Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА СЕРВІСУ ДЛЯ КАТАЛОГІЗАЦІЇ І УПРАВЛІННЯ ПЕРЕГЛЯНУТОГО АНІМЕ ТА МАНГИ
(тема)

Виконав:
здобувач 4 року навчання,

групи ІТІНФ-21-2

Дрюков В.В.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Машталір С.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_ _ »_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Дрюкову Владиславу Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка сервісу для каталогізації і управління переглянутого аніме та манги

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 28 травня 2025 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, тематичні форуми, бібліотека для розробки користувацького інтерфейсу React, відкрите середовище виконання Node.js.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз існуючих методів для каталогізації і управління переглянутого аніме та манги.
2. Проєктування системи та опис основних компонентів.
3. Програмна реалізація застосунку «Персональна бібліотека аніме та манги».

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність каталогізації та управління переглянутими творами у жанрі аніме та манги, постановка задачі, лістинги реалізацій застосунку, тестові зображення.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	08.04.25-10.04.25	
3	Аналіз літератури з досліджуваної проблеми	11.04.25-14.04.25	
4	Аналіз технічних засобів	15.04.25-20.04.25	
5	Розробка методу	21.04.25-27.04.25	
6	Програмна реалізація	28.04.25-15.05.25	
7	Оформлення пояснювальної записки	15.05.25-25.05.25	
8	Перевірка на нормоконтроль	21.05.25-01.06.25	
9	Перевірка на плагіат	21.05.25-01.06.25	
10	Рецензування	21.05.25-01.06.25	
11	Підготовка презентації та доповіді	21.05.25-18.06.25	
12	Занесення роботи в електронний архів	02.06.25-18.06.25	
13	Попередній захист кваліфікаційної роботи	02.06.25-18.06.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ___проф. Машталір С. В._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 64 с., 7 табл., 22 рис., 30 джерел.

КОРИСТУВАЧ, АНІМЕ, МАНГА, ПЕРСОНАЛЬНА БІБЛІОТЕКА, ЗАСТОСУНОК, JAVASCRIPT, REACT, FASTAPI, VISUAL STUDIO CODE, POSTGRESQL, TAILWIND CSS, NODE.JS.

Об’єктом роботи є каталогізація та управління прочитаними японськими художніми творами, у категорії аніме та манга. Метою даної роботи є розробка застосунку, що призначений для легкої та ефективної каталогізації прочитаних творів та управління особистою бібліотекою користувача. Також застосунок дозволить користувачам планувати свій майбутній перегляд або читання, відстежувати прогрес у сторінках чи серіях, ставити оцінки, та вести записи про твори, з якими вони бажають ознайомитися.

У процесі розробки сервісу були використані сучасні технології, бібліотеки та фреймворки, такі як React для розробки застосунку, Tailwind CSS для ствоерення інтерфейсу, Node.js для розробки серверної частини, FastAPI для створення API та PostgreSQL для зберігання даних.

USER, ANIME, MANGA, PERSONAL LIBRARY, APPLICATION, JAVASCRIPT, REACT, FASTAPI, VISUAL STUDIO CODE, POSTGRESQL, TAILWIND CSS, NODE.JS.

The object of the work is the cataloging and management of read Japanese fiction works in the anime and manga categories. The aim of this qualification project is to develop an application designed for easy and efficient cataloging of read works and managing the user’s personal library. The application will also allow users to plan their future viewing or reading, track their progress through pages or episodes, rate titles, and keep notes about works they wish to explore.

During the development of the service, modern technologies, libraries, and frameworks were used, such as React for application development, Tailwind CSS for creating the user interface, Node.js for backend development, FastAPI for building the API, and PostgreSQL for data storage.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕМРІНІВ

API – Application Programming Interface (комплект програмних інтерфейсів, структур даних та процедур, призначених для використання у розробці програмного забезпечення. API використовуються для забезпечення взаємодії між різними програмами та сервісами, дозволяючи їм обмінюватися інформацією та виконувати різноманітні завдання, тим самим сприяючи інтеграції та сумісності програмних продуктів)

CSS – Cascading Style Sheets (каскадні таблиці стилів) формальна мова декорування та опису зовнішнього вигляду документа (вебсторінки), написаного з використанням мови розмітки (найчастіше HTML або XHTML).

DOM – Document Object Model (програмний інтерфейс (API), який дає змогу розробникам отримувати доступ до структури та вмісту HTML- або XML-документів. DOM являє собою документ у вигляді ієрархічного дерева елементів, де кожен елемент є об’єктом)

HTTP – HyperText Transfer Protocol (протокол передачі гіпертекстових документів, протокол передачі даних, що використовується в комп’ютерних мережах)

ВСТУП

Сучасний інформаційний простір насичений величезною кількістю контенту, включаючи аніме та мангу, що набувають все більшої популярності серед користувачів по всьому світу. Однак зростання обсягу переглянутого контенту створює проблему його упорядкування та збереження інформації про вже переглянуті або заплановані до перегляду твори. Існуючі платформи, такі як MyAnimeList, AniList та Kitsu, надають базові можливості для каталогізації, проте не завжди відповідають індивідуальним запитам користувачів [1]. Саме тому розробка нового сервісу для каталогізації і управління переглянутим аніме та мангою є актуальною задачею, яка може запропонувати покращені можливості та персоналізований підхід.

Метою роботи є розробка вебзастосунку «Персональна бібліотека аніме та манги», який дозволить користувачам ефективно каталогізувати, відстежувати та управляти списком переглянутого аніме та прочитаної манги, а також взаємодіяти з контентом через сучасний, зручний та функціональний інтерфейс.

Об’єктом роботи є процеси каталогізації та управління інформацією про переглянуті аніме та мангу в онлайн-середовищі.

Предметом роботи є методи та засоби розробки вебзастосунку для ведення особистого каталогу аніме та манги, а також аналіз існуючих платформ та технологій, що використовуються для реалізації подібних сервісів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз основних методів для створення вебзастосунку «Персональна бібліотека аніме та манги»;

– дослідити перспективність розробки подібного застосунку;

– проаналізувати існуючі платформи для каталогізації та управління контентом (MyAnimeList, AniList, Kitsu);

– розглянути інструменти, що будуть використані для розробки (React, FastAPI, PostgreSQL, Tailwind CSS);

– сформулювати основні завдання проекту;

– спроєктувати систему, її компоненти та базу даних;

– визначити архітектуру та вимоги до системи;

– описати основні сторінки та компоненти інтерфейсу;

– розробити структуру бази даних;

– обґрунтувати вибір середовища розробки;

– реалізувати програмну частину вебзастосунку;

– розробити серверну та клієнтську частини системи;

– описати інструкцію користувача;

– розглянути можливі шляхи покращення застосунку.

Розроблений вебзастосунок надасть користувачам можливість зручно вести каталог переглянутого аніме та манги, персоналізувати свій досвід користування, а також отримати покращену функціональність порівняно з існуючими платформами. Отримані результати можуть бути використані як основа для подальшого вдосконалення подібних сервісів.

Таким чином, дана кваліфікаційна робота спрямована на створення ефективного інструменту для шанувальників аніме та манги, що дозволить їм легко керувати власною бібліотекою контенту.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ДЛЯ КАТАЛОГІЗАЦІЇ І УПРАВЛІННЯ ПЕРЕГЛЯНУТОГО АНІМЕ ТА МАНГИ

1.1 Аналіз перспективності створення вебзастосунку

Аніме та манга стали невід'ємною частиною світової поп-культури, а їхня популярність продовжує зростати. За останнє десятиліття ринок аніме розширився завдяки зростанню потокових сервісів, таких як Crunchyroll, Netflix, Funimation, які пропонують легальний доступ до контенту. Це спричинило збільшення кількості нових глядачів, які активно споживають контент і прагнуть його впорядкувати [3].

Користувачі стикаються з проблемою збереження інформації про переглянуті серії, прочитані розділи манги, обране та бажане до перегляду або прочитання. Багато хто створює списки вручну або використовує вже існуючі платформи, такі як MyAnimeList, AniList та Kitsu. Проте ці сервіси мають ряд обмежень:

Старий або незручний інтерфейс – зокрема, MyAnimeList має застарілий дизайн, що ускладнює навігацію.

Недостатня кастомізація – користувачі не можуть створювати власні категорії, додавати особисті нотатки чи редагувати статус перегляду з гнучкістю.

Обмежена інтеграція – відсутність синхронізації з деякими потоковими сервісами або соціальними платформами.

Відсутність розширеного аналізу контенту – немає інструментів для генерації рекомендацій на основі уподобань користувача [3].

Зважаючи на ці недоліки, розробка нового сервісу для каталогізації аніме та манги є актуальною, оскільки дозволить запропонувати користувачам сучасніші рішення, що відповідатимуть їхнім потребам.

Розвиток аніме-індустрії та цифрових технологій призвів до появи багатомільйонної аудиторії шанувальників, які регулярно використовують різні платформи для збереження інформації про переглянуті тай тли [2]. Про популярність подібних сервісів свідчать статистичні дані:

– MyAnimeList – понад 18 мільйонів зареєстрованих користувачів;

– AniList – понад 5 мільйонів активних користувачів;

– Kitsu – близько 500 тисяч користувачів.

Незважаючи на велику базу користувачів, значна частина фанатів аніме та манги шукає альтернативні рішення, що відповідають сучасним вимогам. Огляди на форумах, соціальних мережах та інших ресурсах часто містять критику на адресу існуючих платформ через обмежені можливості та складність використання.

Таким чином, розробка нового вебзастосунку, який поєднає зручний дизайн, гнучку систему управління контентом та розширені можливості інтеграції, має високу перспективність.

Основними користувачами вебзастосунку будуть:

– прихильники аніме та манги – люди, які активно переглядають або читають контент і хочуть його впорядкувати;

– нові глядачі аніме – користувачі, які тільки починають знайомитися з жанром і потребують зручного інструменту для ведення списку рекомендацій;

– фанати, які ведуть детальну статистику – ті, хто прагне не лише зберігати списки, а й аналізувати переглянуте, відстежувати час, витрачений на контент, та отримувати персоналізовані поради;

– блогери та рецензенти – користувачі, які створюють контент про аніме та мангу і бажають швидко отримувати інформацію про свої перегляди [4].

