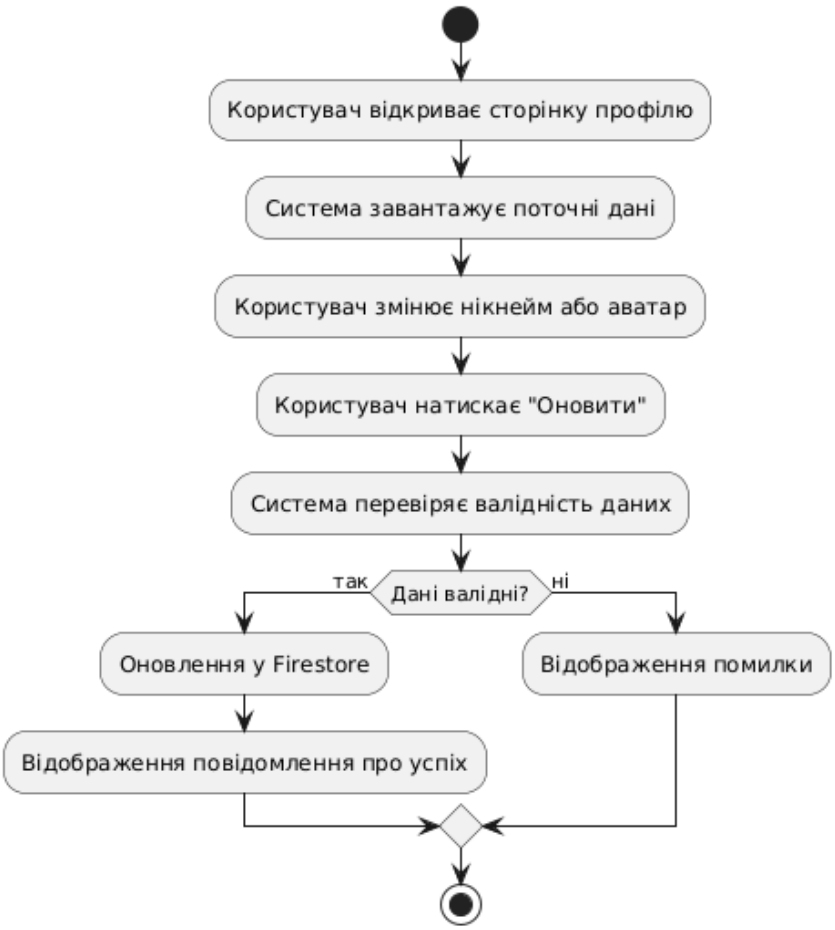


Рисунок 2.5 – Діаграма діяльності «Редагування профілю»

Процес починається з того, що користувач відкриває сторінку профілю. Система автоматично завантажує поточні дані користувача з бази даних. Далі користувач може змінити нікнейм або аватар, після чого натискає кнопку "Оновити". Система перевіряє валідність введених даних.

Якщо дані є валідними, відбувається оновлення інформації у Firestore, і користувачу відображається повідомлення про успішне збереження змін. Якщо ж дані не проходять перевірку, система повідомляє про помилку, і користувач може виправити введену інформацію. Така послідовність забезпечує зручність, надійність та безпеку процесу редагування профілю.

2.2.4 Діаграма розгортання

Діаграма розгортання ілюструє, як основні компоненти системи AniMore взаємодіють між собою на рівні інфраструктури, зображено на рисунку 2.6.

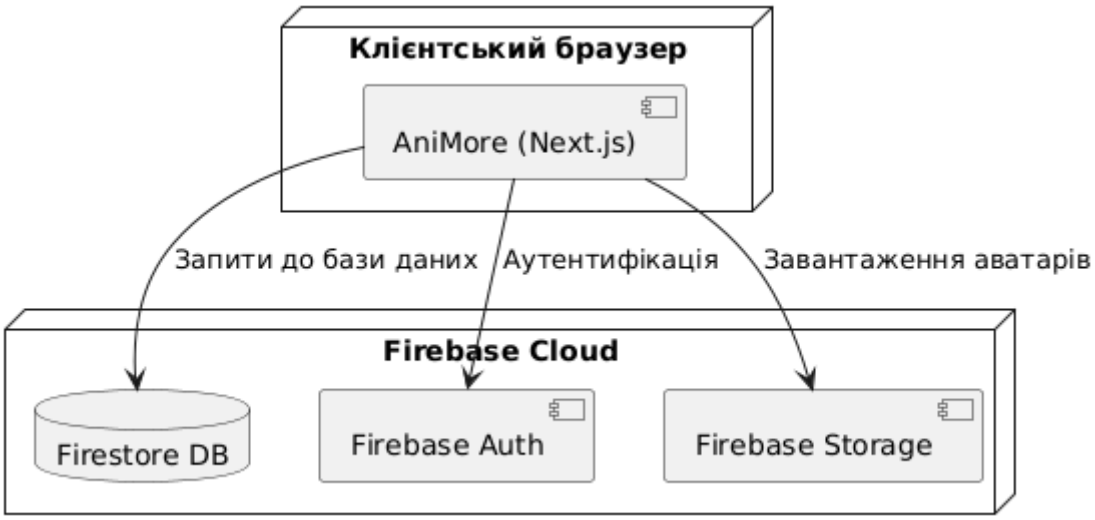


Рисунок 2.6 – Діаграма розгортання взаємодії вебсистеми

Клієнтський браузер користувача запускає застосунок AniMore, розроблений на Next.js. Вся логіка роботи з даними, автентифікацією та зберіганням зображень реалізована через хмарні сервіси Firebase. Зокрема, AniMore напряму звертається до Firestore DB для виконання запитів до бази даних, до Firebase Auth для проходження аутентифікації користувачів, а також до Firebase Storage для завантаження та отримання аватарів. Усі ці сервіси розміщені

у хмарі Firebase, що забезпечує масштабованість, високу доступність і безпеку даних. Така архітектура дозволяє уникнути власного серверного бекенду, спрощує розгортання та обслуговування проєкту.

2.2.5 Діаграма пакетів

Діаграма пакетів відображає організацію структури клієнтської частини проєкту AniMore та взаємозв’язки між основними модулями, зображено на рисунку 2.7.

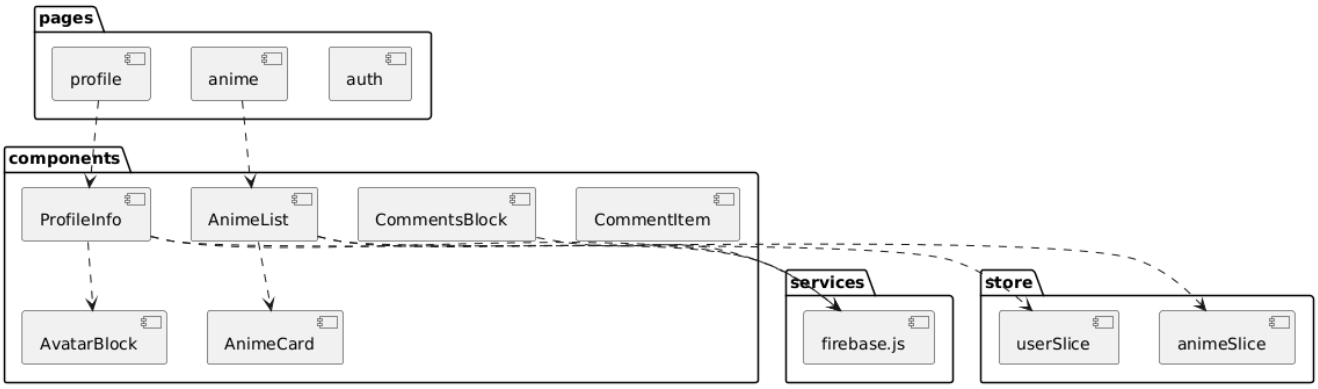


Рисунок 2.7 – Діаграма пакетів взаємозв’язків між компонентами

У верхній частині діаграми знаходиться пакет pages, який містить сторінки застосунку: profile, anime та auth. Кожна сторінка використовує відповідні компоненти з пакету components. Наприклад, сторінка профілю підключає компонент ProfileInfo, сторінка аніме — AnimeList, а сторінка авторизації — відповідний компонент для входу.

Пакет components містить універсальні компоненти, які реалізують окремі частини інтерфейсу: ProfileInfo, AnimeList, CommentsBlock, CommentItem, AvatarBlock, AnimeCard тощо. Деякі з них, наприклад, ProfileInfo та AnimeList, використовують вкладені підкомпоненти (AvatarBlock, AnimeCard).

Для роботи з даними компоненти звертаються до пакету services, де розміщено файл firebase.js із логікою взаємодії з Firebase (авторизація, робота з базою даних, зображеннями тощо).

Управління глобальним станом застосунку здійснюється через пакет store, який містить Redux-слайси: userSlice для даних користувача та animeSlice для

списку аніме. Компоненти, що потребують доступу до глобального стану, підключають відповідні слайси.

Діаграма демонструє модульну, зрозумілу та масштабовану структуру проєкту, де кожен пакет відповідає за свою частину функціоналу, а залежності між ними чітко визначені. Такий підхід спрощує підтримку, розширення та тестування застосунку.

2.3 Алгоритм роботи вебсистеми

У процесі моделювання програмного забезпечення вебплатформи AniMore було розроблено алгоритм роботи системи, який відображає основні сценарії взаємодії користувача з платформою та внутрішню логіку обробки даних. Моделювання дозволяє виявити ключові етапи функціонування, оптимізувати структуру застосунку та забезпечити його масштабованість і зручність для кінцевого користувача.

Робота вебсистеми починається з ініціалізації застосунку та підключення до хмарної бази даних Firestore. Після відкриття головної сторінки користувач отримує доступ до динамічно сформованого списку аніме, жанрів і рекомендацій. Якщо користувач вже був авторизований раніше, система автоматично відновлює його сесію та підтягує персональні дані.

Далі користувач може здійснювати пошук або фільтрацію аніме за різними параметрами, переглядати детальну інформацію про тайтли, епізоди та коментарі. Усі ці дії виконуються асинхронно, що забезпечує швидке оновлення інтерфейсу без перезавантаження сторінки.

Для виконання персоналізованих дій (додавання улюбленого, залишення коментаря, редагування профілю) система перевіряє статус авторизації користувача. Якщо користувач не авторизований, його перенаправляють на

сторінку входу. Після успішної автентифікації він повертається до виконання обраної дії.

Всі зміни, які вносить користувач, одразу зберігаються у Firestore, а інтерфейс оновлюється в реальному часі завдяки підписці на зміни. Це дозволяє забезпечити інтерактивність і актуальність даних для всіх користувачів платформи.

Моделювання цього алгоритму дозволяє чітко визначити взаємозв’язки між компонентами системи, сценарії використання та точки інтеграції з хмарними сервісами, .модель алгоритму зображено на рисунку 2.8.

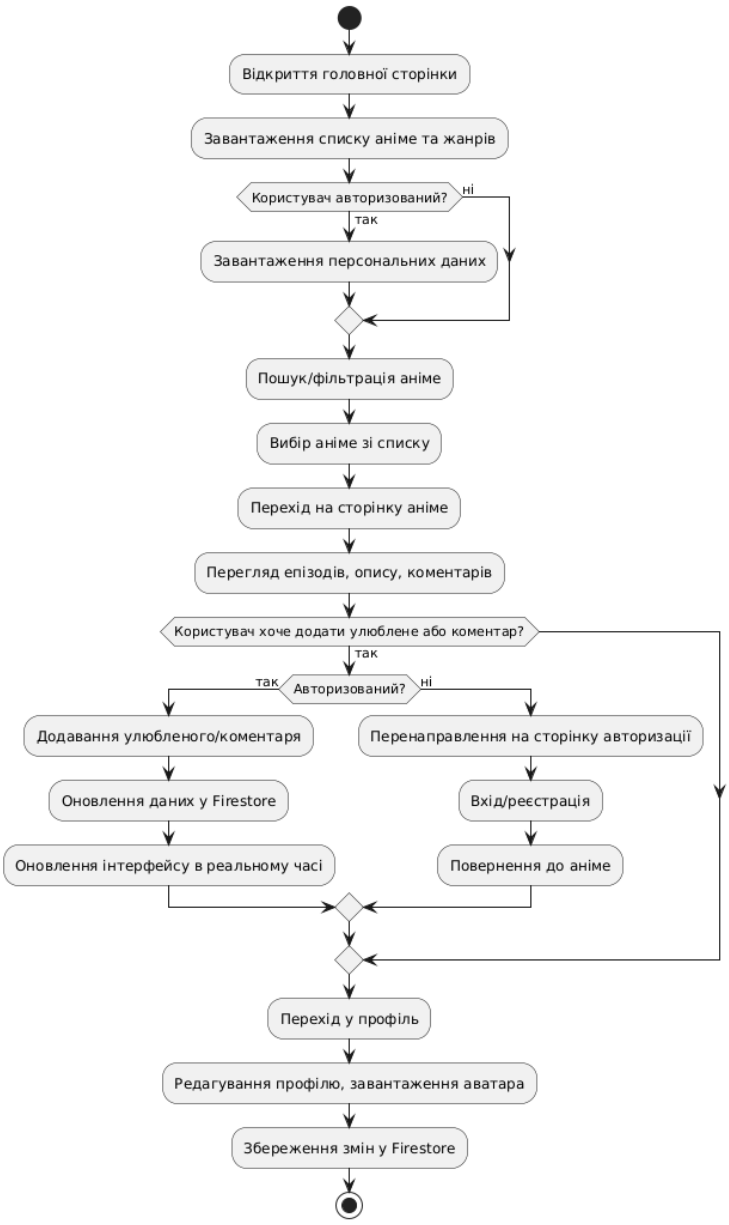


Рисунок 2.8 – Алгоритм роботи вебсистеми

2.4 Опис політики безпеки

Для забезпечення захисту даних користувачів та цілісності інформації у вебплатформі AniMore використовується система політик безпеки, що реалізується за допомогою правил доступу Firestore Security Rules, зображено у рисунку 2.9. Ці правила визначають, хто і які дії може виконувати з даними у базі, а також обмежують несанкціонований доступ до персональної інформації.