Таким чином, аудиторія застосунку є широкою, що підвищує потенційну популярність та затребуваність сервісу.

Аналіз тенденцій ринку демонструє перспективність створення вебзастосунку для каталогізації та управління переглянутим аніме та мангою.

Наявний попит серед користувачів та недоліки існуючих рішень вказують на необхідність нового сервісу з розширеними можливостями персоналізації, інтеграції та покращеного користувацького досвіду. Це забезпечить конкурентні переваги та зробить застосунок затребуваним серед шанувальників аніме та манги.

1.2 Огляд платформ для каталогізації і управління переглянутим контентом

Серед найбільш популярних вебсервісів для ведення каталогу переглянутого аніме та прочитаної манги можна виділити три ключові платформи: MyAnimeList, AniList та Kitsu. Вони надають користувачам можливість зберігати списки переглянутого контенту, ставити оцінки, залишати рецензії та взаємодіяти з іншими шанувальниками аніме та манги. Проте кожен з цих сервісів має свої особливості, переваги та недоліки.

1.2.1 MyAnimeList

MyAnimeList (MAL) – один із найстаріших і найпопулярніших сервісів для каталогізації аніме та манги, заснований у 2004 році (рис 1.1). Він має велику базу даних, яка містить детальну інформацію про тисячі тайтлів, включаючи жанри, студії, авторів, рейтинги та коментарі користувачів [5].

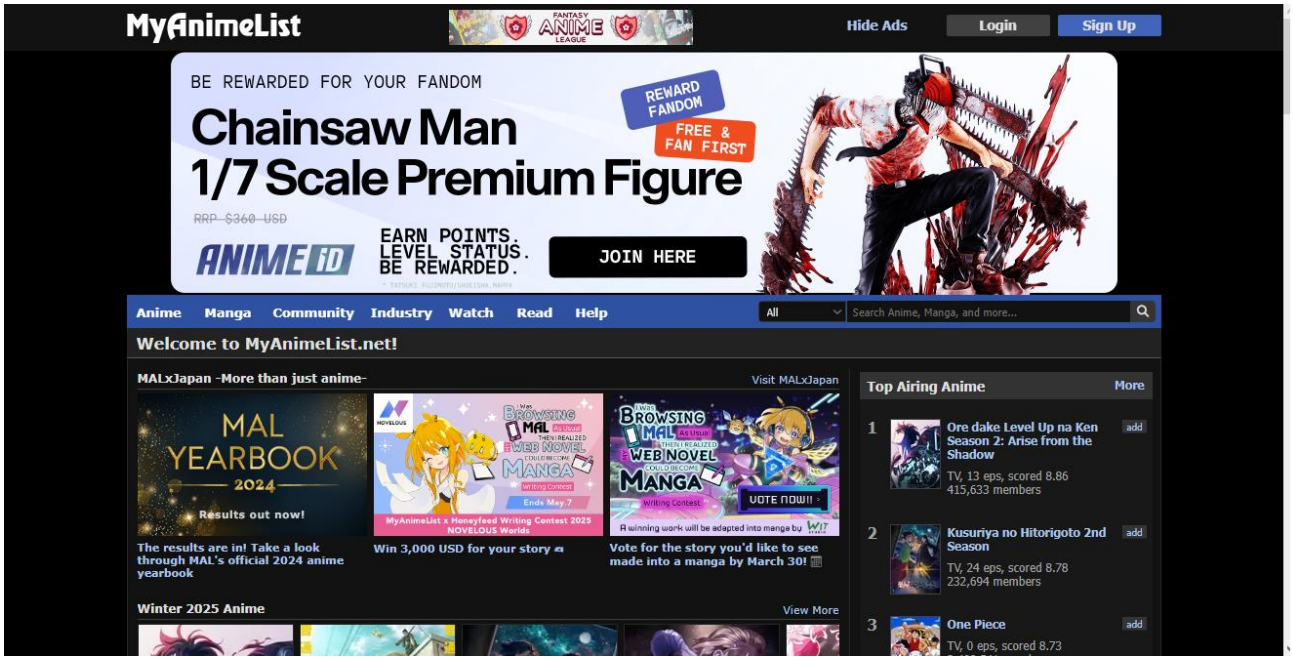

Рисунок 1.1 – Платформа MyAnimeList

Основні можливості:

– ведення персонального списку аніме та манги з розподілом за статусами (переглядаю, заплановано, переглянуто тощо);

– оцінювання контенту за 10-бальною шкалою;

– написання рецензій та коментарів;

– взаємодія з іншими користувачами через форуми та блоги;

– API для інтеграції з іншими сервісами;

– щоденні оновлення бази даних.

Переваги:

– велика база даних – містить майже всі офіційно випущені аніме та мангу;

– активна спільнота – користувачі регулярно обговорюють новинки та залишають рецензії;

– стабільна робота – сервіс існує більше 20 років і має високий рівень довіри.

Недоліки:

– застарілий інтерфейс – порівняно з сучасними платформами, дизайн виглядає незручним і потребує оновлення;

– обмежена кастомізація – мало можливостей для персоналізації списків та налаштування профілю;

– обмежений API – деякі функції недоступні для зовнішніх розробників через суворі обмеження API [6].

1.2.2 AniList

AniList – це сучасний вебсервіс для ведення списків аніме та манги, який пропонує розширені можливості для користувачів та розробників. Сайт став популярним завдяки мінімалістичному дизайну, широким можливостям кастомізації та відкритому API (рис 1.2) [7].

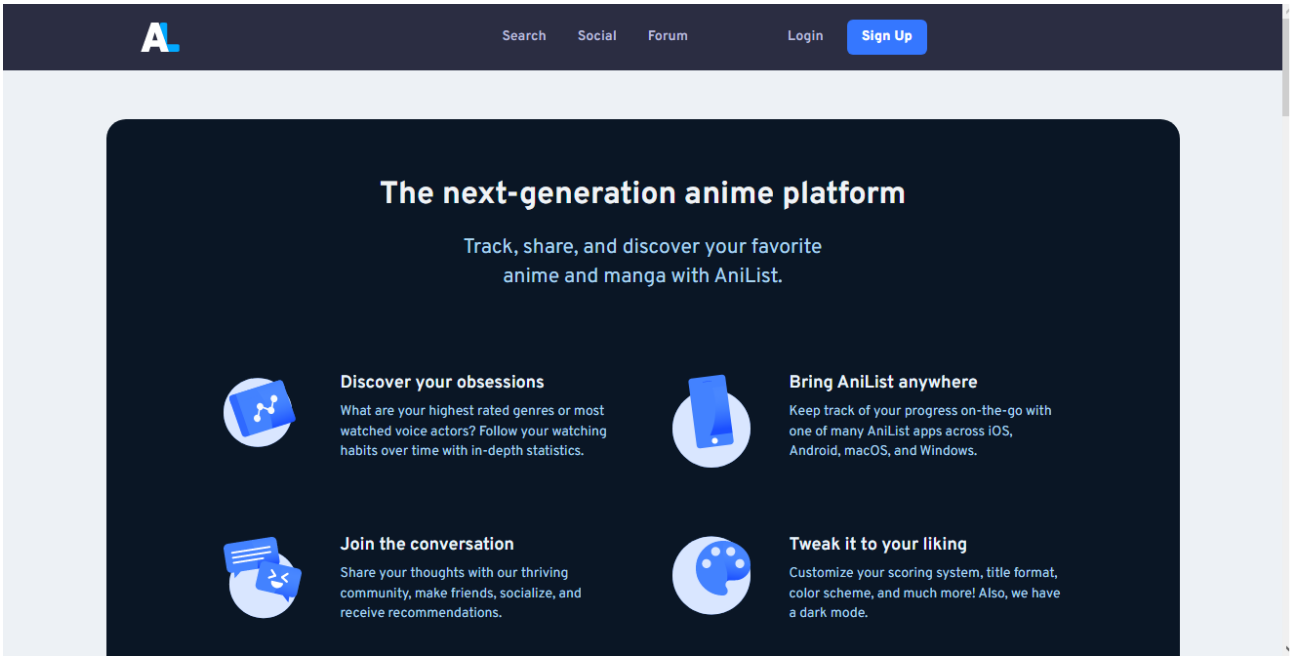

Рисунок 1.2 – Платформа AniList

Основні можливості:

– гнучка система персоналізації профілю та списків;

– візуально привабливий та зручний інтерфейс;

– глибока інтеграція з API, що дозволяє створювати сторонні додатки;

– підтримка як світлої, так і темної теми оформлення;

– можливість швидкого оновлення статусу перегляду через динамічні вкладки;

– рекомендації на основі вподобань користувача.

Переваги:

– сучасний дизайн – сайт має зручний та адаптивний інтерфейс;

– гнучкі налаштування профілю – користувачі можуть змінювати тему оформлення, налаштовувати списки, додавати власні категорії;

– відкритий API – дає можливість стороннім розробникам створювати інтеграції та боти для взаємодії з платформою;

– зручний пошук – AniList пропонує швидку фільтрацію тайтлів за жанрами, рейтингами, популярністю.

Недоліки:

– менша база користувачів – порівняно з MyAnimeList, AniList має значно меншу спільноту;

– менше рецензій та обговорень – активність на форумах та у відгуках нижча, ніж на MAL;

– система оцінювання – деяким користувачам не подобається 100-бальна система, яка може бути менш зручною за стандартну 10-бальну шкалу [8].

1.2.3 Kitsu

Kitsu – це ще одна платформа для каталогізації аніме та манги, яка націлена на соціальну взаємодію між користувачами. Вона відрізняється сучасним дизайном та функціями, що дозволяють користувачам швидко

знаходити новий контент на основі рекомендацій та взаємодіяти з іншими фанатами (рис 1.3) [9].