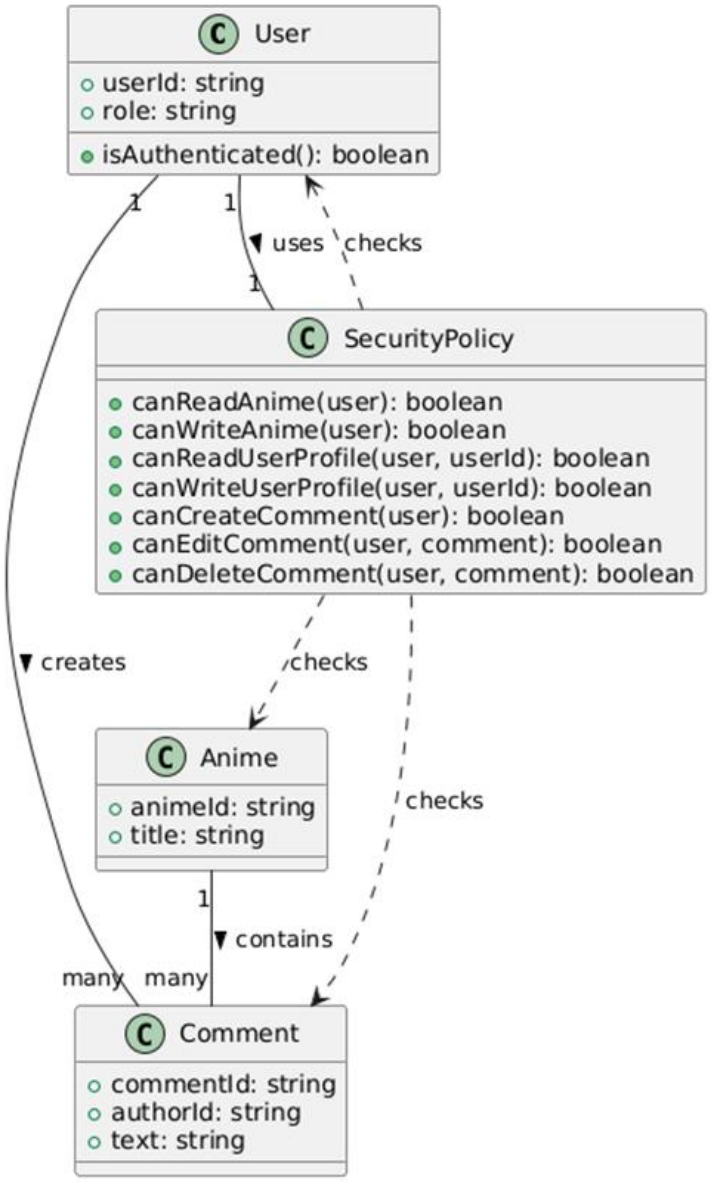


Рисунок 2.9 - Firestore Security Rules AniMore

Основні принципи політик безпеки у проєкті:

- доступ до загальнодоступних даних (наприклад, список аніме, жанри, релізи) дозволений для всіх користувачів, навіть неавторизованих. Це забезпечує відкритість платформи для нових відвідувачів;

- доступ до персональних даних (наприклад, профіль користувача, улюблене, підписки) дозволений лише для автентифікованого користувача, причому кожен користувач може читати й змінювати лише власні дані;

- додавання коментарів дозволено лише авторизованим користувачам. Видаляти або редагувати коментар може лише його автор або адміністратор;

- адміністративні дії (наприклад, видалення коментарів, модерація контенту) доступні лише користувачам із відповідною роллю (адміністратор).

3 ПРОЄКТУВАННЯ ВЕБСИСТЕМИ

3.1 Архітектура вебсервісу

Архітектура вебсервісу AniMore розроблена з урахуванням сучасних вимог до продуктивності, масштабованості, безпеки та зручності супроводу. Вона базується на поєднанні клієнтської частини, реалізованої на фреймворку Next.js, та хмарних сервісів Firebase, що дозволяє ефективно розподіляти навантаження, забезпечувати швидкий доступ до даних і гнучко реагувати на зміни у вимогах до функціоналу.

Клієнтська частина системи реалізована на основі Next.js — одного з найпопулярніших фреймворків для React. Next.js забезпечує серверний рендеринг сторінок (SSR), що позитивно впливає на SEO-оптимізацію та швидкість початкового завантаження застосунку. Завдяки вбудованій маршрутизації та підтримці динамічних сторінок, розробка інтерфейсу є зручною та структурованою.

Всі основні функціональні блоки (сторінка профілю, список аніме, перегляд і додавання коментарів, сторінка авторизації тощо) реалізовані у вигляді окремих компонентів. Це дозволяє легко розширювати функціонал, повторно використовувати код та підтримувати чисту архітектуру проєкту. Для управління глобальним станом застосунку використовується Redux, що забезпечує синхронізацію даних між різними компонентами та дозволяє уникнути дублювання логіки.

Інтерфейс користувача розроблений із урахуванням принципів адаптивного дизайну, що гарантує коректне відображення на різних пристроях — від настільних комп'ютерів до смартфонів. Для стилізації використовуються CSS-модулі, що дозволяє уникати конфліктів стилів та підтримувати єдиний стиль усього застосунку.

3.1.1 Хмарна інфраструктура

Для реалізації серверної логіки, зберігання даних та автентифікації користувачів використовується хмарна платформа Firebase, яка об'єднує кілька ключових сервісів.

Firestore DB - це сучасна NoSQL база даних, що забезпечує зберігання інформації про користувачів, аніме, коментарі, підписки та інші сутності. Вона підтримує масштабування, транзакції, оновлення даних у реальному часі та гнучкі правила безпеки.

Для реєстрації, авторизації та управління сесіями користувачів застосовується сервіс firebase Auth, який підтримує різні способи автентифікації (email, Google, Facebook тощо) і гарантує захист персональних даних.

Зберігання медіафайлів, зокрема аватарів користувачів, реалізовано за допомогою firebase Storage - це хмарне сховище, яке дозволяє зручно завантажувати, зберігати та отримувати зображення з будь-якого пристрою.Взаємодія між клієнтською частиною та хмарними сервісами здійснюється через Firebase SDK. Це дозволяє напряму виконувати всі необхідні операції - читання, запис, оновлення та видалення даних, а також автентифікацію та роботу з файлами - без необхідності розгортання окремого серверного

застосунку. Такий підхід значно спрощує архітектуру, зменшує затримки при обробці запитів і підвищує загальну продуктивність системи. Порівняльна таблиця, з порівнянням Firebase з іншими популярними рішеннями зображена в таблиці 3.1.

Таблиця 3.1 Порівняння Firebase з іншими СКБД

Критерій	Firebase (Firestore, Auth, Storage)	AWS (DynamoDB, Cognito, S3)	Google Cloud (Datastore, Identity Platform, Cloud Storage)	Власний сервер (Node.js + MongoDB)
Масштабованість	Автоматична	Автоматична	Автоматична	Потрібно налаштовувати вручну
Доступність	Висока (SLA від Google)	Висока (SLA від Amazon)	Висока (SLA від Google)	Залежить від хостингу
Безпека	Вбудована, прості правила доступу	Гнучкі політики IAM	Гнучкі політики IAM	Потрібно реалізовувати самостійно
Інтеграція сервісів	Висока, все в одному SDK	Висока, але різні SDK	Висока, але різні SDK	Потрібно інтегрувати вручну
Реальний час	Підтримується	Потрібно додатково налаштовувати	Потрібно додатково налаштовувати	Потрібно реалізовувати окремо
Простота розгортання	Дуже проста	Середня	Середня	Складна, потрібен DevOps
Вартість	Безкоштовний тариф, Pay-as-you-go	Безкоштовний тариф, Pay-as-you-go	Безкоштовний тариф, Pay-as-you-go	Витрати на сервер, адміністрування
Документація	Дуже детальна, багато прикладів	Дуже детальна	Дуже детальна	Залежить від вибраних технологій
Підтримка	Від Google, велика спільнота	Від Amazon, велика спільнота	Від Google, велика спільнота	Залежить від спільноти

Firebase забезпечує просту інтеграцію, автоматичне масштабування, вбудовану безпеку та швидкий старт для сучасних вебпроєктів. AWS та Google Cloud пропонують більше гнучкості для великих корпоративних рішень, але вимагають більше налаштувань. Власний сервер дає максимальний контроль, але потребує значних ресурсів на підтримку та безпеку.

3.1.2 Cloudinary

Для зберігання та обробки зображень у проєкті AniMore, окрім стандартного сховища Firebase Storage, було інтегровано хмарний сервіс Cloudinary, рисунок 3.1. Cloudinary є спеціалізованим рішенням для завантаження, зберігання, оптимізації та трансформації медіафайлів, що дозволяє значно спростити роботу з графічним контентом та підвищити швидкодію вебсистеми.

Основні переваги Cloudinary:

- автоматична оптимізація зображень для різних пристроїв і швидке завантаження;
- можливість масштабування, кадрування, зміни формату та інших трансформацій «на льоту»;
- зручний API для інтеграції у фронтенд- і бекенд-частини застосунку;
- безпечне зберігання та керування правами доступу до файлів.

У процесі розробки функціоналу редагування профілю користувача (зміна аватара) було вирішено використовувати Cloudinary для завантаження зображень. Коли користувач вибирає новий аватар, файл завантажується на сервер Cloudinary через API, після чого у Firestore зберігається лише посилання на оптимізоване зображення. Це дозволяє:

- зменшити навантаження на Firebase Storage,
- забезпечити швидке відображення аватарів навіть при повільному інтернет-з’єднанні,
- використовувати автоматичну оптимізацію та трансформацію зображень під різні розміри і формати, що підвищує якість відображення на різних пристроях.

Інтеграція Cloudinary здійснювалася за допомогою офіційного SDK та REST API. Для безпечного завантаження використовувалися підписані запити, а також налаштовано обмеження на розмір і тип файлів.

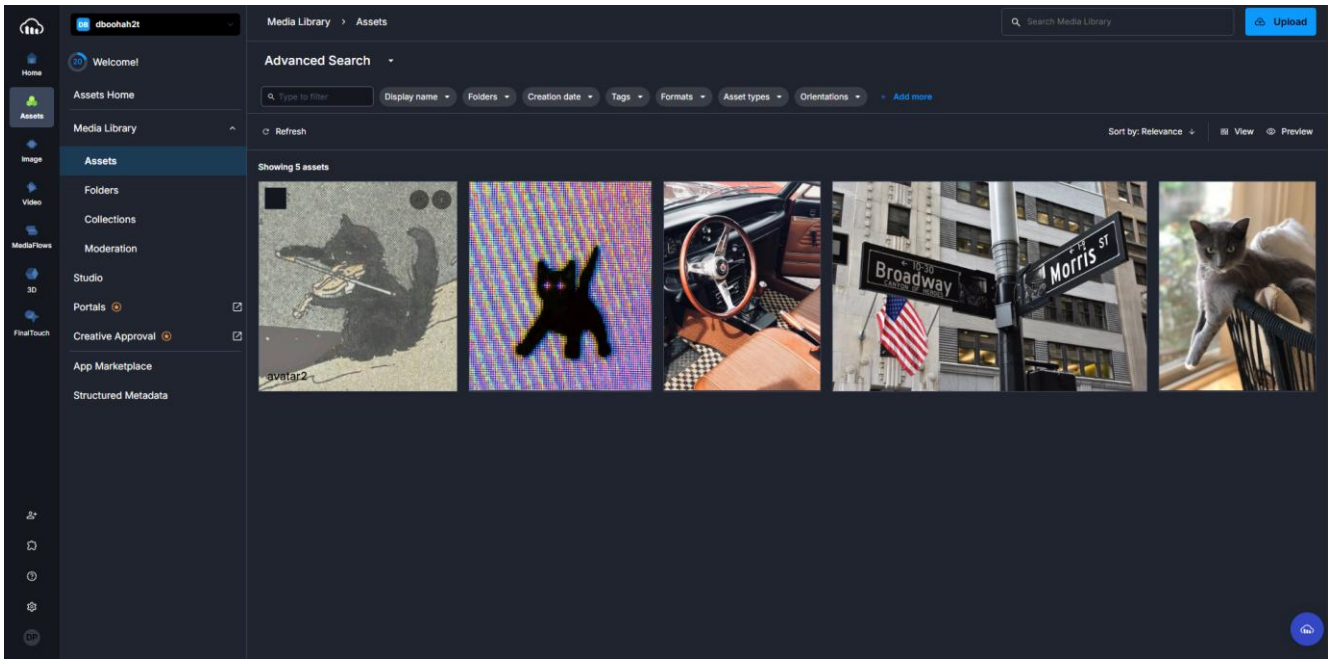


Рисунок 3.1 – Кабінет проекту в Cloudinary

3.1.3 Роль Firestore у проєкті

Для зберігання, організації та обробки даних у вебсистемі AniMore використовується хмарна база даних Cloud Firestore від платформи Firebase. Firestore є сучасною NoSQL базою даних, яка забезпечує масштабованість, високу доступність, роботу в реальному часі та гнучкі можливості для організації даних.

Основні переваги Firestore:

- гнучка структура даних;
- масштабованість;
- реальний час;
- гнучкі правила безпеки;
- інтеграція з іншими сервісами Firebase.

Firestore організовує дані у вигляді колекцій та документів, що дозволяє зберігати різнорідну інформацію (наприклад, користувачі, аніме, коментарі, підписки) у зручному та структурованому форматі.

База даних автоматично масштабується під навантаження, тому підходить як для невеликих проєктів, так і для великих спільнот. Завдяки підтримці підписки на зміни даних, інтерфейс користувача може миттєво оновлюватися без необхідності перезавантаження сторінки. Доступ до даних контролюється за

допомогою спеціальних правил безпеки, які можна налаштовувати відповідно до ролей користувачів та типу операцій. Крім того, Firestore легко інтегрується з іншими сервісами Firebase, такими як Auth та Storage, що забезпечує комплексну роботу застосунку.

У проєкті AniMore Firestore використовується для зберігання основних сутностей платформи:

- користувачі;
- аніме;
- коментарі;
- підписки.

У Firestore для кожної з цих сутностей зберігається відповідна інформація. Для користувачів фіксуються дані профілю, улюблені аніме, підписки, аватар (у вигляді посилання на Cloudinary або Firebase Storage) та роль (звичайний користувач чи адміністратор).

Аніме представлено у вигляді детального переліку з назвою, описом, жанром, роком випуску, зображенням, рейтингом та іншими характеристиками.

Для кожного аніме окремо зберігаються коментарі користувачів із зазначенням автора, тексту, дати та часу створення, що дозволяє формувати хронологічний список обговорень.

Підписки дають змогу відстежувати, на які аніме або події підписаний користувач, і формувати персоналізовані списки та сповіщення про нові серії чи оновлення улюблених тайтлів.

Взаємодія з Firestore здійснюється через Firebase SDK. Усі основні операції (читання, запис, оновлення, видалення) виконуються безпосередньо з клієнтської частини застосунку. Для оптимізації роботи використовуються запити з фільтрацією, сортуванням та пагінацією. Структура даних в Firestore зображено в таблиці 3.2.

Таблиця 3.2 – Структура даних Firestore

Колекція	Документ (ID)	Поля документа	Підколекції	Опис
users	userId	nickname, email, avatarUrl, favorites (масив animeId), role (user/admin), createdAt	—	Дані профілю користувача
anime	animeId	title, description, genre, year, imageUrl, rating, createdAt	comments	Інформація про аніме
anime/comments	commentId	userId, text, createdAt, updatedAt	—	Коментарі до конкретного аніме
subscriptions	userId	animeIds (масив), createdAt	—	Підписки користувача на аніме

Пояснення до структури даних:

- users - колекція користувачів, кожен документ містить дані профілю;
- anime - колекція аніме, кожен документ містить інформацію про одне аніме;
- anime/comments - підколекція коментарів для кожного аніме;
- subscriptions - колекція підписок, де зберігаються ідентифікатори аніме, на які підписаний користувач.

Для захисту даних у Firestore налаштовано правила безпеки, які дозволяють:

- читати загальнодоступну інформацію (наприклад, список аніме) всім користувачам,
- змінювати персональні дані лише власнику профілю,
- додавати коментарі лише авторизованим користувачам,
- виконувати адміністративні дії (модерація, керування користувачами) лише адміністраторам.

3.2 Проєктування та розробка інтерфейсу користувача

Інтерфейс користувача (UI) платформи AniMore розроблявся з урахуванням потреб кінцевих користувачів, щоб забезпечити просту, приємну та ефективну взаємодію з системою. Основна мета — надати інтуїтивний доступ до всіх ключових функцій: перегляд аніме, додавання до улюбленого, робота з профілем і коментарями, при цьому підтримуючи сучасний і легкий візуальний стиль. Завдяки використанню Next.js та React, інтерфейс є модульним і компонентно-орієнтованим, що дає змогу легко оновлювати окремі елементи без впливу на загальну структуру. У цьому розділі детально описано основні екрани застосунку, їхню функціональність, візуальні особливості та підхід до забезпечення зручності для користувачів.

Основні принципи розробки інтерфейсу:

- інтуїтивність;
- лаконічність;
- адаптивність;
- швидкодія;
- зворотний зв’язок.

Інтуїтивність передбачає, що навіть новий користувач миттєво розуміє, як працювати з інтерфейсом, — усі кнопки та елементи управління розташовані логічно і відповідають очікуванням.

Лаконічність досягається мінімалістичним підходом: використовується обмежена палітра кольорів, чіткі шрифти та лише найнеобхідніші елементи, щоб уникнути зайвого навантаження на користувача.

Адаптивність забезпечує коректне відображення інтерфейсу на будь-яких пристроях — від великих екранів десктопів до смартфонів, адже компоненти автоматично підлаштовуються під розміри екрану.

Швидкодія гарантує, що ключові дії (наприклад, перегляд аніме, додавання до улюбленого або редагування профілю) виконуються максимально швидко та в кілька кліків.

Зворотний зв’язок полягає в тому, що кожна дія користувача супроводжується зрозумілим повідомленням: після додавання до обраного або публікації коментаря відразу ж з’являється сповіщення про результат операції.

3.2.1 Головна сторінка

Головна сторінка є центральним елементом платформи AniMore та першою точкою взаємодії користувача із системою. Вона містить:

- верхню навігаційну панель;
- панель пошуку та фільтрації;
- сітку карток аніме.

Верхня навігаційна панель включає логотип AniMore, основні розділи (Головна, Улюблене, Профіль), кнопку авторизації/виходу та перемикач теми (світла/темна).

Панель пошуку та фільтрації розташована під навігацією. Містить поле пошуку за назвою аніме, випадаючі списки для фільтрації за жанром, роком випуску, рейтингом, а також кнопку для скидання фільтрів. Це дозволяє користувачу швидко знаходити потрібний контент.

Сітка карток аніме є основною частиною сторінки — кожна картка відображає постер, назву, короткий опис і кнопку “Додати до улюбленого”.

Кожна картка містить:

- зображення(обкладинку аніме);
- назву аніме;
- кнопку «Детальніше» для переходу на сторінку аніме;
- кнопку для додавання або видалення з обраного.

Картки мають сучасний вигляд із акцентом на зображення, а основні дії доступні безпосередньо з картки. Також на головній сторінці є:

- пагінація: якщо аніме багато, над сіткою розташовані кнопки переходу між сторінками;

- футер: містить контактну інформацію, посилання на соцмережі, копірайт.

Головна сторінка виконана в темних тонах із використанням насичених акцентів для кнопок та підписів, а також можливість перемикання на світлу тему. Всі елементи мають достатній контраст, що забезпечує доступність для користувачів із порушеннями зору. Дизайн адаптивний: на мобільних пристроях сітка карток змінюється на вертикальний список.

3.2.2 Сторінка релізу

Сторінка детального перегляду аніме на платформі AniMore надає користувачу повну інформацію про обране аніме та дозволяє взаємодіяти з контентом у кілька кліків. Основні елементи сторінки:

- верхній інформаційний блок;
- опис та основна інформація;
- відеоплеєр;
- список епізодів/глав;
- блок коментарів;
- адаптивність та доступність.

У верхній частині сторінки розташовано велике зображення (банер або постер), яке створює яскравий візуальний акцент і задає атмосферу для перегляду. Поруч із зображенням великим шрифтом відображається назва аніме, а також основні характеристики — жанр, рік випуску та рейтинг, які подані у вигляді тегів або піктограм для швидкого сприйняття. Біля назви або під основною інформацією розміщена кнопка "Додати до улюбленого" (або "Видалити з улюбленого", якщо аніме вже додано), яка має яскравий вигляд і легко помітна для користувача.

Далі йде розширений опис, що містить детальний сюжет, особливості, інформацію про авторів, студію, тривалість, вікові обмеження та інші важливі деталі. Вся текстова інформація структурована за допомогою підзаголовків, списків і відступів, що робить її легкою для читання та сприйняття.

Особливе місце на сторінці займає вбудований відеоплеєр, який дозволяє переглядати епізоди аніме безпосередньо на сторінці. Плеєр підтримує основні функції: відтворення та паузу, перемотування, вибір якості, повноекранний режим і регулювання гучності. Якщо для аніме доступно кілька епізодів, під плеєром розташований список серій, що дозволяє швидко перемикатися між ними. Клік по епізоду миттєво перемикає відеоплеєр на відповідну серію. Додатково може відображатися інформація про дату виходу, тривалість та статус перегляду кожної серії.

Нижче розміщено блок коментарів, де користувачі можуть переглядати відгуки інших, залишати власні коментарі або видаляти свої записи. Для адміністратора доступна функція модерації коментарів. Кожен коментар містить аватар, ім’я користувача, текст і дату публікації. Додавання чи видалення коментаря супроводжується toast-сповіщеннями, які інформують про успішність дії або помилку, приклад зображено у лістингу 3.1.

Весь дизайн сторінки є адаптивним: на мобільних пристроях блоки розташовуються вертикально, відеоплеєр і список епізодів займають всю ширину екрану, а навігація по серіях реалізована свайпом або випадаючим списком. Всі елементи мають достатній контраст, підписи до кнопок і підтримують клавіатурну навігацію, що робить інтерфейс доступним для всіх користувачів.

Головний акцент у візуальному оформленні зроблено на великому зображенні та назві аніме. Всі блоки чітко відділені відступами, використовуються заголовки для логічного поділу інформації, а кнопки дій мають яскравий колір і добре помітні. Функціонал відеоплеєра реалізовано через окремий блок або вкладку на сторінці аніме. Якщо для тайтлу є манга, користувач може перемкнутися на вкладку з мангою (MangaChaptersReader); якщо манга відсутня — відображається відповідне повідомлення.

Лістинг 3.1 – Фрагмент коду для блоку коментарів

```
<form className={styles.commentForm} onSubmit={handleCommentSubmit}>
  <textarea
    className={styles.commentInput}
```

```
    placeholder="Напишите комментарий..."
    value={comment}
    onChange={e => setComment(e.target.value)}
    rows={3}
  />
  <button                                                    type="submit"
className={styles.commentBtn}>Отправить</button>
</form>
<div className={styles.commentsList}>
  {comments.length === 0 ? (
    <div   className={styles.noComments}>Тут   будет   ваш   первый
комментарий</div>
  ) : (
    comments.map((c, idx) => (
      <div key={idx} className={styles.commentItem}>
        <div className={styles.commentAuthor}>{c.author}</div>
        <div className={styles.commentText}>{c.text}</div>
        <div className={styles.commentDate}>
          {c.date.toLocaleString('ru-RU',  {  day:  '2-digit',  month:
'2-digit', year: '2-digit', hour: '2-digit', minute: '2-digit' })}
        </div>
      </div>
    ))
  )}
</div>
```

3.2.3 Сторінка профілю користувача

Сторінка профілю користувача на платформі AniMore є персональним простором, де кожен користувач може переглядати та редагувати свої особисті дані, а також керувати улюбленими аніме. У верхній частині сторінки розташований аватар користувача у круглій рамці, поряд відображається нікнейм, email та, за бажанням, короткий опис профілю. Всі ці дані структуровані для легкого сприйняття, а візуальний стиль сторінки витриманий у мінімалістичному ключі з акцентами на основних діях.

Особливу увагу приділено можливості редагування профілю. Кнопка "Редагувати профіль" відкриває форму, де користувач може змінити нікнейм, email, опис, а також завантажити новий аватар. Завантаження аватара реалізовано через інтеграцію з Cloudinary: після вибору зображення користувач бачить попередній перегляд, що дозволяє переконатися у правильності вибору перед збереженням змін, фрагмент реалізації зображено у лістингу 3.2.

Нижче на сторінці розташований список улюблених аніме, які користувач додав до свого профілю. Вони відображаються у вигляді карток із зображенням, назвою та кнопкою для швидкого переходу до детального перегляду. Це дозволяє користувачу швидко знаходити та переглядати свої улюблені тайтли.

Весь інтерфейс профілю адаптивний: на мобільних пристроях елементи розташовуються вертикально, а кнопки та поля вводу мають достатній розмір для зручної взаємодії. Дизайн враховує доступність — усі елементи мають підписи, контрастні кольори, а також підтримують клавіатурну навігацію.

Лістинг 3.2 – Фрагмент коду для форми редагування профілю

```
import { useState } from "react";
function ProfileEditForm({ user, onSave }) {
  const [nickname, setNickname] = useState(user.nickname);
  const [email, setEmail] = useState(user.email);
  const [avatar, setAvatar] = useState(user.avatarUrl);
  const [avatarPreview, setAvatarPreview] =
useState(user.avatarUrl);
  const handleAvatarChange = (e) => {
    const file = e.target.files[0];
    if (file) {
      setAvatar(file);
      setAvatarPreview(URL.createObjectURL(file)); } };
  const handleSubmit = (e) => {
    e.preventDefault();
    onSave({ nickname, email, avatar });
  };
  return (
    <form onSubmit={handleSubmit}>
      <label>
        Аватар:
        <input type="file" accept="image/*"
onChange={handleAvatarChange} />
      </label>
      {avatarPreview && (
        <img
          src={avatarPreview}
          alt="Попередній перегляд аватара"
          style={{ width: 100, height: 100, borderRadius: "50%",
margin: 8 }} /> )}
      <label>
        Нікнейм:
        <input
          type="text"
          value={nickname}
          onChange={(e) => setNickname(e.target.value)}
```

```
        />
      </label>
      <label>
        Email:
        <input
          type="email"
          value={email}
          onChange={(e) => setEmail(e.target.value)}  />
      </label>
      <button type="submit">Зберегти зміни</button>
    </form>
  ); }
export default ProfileEditForm;
```

Такий підхід до організації сторінки профілю забезпечує користувачу простий і зручний спосіб керування своїми даними та улюбленим контентом, а також створює сучасний і привабливий користувацький досвід.

4 РЕАЛІЗАЦІЯ ВЕБСИСТЕМИ ANIMORE

У цьому розділі детально описано підходи до розробки, структуру коду, основні компоненти та інтеграцію із зовнішніми сервісами. Для кожного ключового функціоналу наведено пояснення та приклади коду з посиланнями на відповідні фрагменти.

4.1 Структура проєкту

AniMore — це сучасний вебзастосунок, побудований на основі Next.js та React, дивитись рисунок 4.1. Такий підхід дозволяє поєднувати серверний і клієнтський рендеринг, забезпечуючи швидке завантаження сторінок та SEO-оптимізацію. Структура проєкту організована за принципом розділення відповідальностей, зображено на рисунку 4.2.