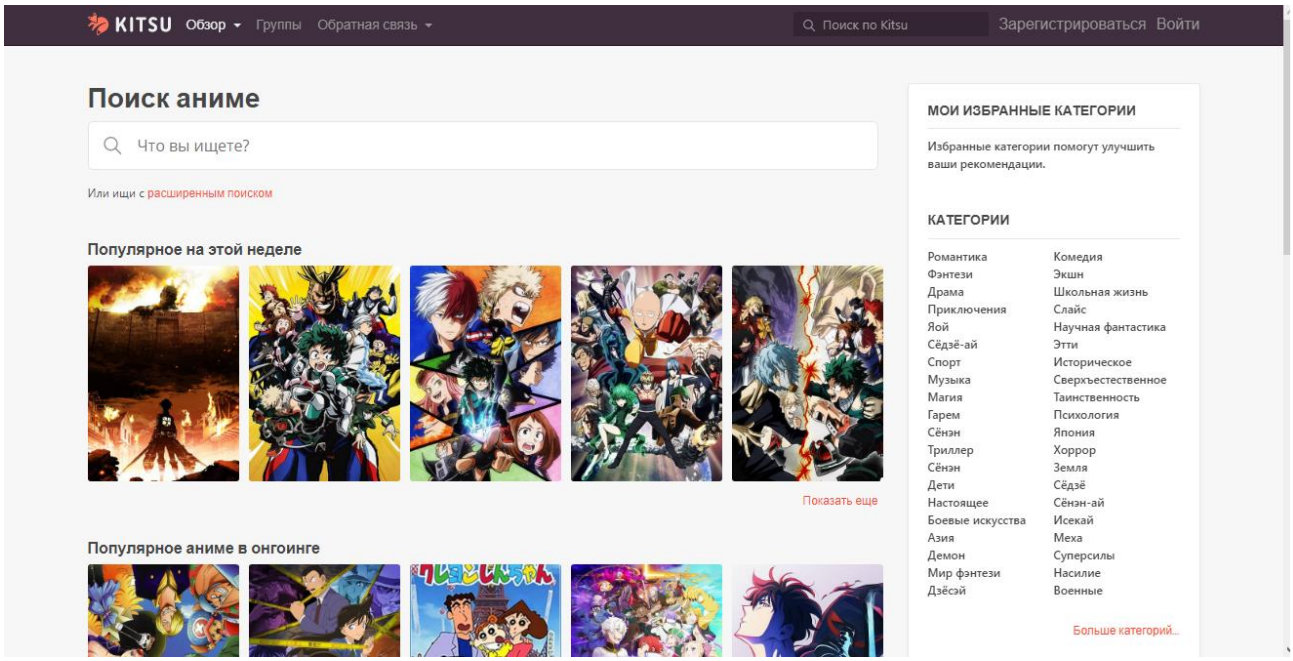

Рисунок 1.3 – Платформа Kitsu

Основні можливості:

– каталогізація переглянутого аніме та манги;

– інтерактивна стрічка новин з постами користувачів;

– рекомендації на основі вподобань;

– простий і швидкий інтерфейс;

– вбудовані обговорення та коментарі.

Переваги:

– фокус на соціальну взаємодію – можливість спілкуватися з іншими користувачами та створювати власні пости;

– сучасний дизайн – простий, мінімалістичний інтерфейс, який легко освоїти;

– автоматичне оновлення статусу перегляду – інтеграція з потоковими сервісами дозволяє автоматично оновлювати списки.

Недоліки:

– менша база даних – порівняно з MyAnimeList та AniList, Kitsu має обмежений список контенту;

– мала активність користувачів – через меншу популярність сервісу деякі функції можуть працювати повільніше, ніж на інших платформах;

– обмежена кастомізація – мало можливостей для налаштування профілю та списків [10].

Нижче приведена таблиця із порівнянням цих платформ (табл. 1.1).

Таблиця 1.1 – Таблиця порівняльного аналізу платформ

Характеристика	MyAnimeList [11]	AniList [12]	Kitsu [13]
Рік запуску	2004	2013	2014
Кількість користувачів	18 млн+	5 млн+	500 тис+
Інтерфейс	Застарілий	Сучасний	Мінімалістичний
Гнучкість налаштувань	Обмежена	Висока	Середня
API для розробників	Обмежений	Відкритий	Обмежений
Фокус платформи	Каталогізація	Каталогізація + кастомізація	Соціальна взаємодія
Інтерактивність	Форуми	Кастомізація списків	Стрічка новин
Рекомендації	Обмежені	Потужні	Середні

Усі три розглянуті платформи мають свої переваги та недоліки. MyAnimeList є найпопулярнішим сервісом із великою базою даних, але його інтерфейс застарілий. AniList пропонує сучасний дизайн і потужні можливості кастомізації, що робить його більш гнучким. Kitsu фокусується на соціальній взаємодії, однак має обмежену базу даних.

Ці особливості слід враховувати при розробці нового вебзастосунку, щоб створити конкурентоспроможний сервіс із сучасним дизайном, гнучкою кастомізацією та зручною каталогізацією контенту.

1.3 Огляд використовуваного інструментарію при створенні подібних застосунків

1.3.1 React

React – це популярна бібліотека для створення вебінтерфейсів, яка дозволяє створювати швидкі, динамічні та інтерактивні застосунки завдяки компонентному підходу та віртуальному DOM (табл. 1.2) [14].

Основні можливості React:

– компонентна архітектура – інтерфейс поділяється на незалежні компоненти;

– дносторінкові застосунки (SPA) – зміна контенту без перезавантаження сторінки;

– Virtual DOM – підвищує продуктивність за рахунок ефективного оновлення елементів;

– велика екосистема – багато бібліотек для управління станом (Redux, Zustand) і маршрутизації (React Router) [15].

Таблиця 1.2 – Таблиця переваг та недоліків React

Переваги	Недоліки
Висока продуктивність завдяки Virtual DOM	Високий поріг входу для новачків
Компонентний підхід для гнучкості коду	Потрібно керувати станом компонентів (може ускладнювати код)
Велика спільнота та підтримка	Вимагає сучасного стеку JavaScript (ES6+)
Масштабованість для великих застосунків	Інколи оновлення бібліотеки може спричиняти конфлікти з іншими інструментами [16]

1.3.2 FastAPI

FastAPI – це високопродуктивний вебфреймворк на основі Python, який забезпечує швидку розробку та ефективне обслуговування API (табл. 1.3) [17].

Основні можливості FastAPI:

– асинхронна обробка запитів (async/await), що покращує швидкодію;

– автоматична документація API (Swagger, ReDoc);

– вбудована валідація даних через Pydantic;

– підтримка JWT-аутентифікації та інтеграція з базами даних [18].

Таблиця 1.3 – Таблиця переваг та недоліків FastAPI

Переваги	Недоліки
Висока продуктивність (швидший за Flask та Django)	Менша популярність порівняно з Django
Простота у використанні	Потрібні знання асинхронного програмування
Генерація документації автоматично	Менша кількість готових бібліотек, ніж у Flask/Django
Підтримка асинхронних запитів	Деякі хостинги не підтримують FastAPI нативно [19]

1.3.3 PostgreSQL

PostgreSQL – це потужна реляційна база даних, яка підтримує складні запити та забезпечує високу продуктивність [20] (табл. 1.4).

Основні можливості PostgreSQL:

– підтримка ACID-транзакцій – забезпечує надійність збереження даних;

– JSONB-типи – гнучка робота з напівструктурованими даними;

– індексація та оптимізація запитів – покращує швидкість вибірки;

– розширюваність – можливість створювати власні функції, процедури [21].

Таблиця 1.4 – Таблиця Переваг та недоліків PostgreSQL

Переваги	Недоліки
Висока стабільність та безпека	Вимагає більше ресурсів, ніж MySQL
Потужні інструменти для аналітики та індексації	Складність налаштування для новачків
Підтримка JSONB для гібридних структур даних	Повільніший за NoSQL-бази для неструктурованих даних
Відкритий код, безкоштовний	Потрібно більше оптимізації для масштабних проєктів [22]

1.3.4 Tailwind CSS

Tailwind CSS – це утилітарний CSS-фреймворк, який дозволяє швидко створювати сучасні та адаптивні інтерфейси [23] (табл. 1.5).

Основні можливості Tailwind CSS:

– використання готових CSS-класів – зменшує потребу в кастомних стилях;

– гнучкість – можливість створення унікального дизайну без написання CSS-файлів;

– адаптивний дизайн – зручна підтримка медіа-запитів;

– швидка розробка – мінімізує час на стилізацію елементів [24].

Таблиця 1.5 – Таблиця переваг та недоліків Tailwind CSS

Переваги	Недоліки
Швидка стилізація без необхідності написання окремих CSS-файлів	Незвичний синтаксис для тих, хто звик до класичного CSS
Висока кастомізація та легка адаптація	Може ускладнювати HTML-код через велику кількість класів
Мінімізація фінального CSS-коду	Вимагає початкового налаштування (конфігураційний файл)
Добре підходить для SPA та React-застосунків	Відсутність готових компонентів, як у Bootstrap [25]

Обраний стек технологій (React, FastAPI, PostgreSQL, Tailwind CSS) забезпечує оптимальне поєднання продуктивності, гнучкості та зручності у

розробці вебзастосунку «Персональна бібліотека аніме та манги». Використання React дозволяє створити динамічний та інтерактивний інтерфейс, FastAPI забезпечує швидку та ефективну серверну частину, PostgreSQL гарантує стабільне управління даними, а Tailwind CSS сприяє швидкій та зручній стилізації. Завдяки цьому стеку застосунок буде швидким, масштабованим і зручним для користувачів.

1.4 Постановка задачі

У сучасному цифровому середовищі існує значний попит на сервіси, які допомагають користувачам каталогізувати та впорядковувати власні медіаколекції. Особливо актуальним це є для шанувальників аніме та манги, оскільки обсяг контенту постійно зростає, а користувачам потрібно зручно відстежувати переглянуті епізоди, прочитані розділи та зберігати списки для майбутнього перегляду.

Основна мета роботи – розробка вебзастосунку для каталогізації та управління переглянутим аніме та мангою, який надасть користувачам зручний інструмент для ведення персональної бібліотеки медіаконтенту. Для досягнення цієї мети необхідно виконати такі завдання:

– спроєктувати архітектуру системи з визначенням основних компонентів, бази даних, API-ендпоінтів та взаємодії між клієнтом і сервером;

– розробити функціонал застосунку, що включає можливість реєстрації користувачів, додавання аніме та манги до особистої бібліотеки, відстеження переглянутих серій та розділів, створення списків для майбутнього перегляду;

– забезпечити зручний та інтуїтивний інтерфейс, який дозволить користувачам легко взаємодіяти із застосунком;

– реалізувати механізм збереження та оновлення даних у базі даних із можливістю швидкого пошуку та фільтрації контенту;

– протестувати та оптимізувати роботу вебзастосунку, включаючи перевірку продуктивності, безпеки та коректного відображення інтерфейсу на різних пристроях.

У результаті виконання поставлених завдань буде створено функціональний вебзастосунок, що надасть користувачам зручний інструмент для ведення власної бібліотеки аніме та манги, забезпечуючи ефективне управління контентом, швидкий доступ до збережених записів та комфортну взаємодію з системою.

2 ПРОЄКТУВАННЯ СИСТЕМИ ТА ОПИС ОСНОВНИХ КОМПОНЕНТІВ

2.1 Проєктування системи

Проєктування системи є важливим етапом розробки програмного забезпечення, який передбачає створення чіткого плану реалізації проєкту. Цей процес включає визначення структури системи, вибір технологій, моделювання її компонентів і планування етапів створення. У рамках цього розділу описано основні кроки, необхідні для розробки вебсервісу, призначеного для каталогізації та управління переглянутими аніме та мангою, а також підходи до організації роботи над проєктом.

Система розробляється як вебсервіс, що дозволяє користувачам вести облік переглянутих аніме та манги, відстежувати прогрес і обмінюватися інформацією з іншими користувачами. Сервіс буде доступний через веббраузер і адаптований для використання на різних пристроях, таких як комп’ютери, планшети та смартфони. Основна мета полягає в забезпеченні зручного інструменту для любителів аніме та манги, який поєднує простоту використання з можливістю розширення функціональності в майбутньому.

Система базується на клієнт-серверній архітектурі. Клієнтська частина відповідатиме за відображення інтерфейсу користувача та взаємодію з ним, тоді як серверна частина оброблятиме запити, зберігатиме дані та забезпечуватиме логіку роботи сервісу. Для обміну даними між клієнтом і сервером буде використано RESTful API з можливістю подальшого переходу на GraphQL. Інтеграція із зовнішніми джерелами, такими як доступні API (наприклад, MyAnimeList), дозволить автоматично отримувати інформацію про твори.

Розробка системи поділена на кілька послідовних етапів, кожен із яких має свої завдання та результати:

На першому етапі, який називається аналізом і плануванням, визначаються основні цілі проєкту, досліджуються потреби цільової аудиторії та складається детальний план робіт. Цей етап завершується створенням технічного завдання, яке слугуватиме основою для подальшої розробки.

Другий етап присвячений проєктуванню інтерфейсу користувача. На цьому кроці створюються прототипи сторінок сервісу, включаючи головну сторінку, сторінку списку користувача та форми додавання творів. Прототипи розробляються з урахуванням зручності навігації та адаптивності для різних пристроїв. Результатом етапу є затверджений дизайн інтерфейсу.

Третій етап охоплює розробку клієнтської частини. На цьому етапі реалізується інтерфейс користувача з використанням сучасних технологій для створення динамічних і швидких сторінок. Основна увага приділяється інтерактивності та коректному відображенню даних, отриманих із серверної частини.

Четвертий етап передбачає розробку серверної частини. Тут створюється логіка обробки запитів, інтеграція із зовнішніми API та механізми зберігання й обробки даних користувачів і творів. Цей етап завершується налаштуванням стабільної взаємодії між клієнтом і сервером.

П’ятий етап – це тестування системи. На цьому кроці проводяться перевірки працездатності всіх компонентів, усуваються виявлені помилки та оптимізується продуктивність. Тестування включає як ручну перевірку, так і автоматизовані тести для забезпечення стабільності роботи сервісу.

Шостий і завершальний етап – розгортання та підтримка. Після успішного тестування система розгортається на хостинг-платформі, стає доступною для користувачів і переходить у режим підтримки. На цьому етапі також планується збір відгуків від користувачів для подальшого вдосконалення.

2.2 Визначення вимог

Визначення вимог є першим етапом розробки програмного забезпечення, спрямованим на з’ясування потреб користувачів, цілей проєкту та обмежень, які необхідно врахувати під час реалізації. Чітке формулювання вимог дозволяє забезпечити відповідність кінцевого продукту очікуванням користувачів і технічним можливостям.

Метою проєкту є розробка вебсервісу, який надасть користувачам зручний інструмент для каталогізації та управління переглянутими аніме та мангою. Сервіс має допомагати користувачам систематизувати інформацію про переглянуті твори, відстежувати прогрес, оцінювати контент і отримувати персоналізовані рекомендації. Проєкт спрямований на полегшення організації хобі, пов’язаного з аніме та мангою, та створення спільноти для обміну досвідом.

Цільовою аудиторією сервісу є любителі аніме та манги віком від 14 до 35 років, які активно переглядають або читають твори цього жанру, а також користувачі, які бажають вести облік переглянутого контенту, планувати подальші перегляди, отримувати рекомендації на основі власних уподобань і ділитися своїми списками та оцінками з іншими учасниками спільноти.

Сервіс призначений для використання через веббраузери на настільних комп’ютерах, ноутбуках, планшетах і смартфонах. Користувачі матимуть доступ до системи в будь-який час за наявності підключення до Інтернету. Передбачається, що основними сценаріями використання будуть додавання нового аніме чи манги до особистого списку після перегляду, оновлення статусу перегляду (наприклад, зміна з «планую» на «переглянуто»), пошук творів за ключовими характеристиками для планування переглядів і перегляд рекомендацій у вільний час для вибору нового контенту.

Розглянемо функціональні вимоги.

Реєстрація та автентифікація:

– можливість створення облікового запису за допомо електронної пошти та пароля;

– вхід у систему з використанням облікових даних;

– відновлення пароля через електронну пошту.

Каталогізація контенту:

– додавання аніме або манги до персонального списку;

– вказівка статусу (переглянуто, в процесі, планується), відгуку та прогресу (кількість переглянутих епізодів/глав);

– редагування або видалення записів у списку.

Пошук і фільтрація:

– пошук творів за назвою, жанром, роком випуску чи типом (аніме/манга);

– фільтрація списку за статусом, оцінкою або іншими параметрами.

Нефункціональні вимоги:

– продуктивність – система повинна забезпечувати швидкий відгук (менше 2 секунд) при обробці запитів до 1000 одночасних користувачів;

– доступність – сервіс має бути доступним 99% часу, за винятком періодів технічного обслуговування;

– інтерфейс користувача – інтуїтивно зрозумілий дизайн із підтримкою адаптивності для різних пристроїв і можливістю вибору темного або світлого режиму;

– безпека – захист персональних даних користувачів через шифрування передачі даних (HTTPS) і безпечне зберігання паролів;

– масштабованість – можливість розширення функціональності (наприклад, додавання нових типів контенту) без значних змін у структурі системи;

2.3 Опис сторінок та компонентів системи

Розроблений вебсервіс для каталогізації та управління переглянутими аніме та мангою складається з набору сторінок і компонентів, які забезпечують зручну взаємодію користувача з системою. Кожна сторінка має визначений набір можливостей, спрямованих на виконання основних завдань користувача, таких як додавання творів, управління списками, пошук контенту та перегляд рекомендацій. Компоненти системи доповнюють сторінки, надаючи модульні елементи для відображення даних і виконання дій. Нижче наведено текстовий опис можливостей сторінок і компонентів системи.

Головна сторінка вітає користувача та пропонує швидкий доступ до основних функцій сервісу. На ній відображається короткий огляд популярних аніме та манги, а також рекомендації, якщо користувач увійшов у систему. Для незареєстрованих користувачів доступна кнопка, яка перенаправляє на сторінку реєстрації або входу. Крім того, головна сторінка містить панель навігації, через яку можна перейти до інших розділів сервісу, таких як особистий список чи пошук.

Сторінка реєстрації дозволяє новим користувачам створити обліковий запис. Вона містить форму, де необхідно вказати електронну пошту, ім’я користувача та пароль. Після заповнення форми користувач отримує підтвердження про успішну реєстрацію та перенаправляється на сторінку входу. Сторінка входу пропонує форму для введення електронної пошти та пароля, а також посилання для відновлення пароля у разі його втрати.

Сторінка особистого списку є центральною для роботи з каталогізацією. На ній відображаються всі аніме та манги, додані користувачем, із зазначенням їхнього статусу, оцінки та прогресу перегляду. Користувач може сортувати список за різними параметрами, наприклад, за назвою чи датою додавання, а також фільтрувати твори за типом чи статусом. На цій сторінці доступна кнопка для додавання нового твору, яка відкриває форму з полями для введення назви, типу, жанру, кількості епізодів або глав, статусу, оцінки

та поточного прогресу. Кожен запис у списку можна редагувати або видалити за допомогою відповідних кнопок поруч із ним.

Сторінка пошуку призначена для знаходження аніме та манги. Користувач може ввести назву або ключові слова в рядок пошуку, а також скористатися додатковими фільтрами, такими як жанр, рік випуску чи тип контенту. Результати пошуку відображаються у вигляді карток із назвою твору, коротким описом і обкладинкою. Натиснувши на картку, користувач переходить на сторінку детальної інформації про твір, де можна додати його до свого списку або залишити коментар.

Сторінка профілю дозволяє користувачу переглядати та редагувати особисту інформацію, таку як ім’я, електронна пошта чи налаштування сповіщень. На цій сторінці також доступна можливість змінити пароль і переглянути статистику, наприклад, кількість доданих творів або середню оцінку. Крім того, користувач може налаштувати приватність свого списку, визначивши, чи можуть інші користувачі його переглядати.

Компонент картки твору використовується на сторінках пошуку, рекомендацій і списку. Він відображає назву твору, обкладинку, тип, жанр і короткий опис. При натисканні на картку відкривається детальна інформація про твір. Компонент форми додавання твору застосовується на сторінці особистого списку та дозволяє вводити дані про твір, включаючи назву, тип, статус і прогрес.