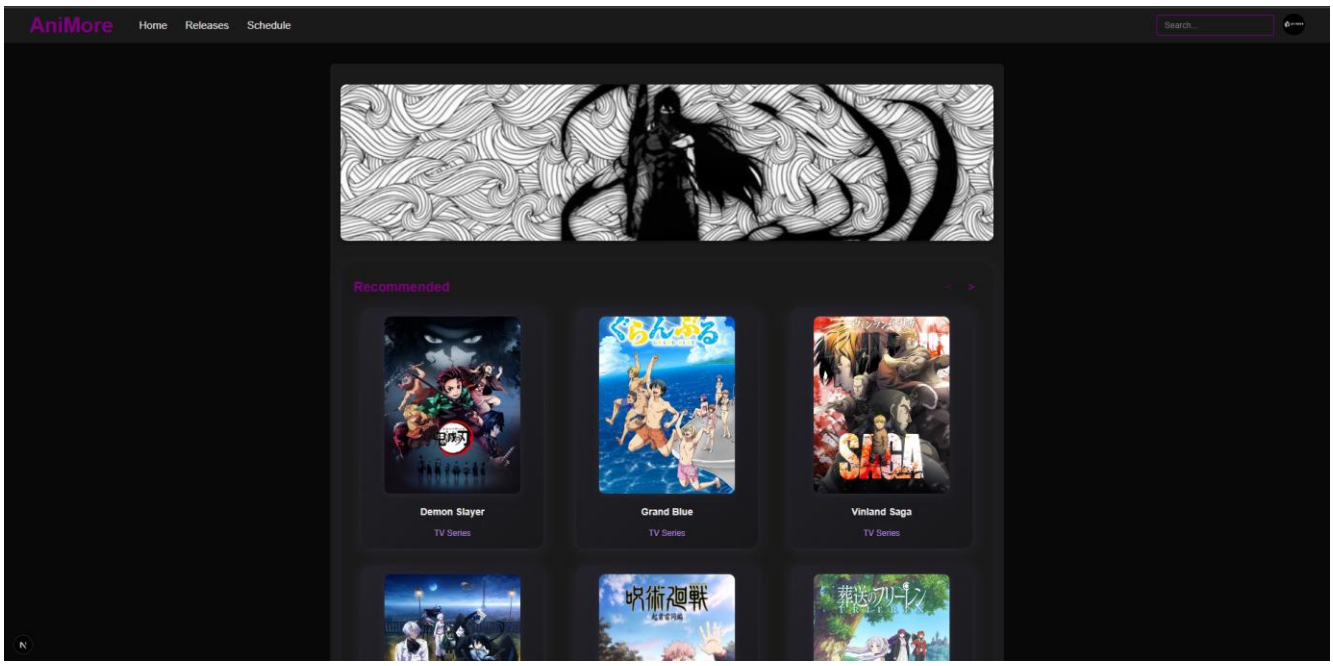


Рисунок 4.1 – Головна сторінка AniMore

```
anime-website/
├── components/          // Універсальні та спеціалізовані React-компоненти
├── pages/               // Сторінки Next.js (роутинг)
├── styles/              // CSS-модулі та глобальні стилі
├── utils/               // Допоміжні функції та API-запити
├── firebase/            // Конфігурація та ініціалізація Firebase
└── public/              // Статичні файли (іконки, зображення)
```

Рисунок 4.2 – Структура вебсистеми

Опис основних директорій:

- components/ — містить усі багаторазові React-компоненти, такі як AnimeCard, ProfileEditForm, AnimePlayer, CommentsBlock тощо. Кожен компонент відповідає за окрему частину інтерфейсу та має власні стилі;
- pages/ — реалізує маршрутизацію згідно з концепцією Next.js: кожен файл у цій директорії автоматично стає окремою сторінкою (наприклад, /pages/index.jsx — головна сторінка, [id].jsx — детальний перегляд аніме);
- styles/ — містить CSS-модулі для компонентів та глобальні стилі, що забезпечують єдиний візуальний стиль застосунку;

- utils/ — включає утиліти для роботи з Firestore, Cloudinary, а також допоміжні функції для обробки даних, валідації форм, форматування дат тощо;
- firebase/ — містить файли конфігурації та ініціалізації Firebase, а також окремі модулі для роботи з Firestore, Auth і Storage;
- public/ — призначена для зберігання статичних ресурсів, таких як зображення логотипу, іконки, фавікон, які використовуються у застосунку;
- hooks/ — містить кастомні React-хуки, наприклад, для отримання даних з Firestore у реальному часі або для керування станом форм.

Переваги такої структури:

- модульність: Кожен функціональний блок ізольований, що спрощує розробку, тестування та підтримку;
- масштабованість: Додавання нового функціоналу не впливає на існуючі компоненти;
- зручність навігації: Чітка структура директорій дозволяє швидко знаходити потрібні файли та логіку;
- легкість інтеграції: Завдяки використанню Next.js та Firebase легко інтегрувати нові сервіси або змінювати існуючі.

4.2 Реалізація автентифікації користувача

Вебплатформа AniMore забезпечує авторизацію користувачів за допомогою хмарного сервісу Firebase Authentication, зображено на рисунку 4.3. Це сучасне готове рішення, яке гарантує безпеку даних, масштабованість і простоту інтеграції у клієнтські застосунки. Система автентифікації дозволяє користувачам реєструватися, входити в систему та виходити з неї.

Основний функціонал автентифікації:

- реєстрація нового користувача;
- вхід у систему;

- вихід із системи;
- обробка помилок та сповіщення.

Реєстрація нового користувача: користувач може зареєструватися на платформі, ввівши адресу електронної пошти й пароль. Всі дані проходять базову валідацію (наявність, мінімальна довжина пароля, валідність email). Після успішної реєстрації обліковий запис створюється у Firebase Auth, а додаткові дані (нікнейм, аватар тощо) зберігаються у Firestore.

Вхід у систему: для входу потрібно вказати email і пароль. Система перевіряє коректність даних та проводить автентифікацію через Firebase. У разі успіху користувач отримує доступ до захищених розділів платформи, а його дані завантажуються у глобальний стан застосунку.

Вихід із системи: користувач може вийти з облікового запису за допомогою відповідної кнопки, що очищає сесію та повертає інтерфейс до стану незалогіненого користувача.

Обробка помилок та сповіщення: усі помилки під час реєстрації, входу чи виходу відображаються у вигляді зрозумілих повідомлень — наприклад, “Невірний пароль” або “Користувача з таким email не знайдено”. Це забезпечує користувача чіткою інформацією про статус операцій.

Приклад коду для реєстрації користувача через email та пароль, зображено у лістингу 4.1

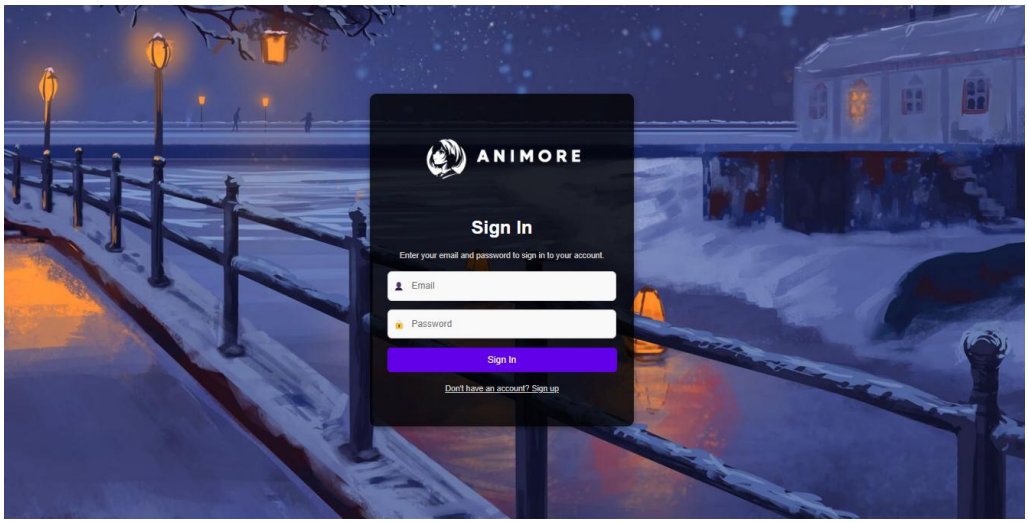


Рисунок 4.3 – Авторизація в особистому кабінеті

Лістинг 4.1 – Приклад коду для реєстрації користувача

```
import { getAuth, createUserWithEmailAndPassword } from
"firebase/auth";

  Реєстрація нового користувача у Firebase Auth
  @param {string} email // Адреса електронної пошти користувача
  @param {string} password // Пароль користувача
  @returns {Promise} // Promise з результатом реєстрації

export async function register(email, password) {
  const auth = getAuth();
  return await createUserWithEmailAndPassword(auth, email,
password);
}
```

Даний код використовує Firebase SDK для створення нового користувача, див. рисунок 4.4 та лістинг 4.2. Якщо email вже існує або пароль не відповідає вимогам, функція створить відповідну помилку.

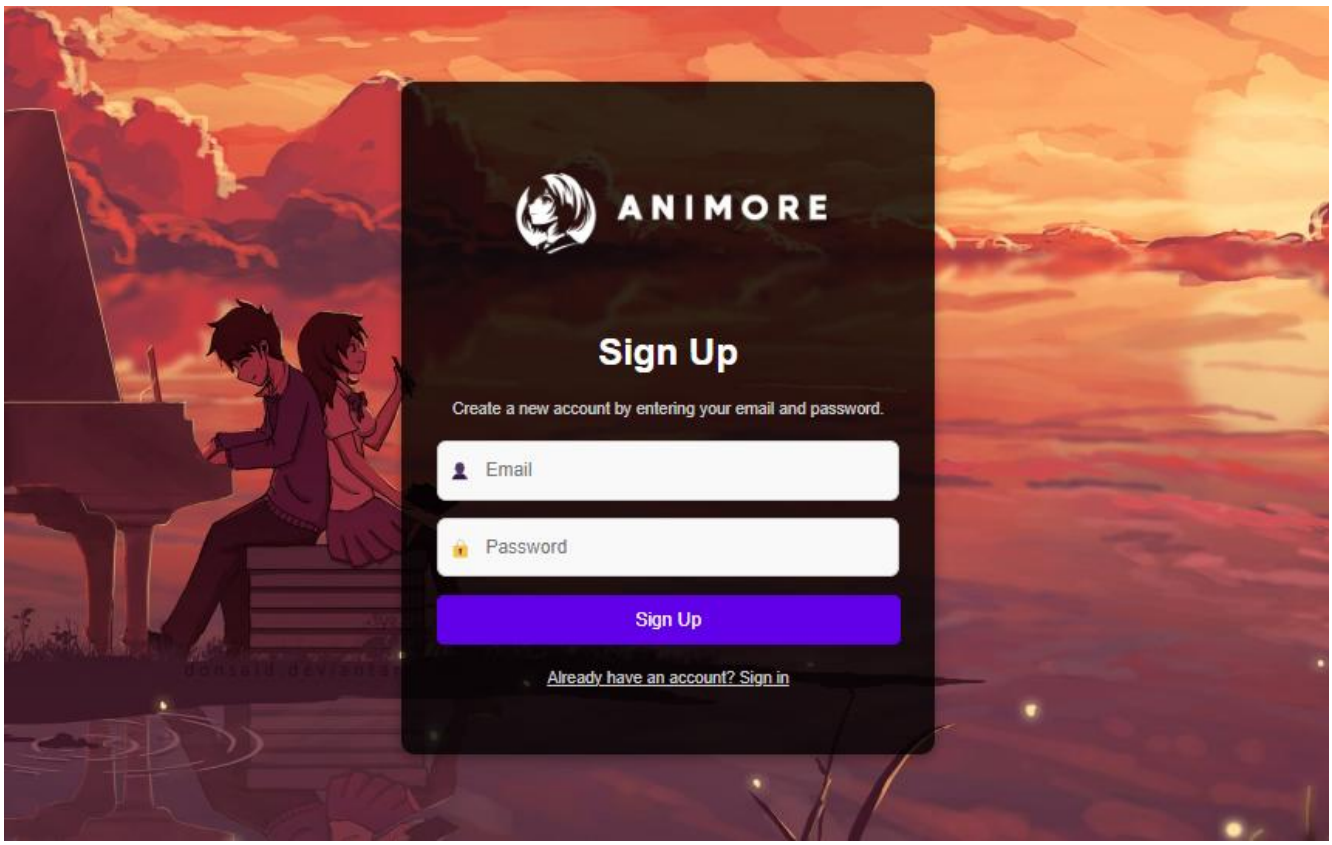


Рисунок 4.4 – Реєстрація на сайті

Лістинг 4.2 - Використання у формі реєстрації

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

```
import { register } from "../utils/auth";

function RegisterForm() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [message, setMessage] = useState("");

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      await register(email, password);
      setMessage("Реєстрація успішна!");
    } catch (error) {
      setMessage(error.message);
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <input
        type="email"
        placeholder="Email"
        value={email}
        required
        onChange={e => setEmail(e.target.value)}
      />
      <input
        type="password"
        placeholder="Пароль"
        value={password}
        required
        minLength={6}
        onChange={e => setPassword(e.target.value)}
      />
      <button type="submit">Зареєструватися</button>
      {message && <div>{message}</div>}
    </form>
  );
}

export default RegisterForm;
```

Дані користувачів не зберігаються у відкритому вигляді, а всі запити до захищених ресурсів перевіряються через токени Firebase.

Для Firestore налаштовано гнучкі правила доступу: незалогінені користувачі можуть лише переглядати загальний контент, а змінювати/додавати дані можуть тільки автентифіковані.

4.3 Робота з базою даних Firestore

Усі основні дані платформи AniMore зберігаються у хмарній NoSQL-базі даних Cloud Firestore від Firebase, рисунок 4.5. Firestore забезпечує масштабованість, роботу в реальному часі, гнучку структуру та просту інтеграцію з фронтендом через офіційний SDK.

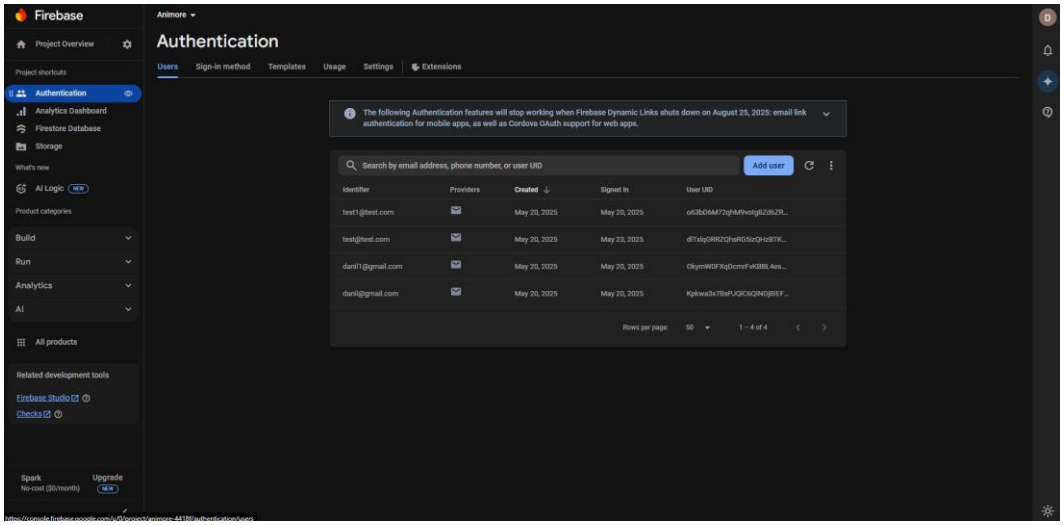


Рисунок 4.5 – БД вебсистеми

Структура даних (зображено на рисунку 4.6) у Firestore організовані у вигляді колекцій та документів:

- колекція(users);
- колекція anime;
- підколекція comments;
- колекція subscriptions.

Колекція users зберігає профілі користувачів. Кожен документ має унікальний ідентифікатор (userId) та містить поля nickname, email, avatarUrl, масив улюблених аніме (favorites), дату реєстрації, роль користувача тощо.

Колекція anime містить інформацію про всі аніме на платформі. Кожен документ (animeId) включає назву, опис, жанр, рік випуску, рейтинг, посилання на зображення, список епізодів та інші метадані.

У кожному документі колекції anime створюється підколекція comments, де зберігаються всі коментарі користувачів до цього аніме. Кожен документ у підколекції містить автора, текст, дату створення і, за потреби, посилання на аватар користувача.

Колекція subscriptions зберігає інформацію про підписки користувачів на аніме або інші події. Документи містять поле userId та масив ідентифікаторів підписок.

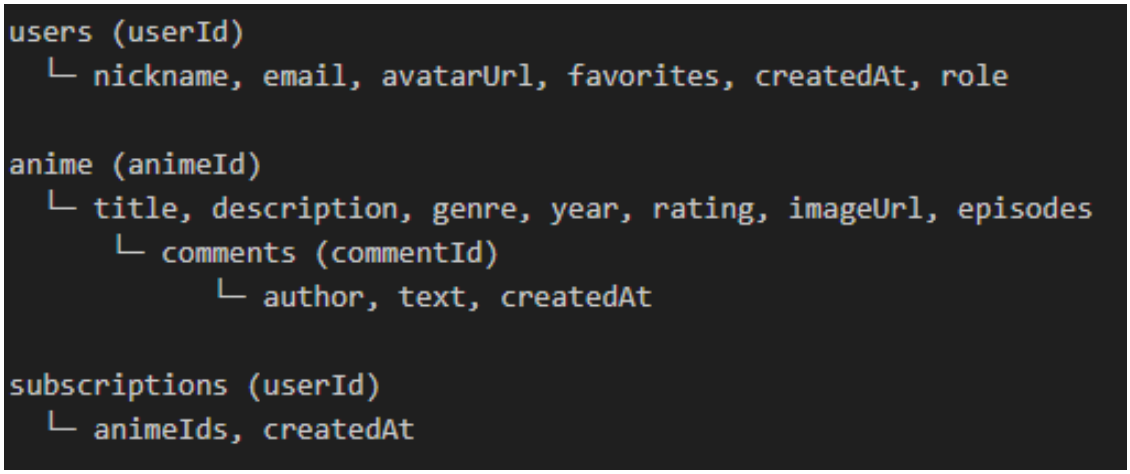


Рисунок 4.6 – Структура даних

На головній сторінці (наприклад, у файлі /pages/index.jsx) отриманий список аніме передається у компонент, який відображає картки аніме у вигляді сітки, зображено у лістингу 4.3. Кожна картка містить зображення, назву, короткий опис, рейтинг та кнопки для взаємодії (додавання до улюбленого, перегляд деталей).

Лістинг 4.3 – фрагмент коду відображення на головній сторінці

```
import { useEffect, useState } from "react";
import { fetchAnimeList } from "../utils/anime";
import AnimeCard from "../components/AnimeCard";

export default function HomePage() {
  const [animeList, setAnimeList] = useState([]);

  useEffect(() => {
    fetchAnimeList().then(setAnimeList);
```

```
}, []);

return (
  <div className="anime-grid">
    {animeList.map(anime => (
      <AnimeCard key={anime.id} anime={anime} />
    ))}
  </div>
); }
```

Особливості роботи з Firestore у проєкті AniMore полягають у тому, що ця база даних дозволяє реалізувати сучасний, інтерактивний і масштабований вебзастосунок із мінімальними зусиллями щодо підтримки інфраструктури. Однією з ключових переваг Firestore є підтримка роботи в реальному часі: для окремих компонентів, наприклад, блоку коментарів, використовується підписка на зміни у базі даних. Це означає, що коли хтось із користувачів додає або видаляє коментар, інтерфейс автоматично оновлюється для всіх інших користувачів без необхідності перезавантаження сторінки.

Ще однією важливою особливістю є можливість виконувати складні запити з фільтрацією та сортуванням. Наприклад, користувач може переглядати аніме за певним жанром, роком випуску або рейтингом, а Firestore дозволяє легко реалізувати такі запити без додаткової серверної логіки.

Безпека даних у Firestore забезпечується за допомогою спеціальних правил доступу (Firestore Security Rules). Вони дозволяють гнучко налаштовувати, хто і які дані може читати чи змінювати, залежно від статусу автентифікації користувача. Наприклад, лише авторизований користувач може додавати коментарі або змінювати власний профіль, тоді як переглядати аніме можуть усі відвідувачі.

Загалом, використання Firestore надає проєкту такі переваги, як гнучкість структури даних (можливість додавати нові поля чи підколекції без зміни всієї схеми), автоматичне масштабування під навантаження та простота інтеграції — адже доступ до даних здійснюється напряму з фронтенду через Firebase SDK. Це дозволяє швидко впроваджувати новий функціонал і забезпечує надійну роботу платформи навіть при зростанні кількості користувачів.

4.4 Завантаження та оптимізація аватара через Cloudinary

Основний сценарій роботи із завантаженням аватара через Cloudinary у проєкті AniMore побудований так, щоб зробити цей процес максимально простим і зручним для користувача. Коли користувач знаходиться на сторінці свого профілю, він може натиснути кнопку для зміни аватара та обрати нове зображення зі свого пристрою. Одразу після вибору система відображає попередній перегляд нового аватара, щоб користувач міг переконатися у правильності вибору.

Після підтвердження вибору зображення на фронтенді формується спеціальний об’єкт FormData, у який додається сам файл зображення та службова інформація, необхідна для завантаження на Cloudinary (наприклад, upload_preset, cloud_name тощо). Далі цей об’єкт надсилається POST-запитом на API Cloudinary. У відповідь сервіс повертає об’єкт із посиланням (secure_url) на оптимізоване зображення, яке вже готове до використання у вебзастосунку, приклад коду зображено у лістингу 4.4.

Отримане посилання на зображення зберігається у профілі користувача у Firestore. Завдяки цьому новий аватар одразу відображається у всіх компонентах інтерфейсу, де використовується аватар користувача — наприклад, у шапці сайту, у коментарях чи на сторінці профілю.

Лістинг 4.4 – Фрагмент коду для завантаження аватара

```
export async function uploadAvatar(file) {
  const formData = new FormData();
  formData.append("file", file);
  formData.append("upload_preset", "YOUR_UPLOAD_PRESET");

  const res = await
fetch("https://api.cloudinary.com/v1_1/YOUR_CLOUD_NAME/image/upload"
, {
    method: "POST",
    body: formData,
  });
  const data = await res.json();
```

```
  return data.secure_url;
}
```

4.5 Відображення відео та епізодів

Однією з ключових функцій AniMore є можливість переглядати епізоди аніме безпосередньо на платформі. Для цього реалізовано інтегрований відеоплеєр, який дозволяє користувачу зручно перемикатися між серіями, контролювати відтворення та отримувати якісний досвід перегляду, зображено на рисунку 4.7.

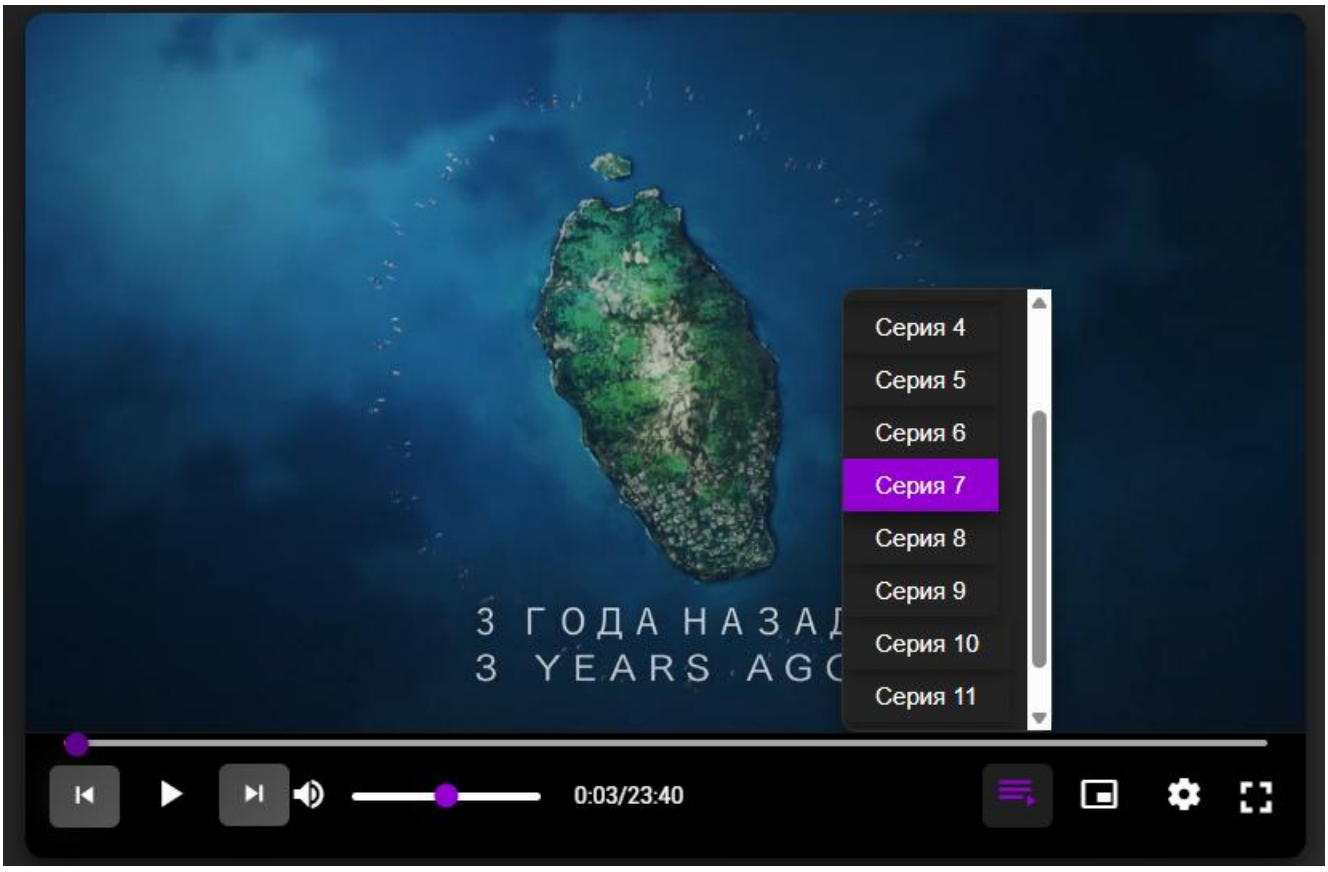


Рисунок 4.7 – Відеоплеєр сайту Animore

Архітектура та логіка роботи з епізодами в AniMore побудована таким чином, щоб забезпечити зручне зберігання, отримання та перегляд серій аніме. У

Firestore для кожного аніме зберігається масив епізодів, де кожен епізод містить унікальний ідентифікатор, назву, опис, посилання на відеофайл (videoUrl), номер серії, дату виходу та інші метадані. Така структура дозволяє ефективно організувати дані, як показано на рисунку 4.8.

Під час завантаження сторінки аніме застосунок виконує запит до Firestore, отримує масив епізодів і зберігає його у стані компонента. Це дає змогу динамічно відображати список серій і передавати їх у відеоплеєр. На сторінці аніме під основною інформацією розташовується компонент відеоплеєра, який за замовчуванням відтворює перший епізод, але користувач може обрати будь-яку серію зі списку. При виборі іншого епізоду змінюється джерело відео у плеєрі, як це показано у лістингу 4.5.

Інтерфейс вибору епізоду реалізовано у вигляді списку кнопок або вкладок під плеєром чи збоку від нього. Кожна кнопка містить номер і назву серії, а активна серія виділяється стилем. Клік по кнопці перемикає відеоплеєр на відповідний епізод. Сам відеоплеєр підтримує стандартні функції: відтворення та паузу, перемотування, зміну гучності, повноекранний режим. Для зручності користувача також можна додати індикатор прогресу, відображення назви поточного епізоду та повідомлення про помилки, наприклад, якщо відео недоступне.