Компонент навігаційної панелі присутній на всіх сторінках і пропонує швидкий доступ до головної сторінки, списку, пошуку, рекомендацій і профілю.

Компонент фільтрації, який використовується на сторінках пошуку та списку, дає змогу обирати параметри для сортування чи відбору творів, таких як жанр чи статус.

2.4 Огляд та проєктування бази даних

База даних є ключовим елементом вебсервісу для каталогізації та управління переглянутими аніме та мангою, забезпечуючи структуроване зберігання даних і швидкий доступ до них. Проєктування бази даних спрямоване на створення ефективної структури, яка підтримує основні потреби користувачів, зберігає цілісність інформації та дозволяє легко розширювати систему в майбутньому. У цьому розділі обґрунтовано вибір бази даних, описано її структуру, таблиці з полями та зв’язки між ними.

Для реалізації бази даних обрано PostgreSQL – реляційну систему управління базами даних із відкритим вихідним кодом. Вибір PostgreSQL пояснюється кількома причинами. По-перше, вона забезпечує високу продуктивність завдяки підтримці індексів і оптимізації складних запитів, що важливо для швидкого пошуку творів чи списків при зростанні кількості користувачів. По-друге, PostgreSQL підтримує різноманітні типи даних, такі як ENUM для статусів творів чи JSON для додаткових метаданих, що відповідає потребам проєкту. По-третє, її можливості транзакцій і забезпечення цілісності даних ідеально підходять для одночасної роботи багатьох користувачів, які оновлюють свої списки. Крім того, PostgreSQL є безкоштовною, має активну спільноту підтримки та добре інтегрується з популярними технологіями розробки, що робить її оптимальним вибором для кваліфікаційного проєкту [26].

Структура бази даних складається з двох основних таблиць: «Users» (Користувачі), «Items» (Твори). Ці таблиці дозволяють зберігати інформацію про користувачів, аніме та мангу, а також відстежувати статуси («В планах», «Читаю/Дивлюсь», «Прочитано/Переглянуто») і прогрес перегляду чи читання творів. Зв’язки між таблицями реалізовані через зовнішні ключі, що забезпечує реляційну модель даних. Нижче наведено детальний опис таблиць із розширеними поясненнями їхньої ролі та полів (табл 2.1).

Ця таблиця призначена для зберігання персональних даних зареєстрованих користувачів сервісу, які є основними суб’єктами системи.

Вона забезпечує унікальну ідентифікацію кожного користувача, автентифікацію та базову інформацію про їхній обліковий запис. Завдяки цій таблиці система може відстежувати, хто саме додає твори до списків і взаємодіє з сервісом.

Таблиця 2.1 – Таблиця моделі «Users» (Користувачі)

Поле	Тип даних	Опис	Обмеження
id	INTEGER	Унікальний ідентифікатор користувача, який автоматично генерується при реєстрації. Використовується як первинний ключ для зв’язку з іншими таблицями.	PRIMARY KEY
username	TEXT	Ім’я користувача, яке відображається в інтерфейсі та використовується для соціальної взаємодії (наприклад, у коментарях чи списках). Унікальність запобігає дублюванню імен.	UNIQUE, NOT NULL
password	TEXT	Хеш пароля, який зберігається для безпечної автентифікації. Використання хешування замість відкритого тексту підвищує безпеку даних. Поле завжди заповнене під час реєстрації.	NOT NULL
email	TEXT	Електронна пошта користувача, що слугує основним способом входу в систему та зв’язку. Обмеження на унікальність гарантує, що один email не може бути використаний кількома користувачами.	UNIQUE, NOT NULL
created_at	DATETIME	Дата і час створення облікового запису, що автоматично заповнюється поточною датою. Дозволяє аналізувати активність користувачів і стаж їхньої участі в сервісі.	DEFAULT CURRENT_TIMESTAMP

Поле password зберігає хеш пароля, згенерований за допомогою bcrypt, для безпечної автентифікації. Поле created_at автоматично заповнюється поточною датою та часом під час створення запису.

Таблиця використовується для автентифікації через JWT-токени (згідно з items.py, де є OAuth2PasswordBearer) (табл. 2.2).

Ця таблиця містить інформацію про аніме та мангу, які є основним контентом сервісу. Вона зберігає метадані творів, які користувачі можуть додавати до своїх списків, і слугує як централізоване джерело даних про доступний контент.

Таблиця 2.2 – Таблиця моделі «Items» (Твори)

Поле	Тип даних	Опис	Обмеження
1	2	3	4
id	INTEGER	Унікальний ідентифікатор твору, який автоматично генерується при додаванні нового аніме чи манги. Використовується як первинний ключ для зв'язку з таблицею 31ористувачів.	PRIMARY KEY
user_id	INTEGER	Ідентифікатор користувача, який додав твір	NOT NULL, FOREIGN KEY (users.id)
title	TEXT	Назва твору, що є основним ідентифікатором для користувачів. Обов'язкове поле, яке відображається в інтерфейсі та використовується для пошуку.	NOT NULL
type	TEXT	Тип твору, що визначає, чи є це аніме чи манга. Використання ENUM із значеннями «anime» і «manga» спрощує класифікацію та фільтрацію. Поле завжди заповнене.	NOT NULL
status	TEXT	Статус твору, який може включати кілька значень (Хочу переглянути/прочитати, Дивлюсь/Читаю, Переглянуто/Прочитано)	NOT NULL
episode	INTEGER	Загальна кількість епізодів (для аніме) або глав (для манги), що визначає прогрес користувача.	NULLABLE

Продовження таблиці 2.2

1	2	3	4
created_at	DATETIME	Дата і час створення облікового запису, що автоматично заповнюється поточною датою. Дозволяє аналізувати активність користувачів і стаж їхньої участі в сервісі.	DEFAULT CURRENT_TIMESTAMP
image_url	TEXT	Посилання на обкладинку твору	
score	DOUBLE PRECISION	Оцінка твору	
genre	TEXT	Жанр твору	
review	TEXT	Відгук користувача про твір	NULLABLE

Поле user_id є зовнішнім ключем, що пов’язує твір із користувачем у таблиці users.

Поле type обмежене значеннями «anime» або «manga», що відповідає вибору в формі додавання (Dashboard.jsx).

Поле status може мати значення «want», «watching», «completed», що відповідає вкладкам у каталозі.

Поле episode заповнюється для творів зі статусом «watching» (наприклад, 50 для «Ван Піс – Епізод 50») і є NULL для інших статусів.

Поле review дозволяє користувачам додавати текстові відгуки, але зазвичай порожнє (з HTML і Dashboard.jsx видно, що відгуки не додаються).

Поле created_at автоматично встановлюється під час створення запису.

Поле image_url дає можливість додати посилання на обклдинку твору.

Поле score дає можливість оцінити твір.

Поле genre дозволяє вказати жанр твору.

Структура зв’язків:

– зв’язок один-до-багатьох – один користувач (users.id) може мати багато творів (items.user_id);

– зовнішній ключ items.user_id посилається на users.id, забезпечуючи цілісність даних (твір не може існувати без користувача).

Проєкт бази даних у PostgreSQL забезпечує надійне зберігання даних про користувачів і твори, підтримку трьох статусів («В планах», «Читаю/Дивлюсь», «Прочитано/Переглянуто») і відстеження прогресу перегляду чи читання.

2.5 Обґрунтування вибору середовища розробки

Вибір середовища розробки є важливим етапом створення вебсервісу для каталогізації та управління переглянутими аніме та мангою, оскільки від нього залежить зручність написання коду, налагодження, тестування та інтеграція всіх компонентів системи.

Для розробки вебсервісу обрано інтегроване середовище розробки (IDE) Visual Studio Code (VS Code). Ця програма є легким, але потужним редактором коду від Microsoft, який підтримує широкий спектр мов програмування та технологій, що робить її ідеальною для створення сучасних вебдодатків. VS Code вирізняється своєю гнучкістю, можливістю налаштування та великою екосистемою розширень, що дозволяє адаптувати її під конкретні потреби проєкту, включаючи роботу з клієнтською та серверною частинами, а також базою даних [27].

Особливості VS Code включають вбудовану підтримку JavaScript, TypeScript, HTML і CSS, які є основними технологіями для розробки вебінтерфейсу сервісу з використанням фреймворку React.js. Програма також добре працює з Node.js, що є основою серверної частини проєкту, побудованого на фреймворку Express. Середовище має інтегрований термінал, який дозволяє запускати команди, такі як встановлення залежностей чи запуск серверу, безпосередньо в редакторі. Крім того, VS Code підтримує систему контролю версій Git, що спрощує управління кодом і співпрацю в команді, якщо проєкт розширюватиметься. Велика бібліотека розширень, таких як Prettier для форматування коду, ESLint для аналізу якості JavaScript або

PostgreSQL для роботи з базою даних, дозволяє налаштувати середовище для всіх аспектів розробки. Ще однією особливістю є підтримка дебагінгу, що дає змогу швидко знаходити й усувати помилки в коді як на клієнтській, так і на серверній стороні [28].

Переваги VS Code:

– безкоштовність, що ідеально підходить для проєкту з обмеженим бюджетом, як-от кваліфікаційна робота;

– легкість і швидкий запуск навіть на середніх за потужністю комп'ютерах, що забезпечує комфортну роботу без затримок;

– гнучкість завдяки розширенням, які дозволяють адаптувати редактор до потреб проєкту, наприклад, додавши підтримку Docker чи інструменти для роботи з REST API;

– висока популярність серед розробників, що забезпечує доступ до великої кількості документації, навчальних матеріалів і активної спільноти для вирішення проблем;

– інтуїтивно зрозумілий інтерфейс із підтримкою автодоповнення коду (IntelliSense), що прискорює написання якісного коду для React.js і Node.js;

– кросплатформність, що дозволяє використовувати програму на Windows, macOS чи Linux без втрати функціональності [29].