Важливою особливістю є адаптивність інтерфейсу: на мобільних пристроях відеоплеєр і список епізодів займають всю ширину екрану, а навігація між серіями може бути реалізована свайпом або випадаючим списком, що забезпечує комфортний перегляд незалежно від типу пристрою.

```
anime (animeId)
  └ episodes: [
      { id: "ep1", title: "Серія 1", videoUrl: "...", description: "...", releaseDate: "...", ... },
      { id: "ep2", title: "Серія 2", videoUrl: "...", ... },
      ...
    ]
```

Рисунок 4.8 – Зберігання масивів в Firestore

Лістинг 4.5 - Приклад коду для відображення відеоплеєра

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

function AnimePlayer({ episodes }) {
  // Стан для поточного епізоду
  const [currentEpisode, setCurrentEpisode] = useState(0);

  // Якщо епізоди не завантажені
  if (!episodes || episodes.length === 0) {
    return <div>Епізоди недоступні</div>;
  }

  return (
    <div className="anime-player">
      <h3>{episodes[currentEpisode].title}</h3>
      <video
        key={episodes[currentEpisode].id}
        controls
        width="100%"
        src={episodes[currentEpisode].videoUrl}
        poster={episodes[currentEpisode].posterUrl}
        style={{ borderRadius: 8, background: "#000" }}
      >
        Ваш браузер не підтримує відео.
      </video>
      <div   className="episodes-list"   style={{   marginTop:   16,
display: "flex", gap: 8, flexWrap: "wrap" }}>
        {episodes.map((ep, idx) => (
          <button
            key={ep.id}
            onClick={() => setCurrentEpisode(idx)}
            style={{
              padding: "8px 16px",
              borderRadius: 4,
              background:  idx  ===  currentEpisode  ?  "#1976d2"  :
"#eee",
              color: idx === currentEpisode ? "#fff" : "#333",
              border: "none",
              cursor: "pointer",
              fontWeight: idx === currentEpisode ? "bold" : "normal"
            }}
          >
            {ep.title || `Серія ${idx + 1}`}
          </button>
        ))}
      </div>
    </div>
  );
}

export default AnimePlayer;
```

4.6 Адаптивність та доступність

Інтерфейс AniMore розроблений із максимальним урахуванням адаптивності, щоб забезпечити комфортну роботу користувачів на будь-яких пристроях — від великих моніторів до смартфонів.

Для цього використовуються сучасні підходи до верстки:

- flexbox та CSS Grid дозволяють компонувати елементи у сітки, які автоматично перебудовуються залежно від ширини екрана;
- всі основні блоки (шапка, сітка карток, бокові панелі, футер) мають гнучкі відступи та розміри, що змінюються на різних breakpoint’ах;
- на мобільних пристроях навігація перетворюється на бургер-меню, а сітка карток — на вертикальний список.

Для підвищення доступності застосунку AniMore було впроваджено низку сучасних рішень, які роблять платформу зручною для максимально широкої аудиторії, включаючи людей із особливими потребами. Приклад сторінок релізу зображено на рисунках 4.9-4.10.

Рисунок 4.9 – Приклад сторінки релізу

В інтерфейсі активно використовуються семантичні HTML-теги, такі як header, nav, main, section та footer, що значно покращує взаємодію зі скрінрідерами та іншими допоміжними технологіями. Для опису ролі елементів, стану кнопок і надання додаткових підказок користувачам із порушеннями зору застосовуються aria-атрибути.

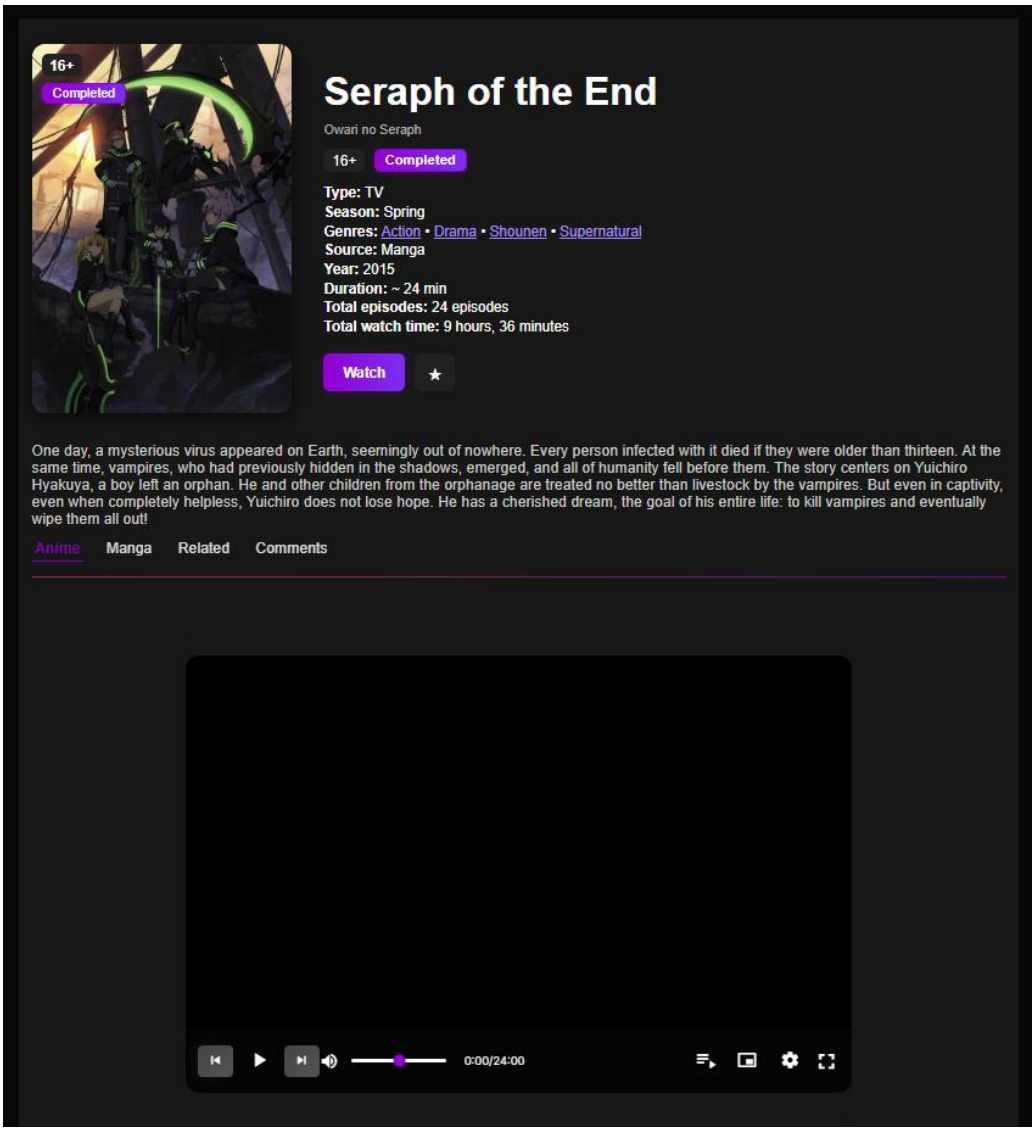


Рисунок 4.10 – Приклад сторінки релізу

Особлива увага приділяється візуальній доступності: у дизайні дотримано достатнього контрасту між текстом і фоном, а також акцентними елементами, що відповідає сучасним стандартам WCAG.

4.7 Додаткові можливості та інтеграції

На головній сторінці реалізовано інструмент для пошуку та фільтрації аніме. Користувач може швидко знайти потрібний тайтл за назвою, а також відфільтрувати список за жанром чи рейтингом, фрагмент коду зображено у лістингу 4.6. Для цього використовуються випадаючі списки та поле пошуку, які динамічно змінюють відображення карток аніме без перезавантаження сторінки.

Лістинг 4.6 – Пошук та фільтрація

```
function AnimeFilters({ filters, setFilters }) {
  return (
    <div className="filters">
      <input
        type="text"
        placeholder="Пошук аніме..."
        value={filters.query}
        onChange={e    =>    setFilters(f    =>    ({    ...f,    query:
e.target.value }))}
      />
      <select
        value={filters.genre}
        onChange={e    =>    setFilters(f    =>    ({    ...f,    genre:
e.target.value }))}
      >
        <option value="">Усі жанри</option>
        <option value="action">Екшн</option>
        <option value="romance">Романтика</option>
      </select>
    </div>
  );
}
```

Для оперативного зворотного зв’язку з користувачем у застосунку використовуються toast-сповіщення, приклад коду зображено у лістингу 4.7. Вони з’являються у зверху екрана після виконання дії (наприклад, додавання аніме до улюбленого, успішна реєстрація, помилка при вході тощо) і автоматично зникають через кілька секунд, приклад зображено на рисунку 4.10. Це дозволяє користувачу одразу отримувати інформацію про результат своїх дій, не відволікаючись від основного контенту.

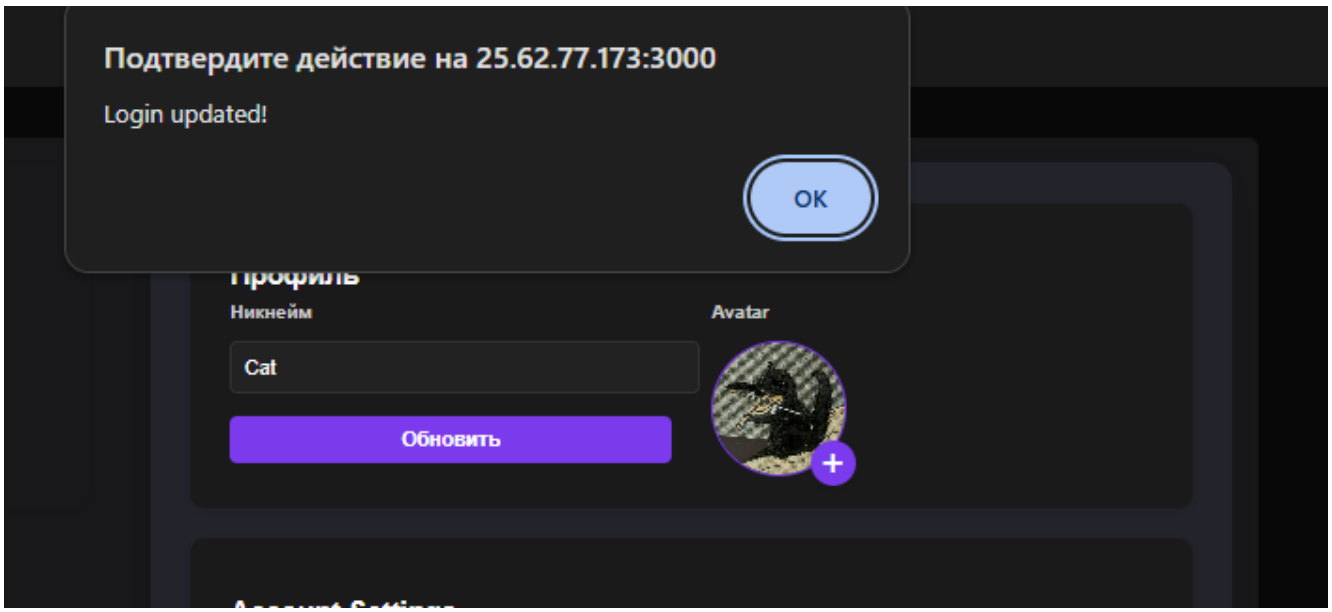


Рисунок 4.10 – Toast-сповіщення

Лістинг 4.7 – Фрагмент коду з Toast-сповіщенням

```
function Toast({ message, type }) {
  return (
    <div className={`toast toast-${type}`}>
      {message}
    </div>
  );
}
```

Для адміністраторів реалізовано окремий інтерфейс модерації, який дозволяє переглядати всі коментарі, видаляти небажані або порушуючі правила повідомлення, а також керувати контентом на платформі, зображено у лістингу 4.8. Це забезпечує підтримку якісного та безпечного середовища для всіх користувачів.

Лістинг 4.8 – Приклад логіки видалення коментаря

```
import { doc, deleteDoc } from "firebase/firestore";
import { db } from "../firebase/config";

export async function deleteComment(animeId, commentId) {
  const commentRef = doc(db, "anime", animeId, "comments",
commentId);
  await deleteDoc(commentRef); }
```

ВИСНОВКИ

У ході виконання дипломної роботи всі поставлені завдання були реалізовані в повному обсязі. Проведено аналіз основних понять і сучасних підходів до створення веб-платформи для перегляду аніме та манги, з урахуванням питань користувацького досвіду, безпеки, адаптивності та інтеграції із хмарними сервісами.

У пояснювальній записці детально розглянуто ключові етапи розробки веб-додатку AniMore із застосуванням сучасних технологій та бібліотек. Визначено вимоги до системи, розроблено архітектуру застосунку, підібрано оптимальні інструменти для клієнтської та серверної частини. Особливий акцент зроблено на організації бази даних у Firestore, інтеграції авторизації через Firebase Authentication, а також використанні Cloudinary для зберігання та оптимізації зображень.

У процесі розробки було створено зручний та зрозумілий інтерфейс. Взаємодія користувача з платформою реалізована на основі сучасного технологічного стеку: Next.js і React для фронтенду, Firebase для бекенду та зберігання даних.

У результаті створено повноцінну веб-платформу AniMore, яка надає користувачам можливість переглядати аніме та мангу, додавати їх до улюбленого, залишати коментарі, переглядати відео-епізоди та керувати власним профілем. Компоненти та модулі побудовані з урахуванням принципів модульності й повторного використання коду, що спрощує підтримку та подальший розвиток функціоналу.

Застосування сучасних технологій і хмарних сервісів забезпечило високу продуктивність, безпеку та масштабованість платформи, а також позитивний користувацький досвід на різних пристроях.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. HTTP Live Streaming (HLS) Authoring Specification for Apple Devices [Electronic resource]. Apple Developer Documentation, 2023. URL: https://developer.apple.com/documentation/http-live-streaming/hls-authoring-specification-for-apple-devices

2. Глушаков І. В. Основи веб-програмування : навч. посіб. / І. В. Глушаков. – Харків : ХНУРЕ, 2018. – 210 с.

3. The State of MPEG DASH 2015. Streaming Media, Oct 2015. URL: https://www.streamingmedia.com/Articles/Editorial/Featured-Articles/The-State-of-MPEG-DASH-2015-102826.aspx

4. ISO/IEC 25010:2011 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. – International Organization for Standardization, 2011.

5. Anime Streaming and Community — A UX/UI Case Study. UX Planet, Oct 2020. URL: https://uxplanet.org/anime-streaming-and-community-a-ux-ui-case-study-156a7d6b8a9e

6. Фіалко В. Ю. Основи проєктування баз даних : навч. посіб. / В. Ю. Фіалко. – Тернопіль : ТНТУ, 2019. – 248 с.

7. ISO/IEC 27001:2013 Information technology – Security techniques – Information security management systems – Requirements. – International Organization for Standardization, 2013.

UX/UI Case Study: Anime Streaming App / Jay Kadam. Medium, Apr 2021. URL: https://j4ykadam.medium.com/ux-ui-case-study-anime-streaming-app-119d62fa9e7

9. Анісімов А.В. Інформаційні системи та бази даних: Навчальний посібник для студентів факультету комп’ютерних наук та кібернетики. / Анісімов А.В., Кулябко П.П. − Київ. – 2017. – 110 с.

10. Node.js Web Development For Beginners: A Step By Step Guide. Amazon Digital, 2023. URL: https://www.amazon.com/Node-js-Development-Beginners-Step-Step/dp/B0D2P99QGV

11. Іванов, К.К. Проектування бази даних. Роль процесу в створенні інформаційної системи / К.К. Іванов, А.А. Єфремов, І.А. Ващенко. - Текст: безпосередній // Молодий вчений. - 2016. - № 18 (122). - С. 40-42. - URL: https://moluch.ru/archive/122/33704/

12. Структура веб-сайтів, різновиди веб-сайтів [Електронний ресурс] URL: http://urokinformatiki.in.ua/urok-26-struktura-veb-sajtiv_riznovidi-veb-sajtiv/

13. Тарасенко В. В. Сучасні тренди та тенденції розвитку веб-дизайну. Proceedings of III International Scientific and Practical Conference, м. Львів, 14 черв. 2022 р. 2022. С. 1236–1239

14. What Is ISO 25010? / Perforce Software Blog, 2020. URL: https://www.perforce.com/blog/sca/what-is-iso-5055

15. Crunchyroll / Anime Website, 2025. URL https://www.crunchyroll.com

16. Anilibria / Anime Website, 2025 URL https://www.anilibria.tv

17. MangaDex / Manga Website, 2025 URL https://mangadex.org

18. YummiAnime / Anime Website, 2025 URL https://yummyanime.club

19. Офіційна документація Firebase Authentication / URL https://firebase.google.com/docs/auth

20. Stack Overflow — відповіді на питання по розробці / URL https://stackoverflow.com/

21. Офіційна документація Cloudinary / URL https://cloudinary.com/documentation

22. Документація по доступності (Web Accessibility Initiative, WAI) / URL https://www.w3.org/WAI/

23. Документація по використанню useRouter у Next.js / URL https://nextjs.org/docs/api-reference/next/router

24. Офіційна документація Firebase / URL https://firebase.google.com/docs

ДОДАТОК А
ЛІСТИНГИ ПРОГРАМ

Лістинг А.1 – Файл Header.jsx