Можливість роботи з Git безпосередньо з редактора, що спрощує керування версіями коду та командну роботу над проєктом. Інтеграція з терміналом дозволяє виконувати команди без переходу між вікнами, що підвищує продуктивність розробника. Постійні оновлення та підтримка з боку Microsoft забезпечують стабільність і впровадження нових функцій відповідно до сучасних вимог розробки. Великий вибір тем оформлення і можливість персоналізації інтерфейсу сприяють створенню комфортного середовища для тривалої роботи. Підтримка відлагодження (debugging) з можливістю встановлення брейкпоінтів, перегляду змінних і виконання кроків коду без потреби у зовнішніх інструментах.

Недоліки VS Code:

– відсутність повноцінних функцій IDE, таких як складний рефакторинг чи вбудоване управління проєктами, порівняно з важкими середовищами, наприклад, IntelliJ IDEA чи WebStorm;

– обмежена підтримка деяких мов програмування в порівнянні з профільними IDE, що може впливати на якість автодоповнення або підсвічування синтаксису;

– необхідність початкового налаштування, адже для повноцінної роботи потрібно встановити й налаштувати розширення, що може бути незручним для новачків;

– можливе збільшення споживання оперативної пам’яті при великій кількості відкритих вкладок або активних розширень, що може вплинути на продуктивність на слабких комп’ютерах [30];

– відсутність єдиної екосистеми – кожен проєкт може потребувати окремого набору розширень, що призводить до несумісності конфігурацій між командами. Періодичні проблеми з оновленнями розширень, які можуть спричиняти конфлікти або порушення стабільності редактора.

Visual Studio Code обрано як основне середовище розробки для створення вебсервісу через його легкість, гнучкість і потужну підтримку технологій, необхідних для проєкту, таких як JavaScript, React.js, Node.js і PostgreSQL. Безкоштовність і велика екосистема розширень дозволяють адаптувати програму до всіх етапів розробки – від написання коду до тестування й інтеграції. Хоча VS Code потребує початкового налаштування й не має деяких функцій повноцінних IDE, його переваги значно переважають недоліки в контексті кваліфікаційної роботи. Ця програма забезпечує оптимальний баланс між простотою, продуктивністю та функціональністю, що робить її найкращим вибором для реалізації сервісу для каталогізації та управління переглянутими аніме та мангою.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ «ПЕРСОНАЛЬНА БІБЛІОТЕКА АНИМЕ ТА МАНГИ»

3.1 Програмна реалізація

У рамках дипломної роботи був розроблений вебсервіс «Аніме та манга» для каталогізації і управління переглянутими аніме та мангою. Для реалізації було обрано такі інструменти: React, FastAPI, PostgreSQL, Tailwind CSS.

3.1.1 Створення серверної частини

Серверна частина вебсервісу для каталогізації та управління переглянутими аніме та мангою є основою для обробки запитів, зберігання даних і забезпечення логіки роботи системи. Вона розроблена з використанням фреймворку FastAPI на мові Python, який відповідає за створення API, взаємодію з базою даних PostgreSQL і обробку запитів від клієнтської частини, побудованої на React із застосуванням Tailwind CSS для стилізації. У цьому розділі описано процес підключення до бази даних, механізми авторизації та реєстрації користувачів, а також інші ключові компоненти серверної частини, такі як управління списками творів і обробка запитів. Для кожного компонента наведено приклади коду, що демонструють їхню реалізацію.

Для взаємодії з базою даних PostgreSQL використано бібліотеку SQLAlchemy як ORM (Object-Relational Mapping), що дозволяє працювати з даними через об’єкти Python, а також бібліотеку psycopg2 для прямого підключення. FastAPI підключається до бази через конфігураційний файл, де задано параметри з’єднання, такі як ім’я бази, користувач, пароль і хост. У коді створюється сесія для роботи з базою, яка відкривається для кожного запиту і

закривається після його виконання, що забезпечує безпечну та ефективну взаємодію.

Головна сторінка пропонує інтуїтивний доступ до основних можливостей сервісу. Користувачі можуть переглянути добірку популярних аніме та манги, представлену у вигляді карток із зображеннями та назвою. Авторизованим користувачам, які увійшли за допомогою безпечного JWT-токена доступна функція додавання твору до свого списку. Для нових відвідувачів передбачені зручні кнопки для реєстрації або входу, що забезпечують швидке створення акаунта з безпечним хешуванням пароля (bcrypt). Панель навігації у верхній частині сторінки дозволяє легко перейти до особистих списків, пошуку чи профілю, забезпечуючи комфортну взаємодію з сервісом.

Лістинг 3.1 Реалізація функції каталогізації і управління списками творів:

```
from fastapi import APIRouter, Depends, HTTPException
from sqlalchemy.orm import Session
from typing import List
from pydantic import BaseModel
from datetime import datetime
from backend.database import SessionLocal
from backend.models import Catalog, User
from backend.auth import get_current_user

router = APIRouter()

# Pydantic моделі
class CatalogBase(BaseModel):
    title: str
```

```
    type: str
    image_url: str | None = None

class CatalogResponse(CatalogBase):
    id: int
    created_at: datetime

    class Config:
        from_attributes = True

# Залежність для отримання сесії бази даних
def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

# Отримання списку всіх творів
@router.get("/catalog", response_model=List[CatalogResponse])
def get_catalog(db: Session = Depends(get_db)):
    return db.query(Catalog).all()

# Додавання нового твору (тільки для суперадміну)
@router.post("/catalog", response_model=CatalogResponse)
def create_catalog(
    item: CatalogBase,
    db: Session = Depends(get_db),
    current_user: User = Depends(get_current_user)
```

```
):
    if not current_user.is_superadmin:
        raise HTTPException(status_code=403, detail="Тільки суперадмін може додавати твори")
    db_item = Catalog(**item.dict(), created_at=datetime.utcnow())
    db.add(db_item)
    db.commit()
    db.refresh(db_item)
    return db_item
```

Реєстрація користувача передбачає створення нового запису в таблиці Users із хешуванням пароля для безпеки. Для цього використано бібліотеку passlib із алгоритмом bcrypt.

Лістинг 3.2 Реалізація функції реєстрації нового користувача:

```
from fastapi import APIRouter, Depends, HTTPException, status
from fastapi.security import OAuth2PasswordRequestForm, OAuth2PasswordBearer
from sqlalchemy.orm import Session
from backend.database import SessionLocal
from backend.models import User
from datetime import timedelta, datetime
from jose import jwt, JWTError
from passlib.context import CryptContext
from pydantic import BaseModel

router = APIRouter()

SECRET_KEY = "your-secret-key-1234567890
```

```
ALGORITHM = "HS256"
ACCESS_TOKEN_EXPIRE_MINUTES = 30

pwd_context = CryptContext(schemes=["bcrypt"], deprecated="auto")
oauth2_scheme = OAuth2PasswordBearer(tokenUrl="token")

class UserCreate(BaseModel):
    username: str
    email: str
    password: str

def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

def verify_password(plain_password, hashed_password):
    return pwd_context.verify(plain_password, hashed_password)

def create_access_token(data: dict):
    to_encode = data.copy()
    expire = datetime.utcnow() + timedelta(minutes=ACCESS_TOKEN_EXPIRE_MINUTES)
    to_encode.update({"exp": expire})
    encoded_jwt = jwt.encode(to_encode, SECRET_KEY, algorithm=ALGORITHM)
    return encoded_jwt
```

```
def hash_password(password: str):
    return pwd_context.hash(password)

async def get_current_user(token: str = Depends(oauth2_scheme), db: Session = Depends(get_db)):
    credentials_exception = HTTPException(
        status_code=status.HTTP_401_UNAUTHORIZED,
        detail="Не вдалося перевірити облікові дані",
        headers={"WWW-Authenticate": "Bearer"},
    )
    try:
        payload = jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])
        username: str = payload.get("sub")
        if username is None:
            raise credentials_exception
    except JWTError:
        raise credentials_exception
    user = db.query(User).filter(User.username == username).first()
    if user is None:
        raise credentials_exception
    return user
@router.post("/register")
async def register(user: UserCreate, db: Session = Depends(get_db)):
    existing_user = db.query(User).filter(User.username == user.username).first()
    if existing_user:
        raise HTTPException(
            status_code=status.HTTP_400_BAD_REQUEST,
            detail="Ім'я користувача вже зареєстровано"
        )
```

```
existing_email = db.query(User).filter(User.email == user.email).first()
if existing_email:
    raise HTTPException(
        status_code=status.HTTP_400_BAD_REQUEST,
        detail="Email вже зареєстровано"
    )
hashed_password = hash_password(user.password)
new_user = User(
    username=user.username,
    email=user.email,
    password=hashed_password,
    created_at=datetime.utcnow()
)
db.add(new_user)
db.commit()
db.refresh(new_user)
return {"message": "Користувача успішно зареєстровано"}
```

Авторизація реалізується через видачу JSON Web Token (JWT), який перевіряється при кожному захищеному запиті. FastAPI використовує залежності для валідації токенів.

Тут /register створює нового користувача з хешованим паролем, а /login перевіряє дані та видає JWT-токен. Функція get_current_user використовується як залежність для захищених ендпоінтів.

Для управління списками творів створено модель Items, а також ендпоінти для додавання, редагування та отримання списків. Користувач може додавати твори до свого списку, вказуючи статус і прогрес. Цей ендпоінт додає твір до списку авторизованого користувача, зберігаючи його статус («planned», «watching_reading», «completed») і прогрес.

До інших компонентів серверної частини входить обробка запитів для пошуку творів і генерація рекомендацій. Наприклад, ендпоінт для пошуку повертає список творів, що відповідають пошуковому запиту, використовуючи нечутливий до регістру пошук (ilike).

Серверна частина на FastAPI інтегрується з React через API, забезпечуючи швидку обробку запитів завдяки асинхронності фреймворку. PostgreSQL зберігає всі дані, а Tailwind CSS на стороні клієнта відповідає за стилізацію інтерфейсу.

3.1.2 Створення клієнтської частини

Клієнтська частина вебсервісу для каталогізації та управління переглянутими аніме та мангою розроблена з використанням бібліотеки React, яка забезпечує створення динамічного та інтерактивного інтерфейсу користувача. Стилізація здійснюється за допомогою Tailwind CSS, що дозволяє швидко створювати адаптивний і сучасний дизайн. Клієнтська частина взаємодіє з серверною частиною, побудованою на FastAPI, через REST API, а дані зберігаються в базі даних PostgreSQL. У цьому розділі описано процес підключення до бази даних через API, авторизацію та реєстрацію користувача, вивід списку творів, додавання нового твору та інші ключові компоненти клієнтської частини з прикладами коду реалізації.