```
import React, { useState, useEffect } from 'react';
import Link from 'next/link';
import { useRouter } from 'next/router';
import { auth } from '../utils/firebase';
import styles from '../styles/Header.module.css';
import { useUser } from '../context/UserContext';

export default function Header() {
  const [isDropdownOpen, setDropdownOpen] = useState(false);
  const [isDarkTheme, setIsDarkTheme] = useState(false);
  const [search, setSearch] = useState('');
  const { userAvatar, setUserAvatar } = useUser();
  const router = useRouter();

  useEffect(() => {
    const savedAvatar = localStorage.getItem('userAvatar');
    if (savedAvatar) {
      setUserAvatar(savedAvatar);
    }
  }, [setUserAvatar]);

  const toggleDropdown = () => {
    setDropdownOpen(!isDropdownOpen);
  };

  const handleLogout = async () => {
    try {
      await auth.signOut();
      localStorage.removeItem('userAvatar');
      setUserAvatar('');
      alert('You have successfully logged out!');
      router.push('/auth');
    } catch (error) {
      console.error('Logout error:', error);
      alert('Failed to log out. Please try again.');
    }
  };

  const toggleTheme = () => {
    setIsDarkTheme(!isDarkTheme);
    document.documentElement.setAttribute(
      'data-theme',
      !isDarkTheme ? 'dark' : 'light'
    );
```

```
  };

  useEffect(() => {
    const savedTheme = localStorage.getItem('theme') || 'light';
    setIsDarkTheme(savedTheme === 'dark');
    document.documentElement.setAttribute('data-theme', savedTheme);
  }, []);

  useEffect(() => {
    localStorage.setItem('theme', isDarkTheme ? 'dark' : 'light');
  }, [isDarkTheme]);

  const handleSearch = (e) => {
    e.preventDefault();
    if (search.trim()) {

router.push(`/search?query=${encodeURIComponent(search.trim())}`);
      setSearch('');
      setDropdownOpen(false);
    }
  };

  return (
    <header className={styles.header}>
      <div className={styles.logo}>
        <Link href="/">AniMore</Link>
      </div>
      <nav className={styles.nav}>
        <Link href="/">Home</Link>
        <Link href="/releases">Releases</Link>
        <Link href="/schedule">Schedule</Link>
      </nav>
      <div className={styles.profile}>
        <form className={styles.searchForm} onSubmit={handleSearch}>
          <input
            type="text"
            placeholder="Search..."
            value={search}
            onChange={e => setSearch(e.target.value)}
            className={styles.searchInput}
          />
        </form>
        <div                          className={styles.avatarWrapper}
onClick={toggleDropdown}>
          {userAvatar ? (
            <img            src={userAvatar}            alt="Profile"
className={styles.avatar} />
          ) : (
            <img         src="/profile-icon.png"         alt="Profile"
className={styles.avatar} />
          )}
        </div>
        {isDropdownOpen && (
```

```
          <div className={styles.profileDropdown}>
            <Link href="/profile">Profile</Link>
            <Link href="/ignore-list">Ignore List</Link>
            <Link href="/security">Security & Login</Link>
            <div className={styles.themeToggle}>
              <span>Site Theme</span>
              <label className={styles.switch}>
                <input
                  type="checkbox"
                  checked={isDarkTheme}
                  onChange={toggleTheme}
                />
                <span className={styles.slider}></span>
              </label>
            </div>
            <button                              onClick={handleLogout}
className={styles.logoutButton}>
              Logout
            </button>
          </div>
        )}
      </div>
    </header>
  );
}
```

Лістинг А.2 – Файл Footer.jsx

```
import React from 'react';
import styles from '../styles/Footer.module.css';
import Link from 'next/link';

export default function Footer() {
  return (
    <footer className={styles.footerContainer}>
      <div className={styles.footerContent}>
        <div className={styles.leftSection}>
          <h2 className={styles.siteName}>AniMore</h2>
          <p  className={styles.slogan}>Your  anime.  Your  manga.  Your
world.</p>
        </div>

        <div className={styles.rightSection}>
          <p className={styles.disclaimer}>
            All materials on the site are provided for home preview
only.
            <br />
            For  copyright  infringement  issues,  please  contact:  <a
href="mailto:copyrights@animore.com">copyrights@animore.com</a>
            <br />
            For   advertising   and   cooperation   inquiries:   <a
href="mailto:contact@animore.com">contact@animore.com</a>
          </p>
        </div>
```

```
      </div>

      <div className={styles.footerBottom}>
        <div className={styles.links}>
          <Link href="/rules" className={styles.link}>Rules</Link>
          <Link    href="/privacy"    className={styles.link}>Privacy
Policy</Link>
          <Link                                          href="/profile"
className={styles.link}>Profile</Link>
        </div>
        <p className={styles.copyright}>© 2025 • AniMore</p>
      </div>
    </footer>
  );
}
```

Лістинг А.3 – Файл [id].jsx

```
import { useRouter } from 'next/router';
import { useState, useRef } from 'react';
import { animeDB } from '../../data/animeDB';
import styles from '../../styles/AnimeTemplate.module.css';
import Link from 'next/link';
import                    MangaChaptersReader                    from
'../../components/MangaChaptersReader';

export default function AnimePage() {
  const router = useRouter();
  const { id } = router.query;
  const [iframeLoaded, setIframeLoaded] = useState(false);

  const [comment, setComment] = useState('');
  const [comments, setComments] = useState([]);
  const [activeTab, setActiveTab] = useState('anime');
  const videoRef = useRef(null);
  const commentsRef = useRef(null);

   if  (!id)  return  <div  style={{  color:  '#fff',  padding:  32
}}>Loading...</div>;

  const data = animeDB[String(id)];

   if  (!data)  return  <div  style={{  color:  '#fff',  padding:  32
}}>Anime not found</div>;

  const handleCommentSubmit = (e) => {
    e.preventDefault();
    if (comment.trim()) {
       setComments([{ text: comment, author: 'You', date: new Date()
}, ...comments]);
      setComment('');
    }
  };
```

```
  const mangaDexIds = {
    '8325': '62040a44-0935-46b7-a691-5ae5833af0ae',
    'jujutsu_kaisen': 'a2e53e4e-2a3c-4d7c-9f6d-2e7e9d8e0f5c',
  };
  const mangaId = mangaDexIds[id] || null;

  return (
    <div className={styles.wrapper}>
      <div className={styles.mainInfo}>
        <div className={styles.posterBlock}>
                        <img    src={data.poster}    alt={data.title}
className={styles.poster} />
          <div className={styles.age}>{data.age}</div>
          <div className={styles.status}>{data.status}</div>
        </div>
        <div className={styles.infoBlock}>
          <h1 className={styles.title}>{data.title}</h1>
          <div className={styles.subtitle}>{data.subtitle}</div>
          <div className={styles.tags}>
            <span className={styles.tag}>{data.age}</span>
            <span className={styles.statusBtn}>{data.status}</span>
          </div>
          <div className={styles.meta}>
            <div><b>Type:</b> {data.type}</div>
            <div><b>Season:</b> {data.season}</div>
            <div>
              <b>Genres:</b>{' '}
              {data.genres.map((genre, idx) => (
                <span key={genre}>
                  <Link

href={`/anime/genre/${encodeURIComponent(genre)}`}
                          style={{  color:  '#a78bfa',  textDecoration:
'underline', cursor: 'pointer' }}>
                    {genre}
                  </Link>
                  {idx < data.genres.length - 1 && ' • '}
                </span>
              ))}
            </div>
            <div><b>Source:</b> {data.origin}</div>
            <div><b>Year:</b> {data.year}</div>
            <div><b>Duration:</b> {data.duration}</div>
            <div><b>Total episodes:</b> {data.episodes}</div>
            <div><b>Total watch time:</b> {data.totalTime}</div>
          </div>
          <div className={styles.buttons}>
            <button
              className={styles.playBtn}
              onClick={() => {
                if (videoRef.current) {
```

```
                        videoRef.current.scrollIntoView({  behavior:
'smooth', block: 'center' });
                  }
                }}
              >
                Watch
              </button>
              <button className={styles.iconBtn}>★</button>
            </div>
          </div>
        </div>
        <div className={styles.description}>
          {data.description}
        </div>

        <div className={styles.tabs}>
          <span
               className={activeTab  ===  'anime'  ?  styles.tabActive  :
styles.tab}
            onClick={() => setActiveTab('anime')}
            style={{ cursor: 'pointer' }}
          >
            Anime
          </span>
          <span
               className={activeTab  ===  'manga'  ?  styles.tabActive  :
styles.tab}
            onClick={() => setActiveTab('manga')}
            style={{ cursor: 'pointer' }}
          >
            Manga
          </span>
          <span
              className={activeTab  ===  'related'  ?  styles.tabActive  :
styles.tab}
            onClick={() => setActiveTab('related')}
            style={{ cursor: 'pointer' }}
          >
            Related
          </span>
          <span
            className={styles.tab}
            style={{ cursor: 'pointer' }}
            onClick={() => {
                      commentsRef.current?.scrollIntoView({  behavior:
'smooth', block: 'start' });
              setActiveTab('comments');
            }}
          >
            Comments
          </span>
        </div>
```

```
      <div className={styles.tabsDivider}></div>

      {activeTab === 'anime' && (
        <div className={styles.videoBlock} ref={videoRef}>
            {!iframeLoaded && <div className={styles.loader}>Loading
player...</div>}
          <iframe
            src={`/video/player.html?anime=${id}`}
            width="100%"
            height="605"
            allowFullScreen
            frameBorder="0"
            title={`Video player for ${data.title}`}
            aria-label={`Video player for ${data.title}`}
            style={{
              borderRadius: 12,
              background: "#000",
              margin: "32px 0",
              display: iframeLoaded ? 'block' : 'none'
            }}
            onLoad={() => setIframeLoaded(true)}
          />
        </div>
      )}

      {activeTab === 'manga' && (
        <div style={{ margin: '32px 0' }}>
          {mangaId ? (
            <MangaChaptersReader mangaId={mangaId} />
          ) : (
                <div  style={{  color:  '#fff',  textAlign:  'center',
padding: 32 }}>
              No manga found for this title.
            </div>
          )}
        </div>
      )}

      {activeTab === 'related' && (
         <div style={{ color: '#fff', textAlign: 'center', padding:
32 }}>
          Related titles information will appear here.
        </div>
      )}

      <div className={styles.tabsDivider}></div>
      <div ref={commentsRef}></div>

                           <form        className={styles.commentForm}
onSubmit={handleCommentSubmit}>
        <textarea
          className={styles.commentInput}
          placeholder="Write a comment..."
```

```
          value={comment}
          onChange={e => setComment(e.target.value)}
          rows={3}
        />
                                          <button         type="submit"
className={styles.commentBtn}>Send</button>
      </form>
      <div className={styles.commentsList}>
        {comments.length === 0 ? (
          <div className={styles.noComments}>Your first comment will
appear here</div>
        ) : (
          comments.map((c, idx) => (
            <div key={idx} className={styles.commentItem}>
              <div className={styles.commentAuthor}>{c.author}</div>
              <div className={styles.commentText}>{c.text}</div>
              <div className={styles.commentDate}>
                   {c.date.toLocaleString('en-US', { day: '2-digit',
month: '2-digit', year: '2-digit', hour: '2-digit', minute: '2-
digit' })}
              </div>
            </div>
          ))
        )}
      </div>
    </div>
  );
}
```

Лістинг А.4 – Файл accountsettings.js

```
import { useState, useEffect } from 'react';
import { auth, db } from '../../utils/firebase';
import { doc, getDoc, updateDoc } from 'firebase/firestore';
import styles from '../../styles/Profile.module.css';

export default function AccountSettings() {
  const [login, setLogin] = useState('');
  const [email, setEmail] = useState('');

  useEffect(() => {
    const fetchData = async () => {
      const user = auth.currentUser;
      if (user) {
        const userDoc = await getDoc(doc(db, 'users', user.uid));
        if (userDoc.exists()) {
          const data = userDoc.data();
          setLogin(data.login || '');
          setEmail(user.email || '');
        }
      }
    };
    fetchData();
```

```
  }, []);

  const handleUpdateLogin = async () => {
    const user = auth.currentUser;
    if (user) {
      await updateDoc(doc(db, 'users', user.uid), {
        login,
      });
      alert('Login updated!');
    }
  };

  return (
    <section id="account-settings" className={styles.sectionBlock}>
      <h2 className={styles.sectionTitle}>Account Settings</h2>
      <p className={styles.sectionDesc}>
        In this section, you can change your account data. Please be
careful, as some changes may be irreversible.
      </p>
      <div className={styles.formRow}>
        <div className={styles.formCol}>
          <label className={styles.label}>Login</label>
          <input
            type="text"
            value={login}
            onChange={(e) => setLogin(e.target.value)}
            className={styles.input}
            placeholder="Enter login"
          />
                              <button     onClick={handleUpdateLogin}
className={styles.button}>
            Update
          </button>
        </div>
        <div className={styles.formCol}>
          <label className={styles.label}>Email</label>
          <input
            type="text"
            value={email}
            className={styles.input}
            disabled
          />
        </div>
      </div>
    </section>
  );
}
 }
```

Лістинг А.5 – Файл globals.css