Клієнтська частина не підключається до бази даних PostgreSQL напряму, а використовує серверну частину FastAPI як посередника. Для цього в React створено сервіс для відправлення HTTP-запитів до API ендпоінтів, які повертають дані з бази. Використовується бібліотека axios для спрощення запитів.

Тут після успішного входу токен зберігається, і користувач перенаправляється на сторінку списку творів.

Список творів користувача відображається шляхом запиту до ендпоінта /user_lists на сервері FastAPI. Дані отримуються через axios і відображаються в компоненті з використанням Tailwind CSS для стилізації. Цей компонент отримує список творів і відображає їх у картках із назвою, типом, статусом і прогресом.

Додавання нового твору реалізується через форму, яка відправляє дані на ендпоінт /user_lists. Користувач вводить інформацію про твір, обирає статус і вказує прогрес. Ця форма дозволяє додати твір до списку, відправляючи запит із введеними даними.

До компонентів також входить пошук творів, який відправляє запит до ендпоінта пошуку і відображає результати. Рекомендації можуть бути додані через окремий ендпоінт на сервері, який аналізує списки користувача.

Клієнтська частина на React із Tailwind CSS забезпечує інтерактивний інтерфейс, який інтегрується з FastAPI через API, а PostgreSQL зберігає дані, доступ до яких здійснюється через сервер.

3.2 Інструкція користувача

Цей розділ є посібником для користувачів вебсервісу, розробленого для каталогізації та управління переглянутими аніме та мангою. Сервіс дозволяє створювати персональні списки творів, відстежувати прогрес перегляду чи читання, шукати нові аніме та мангу, а також отримувати рекомендації. Інструкція описує повний процес роботи з сервісом, від першого входу до використання всіх основних функцій, і призначена для забезпечення зручного та інтуїтивно зрозумілого досвіду.

Для початку роботи з сервісом користувачу необхідно відкрити веббраузер і перейти за адресою.

На головній сторінці відображається два рощділи Аніме та Манга із популярними творами. Вгорі приведена навігаційна панель для перезоду між сторінками (рис 1.3).

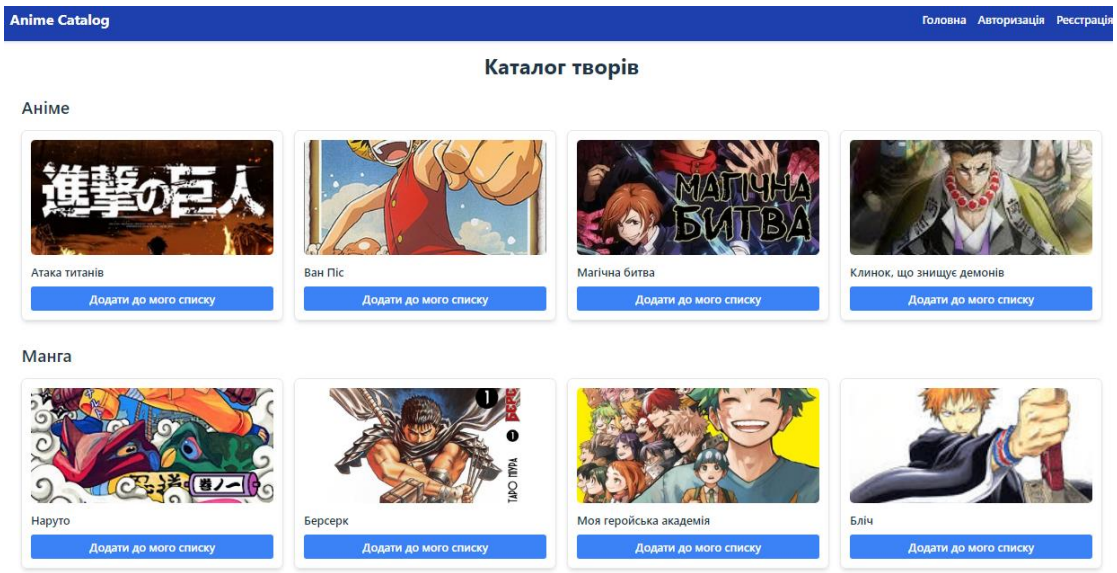

Рисунок 3.1 – Головна сторінка

Через навігаційну панель можна потрапити на форми реєстрації та авторизації (рис 3.2).

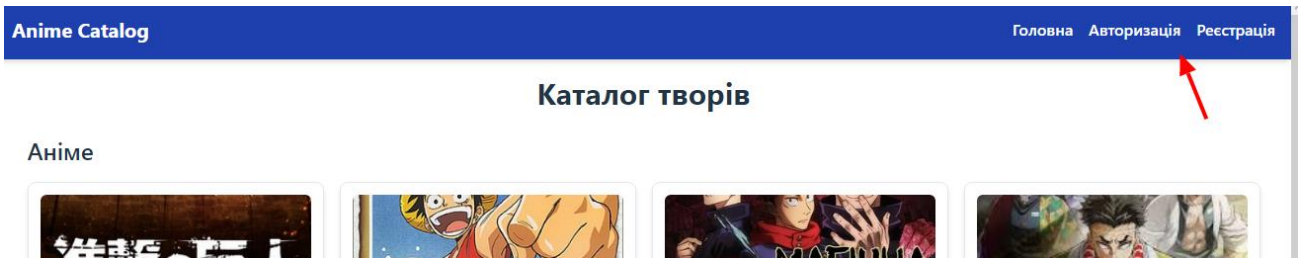

Рисунок 3.2 – Навігаційна панель

Якщо обрати кнопку «Реєстрація», то відбудеться перенаправлення на відповідну сторінку із формою реєстрації (рис. 3.3).

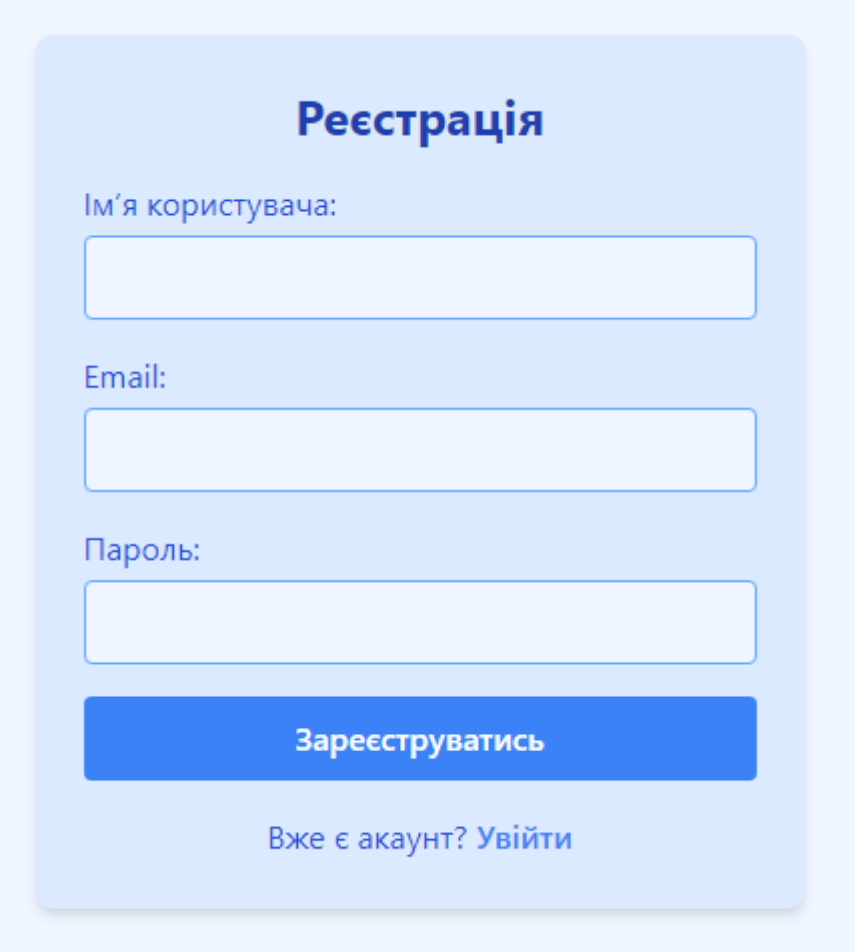

Рисунок 3.3 – Сторінка реєстрації

На сторінці реєстрації користувачу пропонується форма, де потрібно ввести електронну пошту, ім'я користувача та пароль. Електронна пошта має бути унікальною і слугуватиме для входу в систему. Ім'я користувача відображатиметься в інтерфейсі, наприклад, у списках чи коментарях, і також має бути унікальним. Пароль рекомендується зробити достатньо складним для забезпечення безпеки. Після заповнення всіх полів потрібно натиснути кнопку «Зареєструватися». У разі успіху з'явиться повідомлення про успішну реєстрацію, і користувач буде перенаправлений на сторінку входу (рис. 3.4).

На сторінці входу користувачу необхідно ввести свою електронну пошту та пароль, які були вказані під час реєстрації. Після натискання кнопки «Увійти» система перевірить дані, і в разі успіху користувач буде перенаправлений на сторінку зі своїм списком творів. Якщо дані введено

неправильно, з’явиться повідомлення про помилку. Також на цій сторінці доступне посилання «Забули пароль?» для відновлення доступу, хоча ця функція може бути реалізована в майбутніх версіях сервісу.