```
:root {
  --bg-deep: #080808; /* Page background */
  --bg-shade: #1A1A1A; /* Cards, sidebars, footer */
  --violet-dark: #4B0082; /* Gradients, icons, text background */
  --violet-base: #800080; /* Buttons, active tabs, hover effects */
   --violet-neon: #C78CFF; /* Neon glow: outlines, shadows, glow
effects */
  --text-light: #ECECEC; /* Main text */
}

[data-theme='light'] {
  --bg-deep: #f5f5f5; /* Light page background */
  --bg-shade: #ffffff; /* Light cards, sidebars, footer */
  --violet-dark: #4B0082; /* Gradients, icons, text background */
  --violet-base: #6200ea; /* Buttons, active tabs, hover effects */
   --violet-neon: #bb86fc; /* Neon glow: outlines, shadows, glow
effects */
  --text-light: #000000; /* Dark text */
}

body, html {
  height: 100%;
  margin: 0;
  padding: 0;
  background-color: var(--bg-deep);
  color: var(--text-light);
  font-family: 'Arial', sans-serif;
  flex-direction: column;
}

a {
  color: var(--violet-base);
  text-decoration: none;
}

a:hover {
  text-decoration: underline;
  color: var(--violet-neon);
}

main {
  flex: 1;
  margin: 0;
  padding: 0;
  display: flex;
  flex-direction: column;
  background-color: var(--bg-deep);
}
```

Національний університет “Запорізька політехніка”
Факультет комп’ютерних наук і технологій

Розробка вебсистеми представлення аніме/манги із використанням фреймворку Nextjs

Виконавець:
ПАЛЄНОВ Данило Андрійович
студент групи КНТ-512сп

Керівник:
КИРИЧЕК Галина Григорівна
к. т. н., доцент кафедри
комп’ютерних систем та мереж

2025

МЕТА РОБОТИ

Аніме та манга стають дедалі популярнішими серед молоді. Існує потреба у зручному сервісі для перегляду та обговорення аніме.

Мета — створити платформу з сучасним функціоналом для фанатів аніме, яка забезпечує зручний доступ до каталогу аніме, систему коментарів, управління профілем користувача та інтеграцію з хмарними сервісами для зберігання даних і медіафайлів.

ПРЕДМЕТ ДОСЛІДЖЕННЯ

Предметом дослідження є програмні засоби, архітектурні підходи та технології, що використовуються для створення сучасної веб-платформи, яка забезпечує перегляд аніме та читання манги онлайн.

Особлива увага приділяється вибору та застосуванню хмарних сервісів для зберігання даних (наприклад, Firestore), організації безпечної автентифікації користувачів (Firebase Auth), а також інтеграції з хмарними сховищами для медіафайлів (Firebase Storage, Cloudinary).

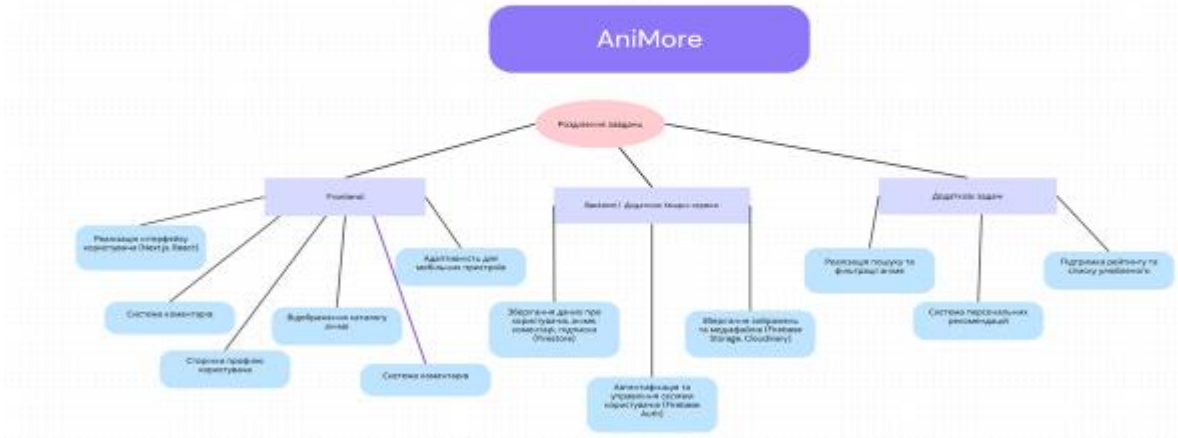
АКТУАЛЬНІСТЬ

- зростання популярності аніме та манги серед молоді й дорослих користувачів в Україні та світі;
- потреба у зручному, безпечному та функціональному сервісі з персоналізацією, рекомендаціями та інтерактивним спілкуванням;
- необхідність впровадження сучасних веб-технологій для забезпечення швидкої роботи, адаптивності та масштабованості платформи.

ВИМОГИ ДО ВЕБСИСТЕМИ

- Адаптивний сучасний інтерфейс для різних пристроїв
- Безпечна автентифікація та захист персональних даних користувачів
- Можливість персоналізації профілю (аватар, улюблене, підписки)
- Інтеграція з хмарними сервісами для зберігання даних і медіафайлів (Firebase, Cloudinary)
- Висока продуктивність і масштабованість

РОЗДІЛЕННЯ ЗАВДАНЬ

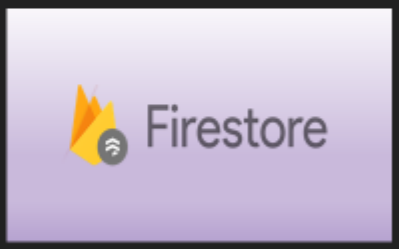


ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТИ

- **Next.js, React** — сучасні фреймворки для розробки швидкого та адаптивного інтерфейсу користувача.
- **Firebase** — використовується для зберігання даних (Firestore), автентифікації користувачів (Auth) та роботи з медіафайлами (Storage).
- **Cloudinary** — хмарний сервіс для зберігання та оптимізації зображень користувачів (аватари, постери аніме).
- **CSS-модулі** — забезпечують ізольовану та зручну стилізацію компонентів інтерфейсу.


ЧОМУ САМЕ FIRESTORE

- Гнучка структура даних: організація інформації у вигляді колекцій і документів, що дозволяє легко масштабувати та змінювати структуру бази.
- Масштабованість: автоматичне підлаштування під навантаження, можливість обслуговувати як невелику кількість користувачів, так і великі спільноти.
- Робота в реальному часі: миттєве оновлення даних на клієнті без перезавантаження сторінки.
- Інтеграція з іншими сервісами Firebase: спрощує автентифікацію, зберігання медіафайлів, аналітику та інші функції для комплексної роботи застосунку.



ВІДЕОПЛЕЄР

- Відеоплеєр розташований на сторінці аніме під основною інформацією та списком епізодів
- Підтримує стандартні функції: відтворення/пауза, перемотування, зміна гучності, повноекранний режим
- Дозволяє обирати епізод зі списку — автоматичне перемикання відео
- Адаптивний дизайн для зручного перегляду на мобільних пристроях
- Додаткові можливості: індикатор прогресу, відображення назви епізоду, повідомлення про помилки

СКЛАДНОЩІ РЕАЛІЗАЦІЇ ПРОЄКТУ

- Інтеграція з хмарними сервісами (Firebase, Cloudinary) вимагала налаштування безпечної взаємодії та оптимізації роботи з медіафайлами.
- Забезпечення захисту персональних даних користувачів та безпечної автентифікації.
- Реалізація адаптивного інтерфейсу для коректної роботи на різних пристроях і розмірах екранів.
- Організація ефективної структури даних у Firestore для швидкого доступу та масштабованості.
- Впровадження системи коментарів і рейтингу з урахуванням можливих навантажень.
- Забезпечення стабільної роботи відеоплеєра та підтримка різних форматів відео.
- Тестування та налагодження всіх компонентів для забезпечення зручності та безперебійної роботи сервісу.

ОБМЕЖЕННЯ ТА ПЕРСПЕКТИВИ РОЗВИТКУ

Обмеження поточної версії:

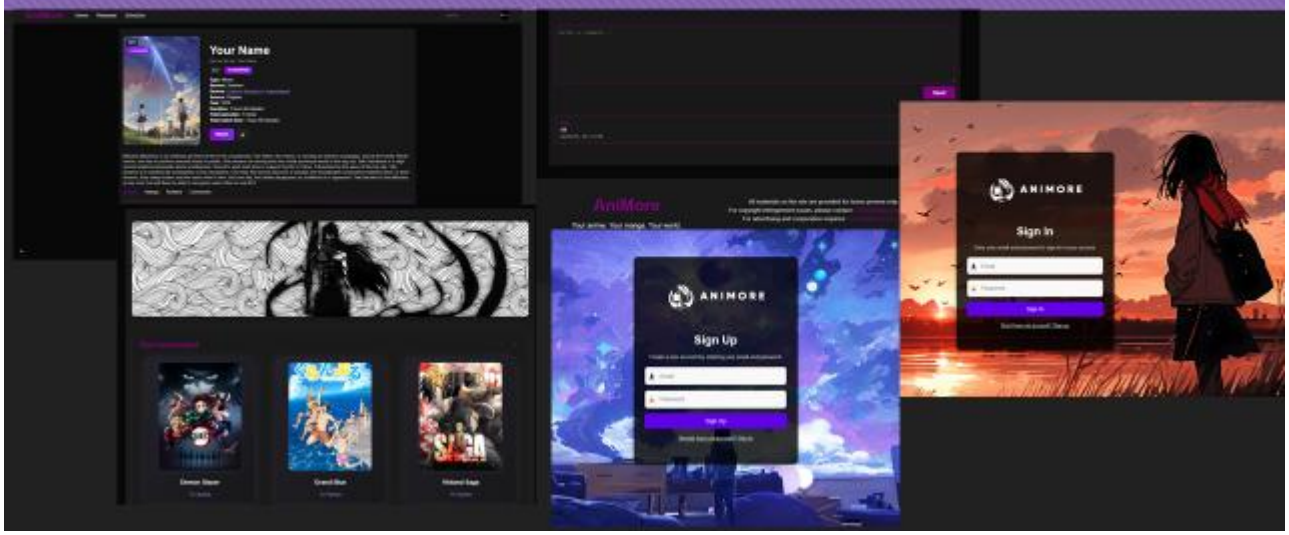
- На даному етапі проєкту не реалізовано власний відеохостинг, тому відтворення відео здійснюється через зовнішні сервіси або посилання.
- Відсутня підтримка live-трансляцій, що обмежує можливості проведення онлайн-подій або спільного перегляду.
- Не впроваджено розширену систему аналітики користувачів, тому немає детального збору статистики щодо активності, вподобань та взаємодії з платформою.

ОБМЕЖЕННЯ ТА ПЕРСПЕКТИВИ РОЗВИТКУ

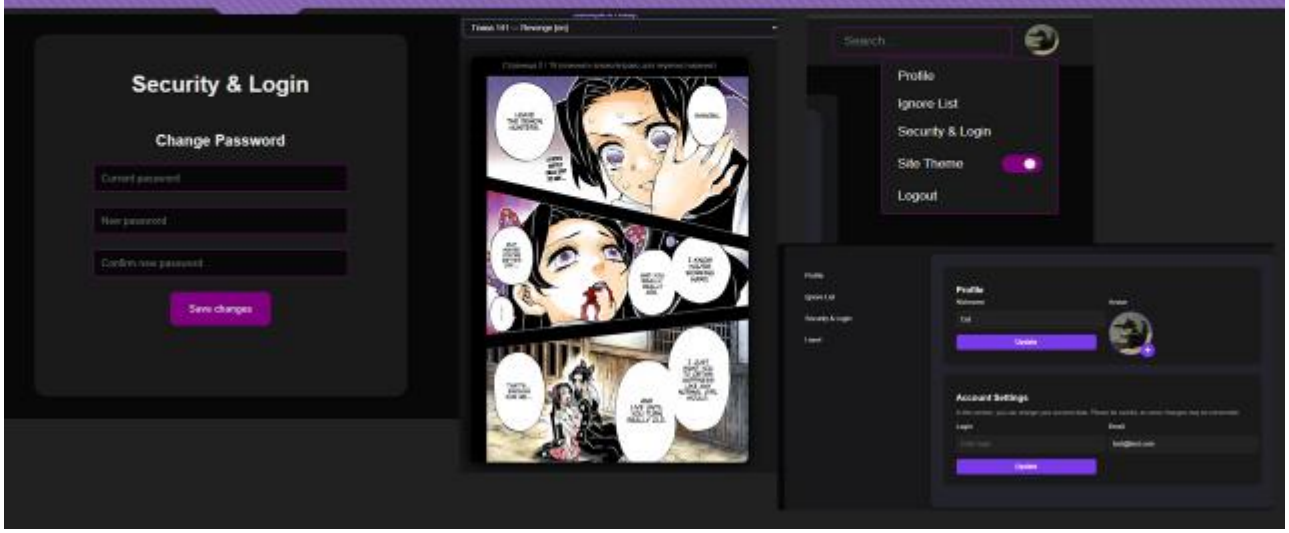
Перспективи розвитку:

- Планується створення мобільного застосунку для ще більшої зручності користувачів та розширення аудиторії.
- Передбачена інтеграція з соціальними мережами, що дозволить спростити процес авторизації, а також дасть змогу ділитися улюбленим контентом із друзями.
- У майбутньому планується впровадження мультимовного інтерфейсу, що зробить платформу доступною для ширшого кола користувачів з різних країн.

ПРИКЛАД ІНТЕРФЕЙСУ ВЕБСИСТЕМИ



ПРИКЛАД ІНТЕРФЕЙСУ ВЕБСИСТЕМИ



ВИСНОВКИ

У результаті виконання проєкту було розроблено сучасну веб-платформу AniMore для перегляду аніме та читання манги, яка поєднує зручний інтерфейс, персоналізовані функції та інтеграцію з хмарними сервісами. Реалізовано основний функціонал: каталог аніме, систему коментарів, профілі користувачів, відеоплеєр, пошук і фільтрацію, а також захист персональних даних.

Використання технологій Next.js, React та Firebase забезпечило гнучкість, масштабованість і швидку роботу платформи. Попри окремі обмеження, проєкт має значний потенціал для подальшого розвитку: впровадження мобільного застосунку, розширення аналітики, інтеграція з соціальними мережами та підтримка мультимовності.

AniMore є прикладом сучасного тематичного онлайн-сервісу, що відповідає актуальним потребам користувачів та сприяє розвитку IT-простору.