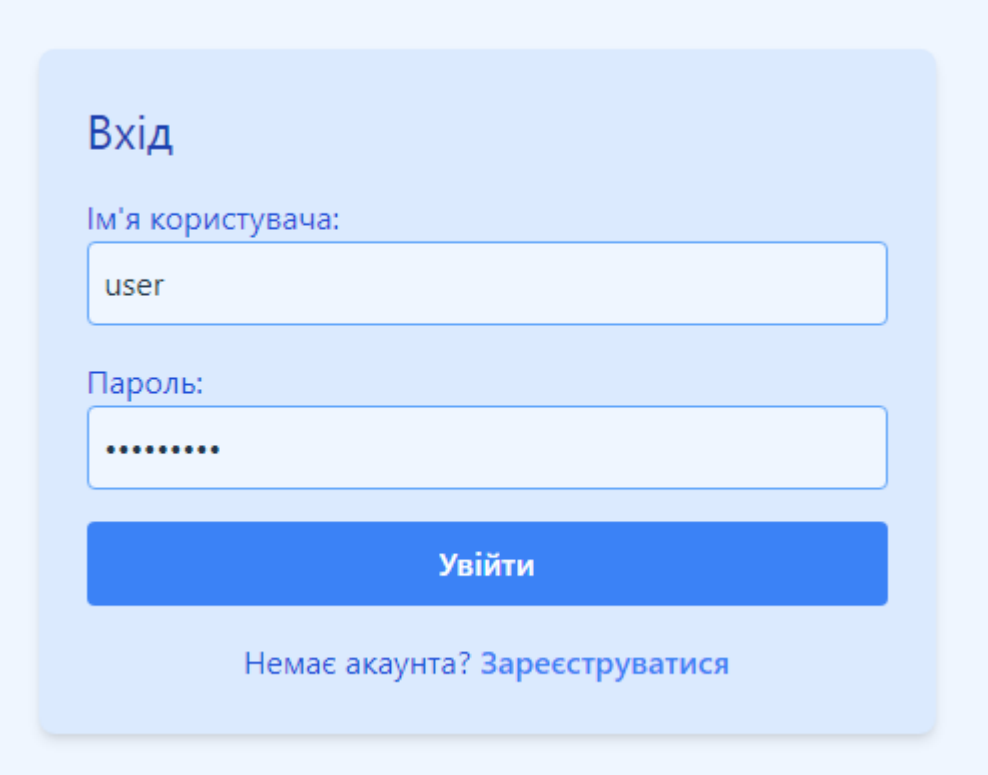

Рисунок 3.4 – Сторінка входу з формою та кнопкою «Увійти»

Після входу користувач потрапляє на сторінку «Мій каталог аніме та манги», де відображаються всі додані ним аніме та манги. (рис 3.5) Кожен запис виводиться карткою, що включає обкладинку, назву твору, його тип (аніме чи манга), статус («В планах», «Читаю/Дивлюсь», «Прочитано/Переглянуто») і прогрес (наприклад, «Розділ 7» для п’яти переглянутих епізодів із дванадцяти), оцінку, жанр.

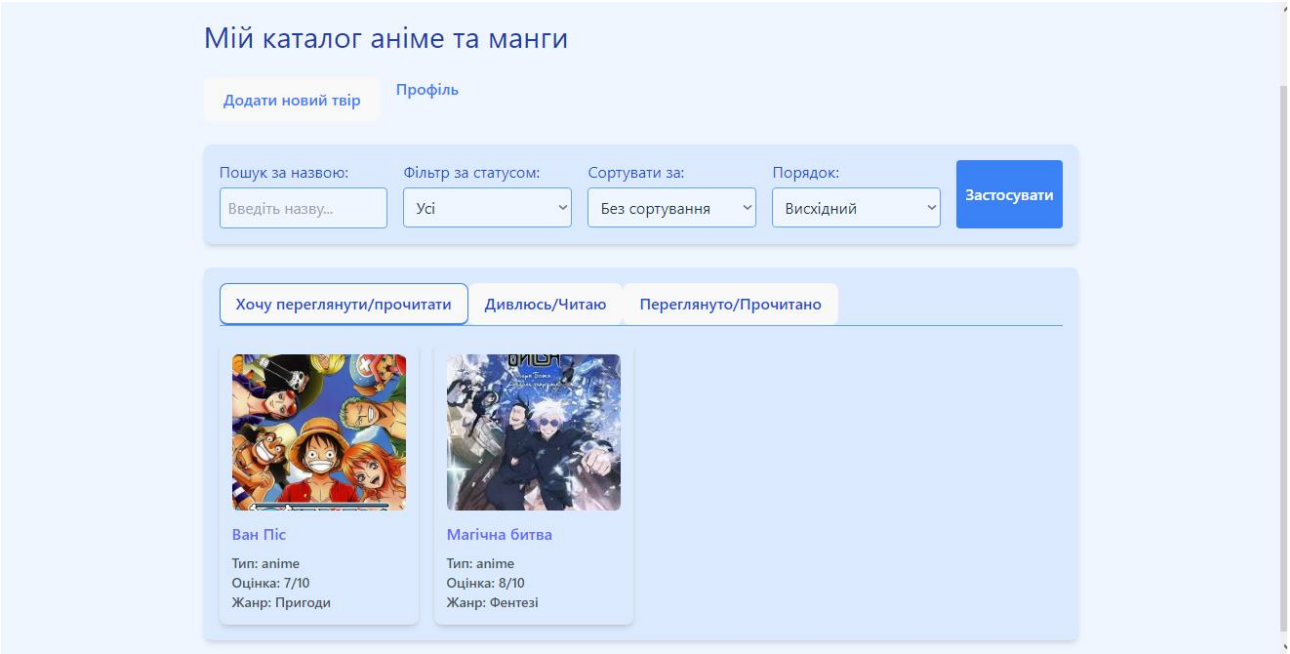

Рисунок 3.5 – Сторінка «Мій каталог аніме та манги» із прикладами доданих творів

Список можна сортувати за статусом, оцінкою, жаром обирючи відповідний фільтр (рис 3.6).

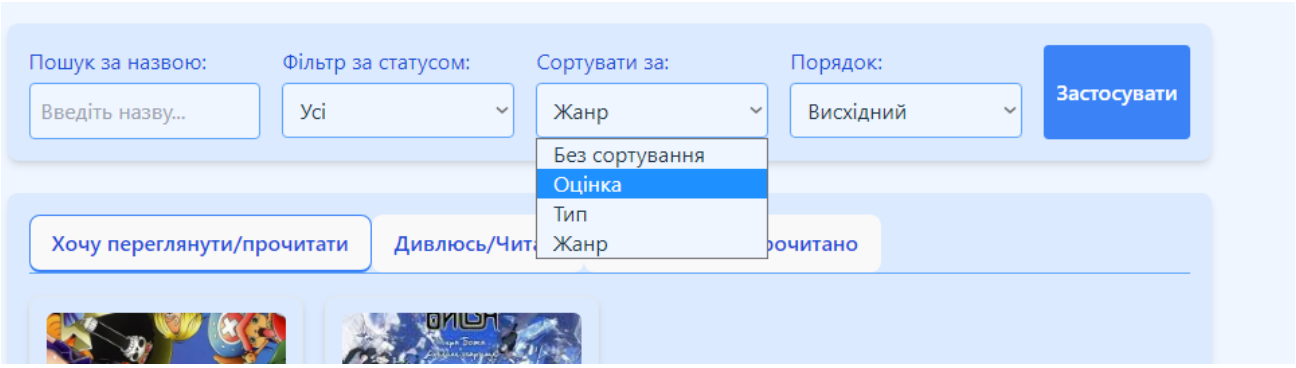

Рисунок 3.6 – Приклад фільтру

Щоб додати новий твір до свого списку користувач може обрати твір із головної сторінки і натиснути на обраному творі кнопку «Додати до мого списку» (рис 3.7).

Рисунок 3.7 – Кнопка «Додати до мого списку»

Після вибору твору, він з'явиться на стрінці списку користувача (рис. 3.8).

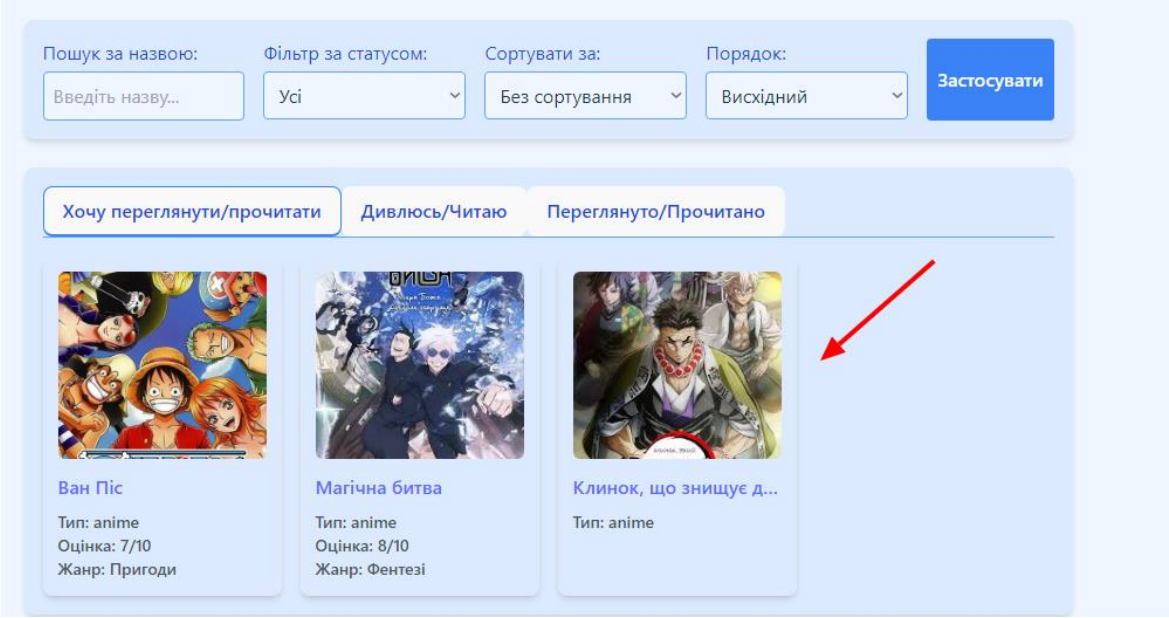

Рисунок 3.8 – Оновлений список творів користувача із доданим твором

Щоб додати новий твір, якого немає на головній сторінці, користувачу потрібно натиснути кнопку «Додати твір» на сторінці списку (рис 3.9).

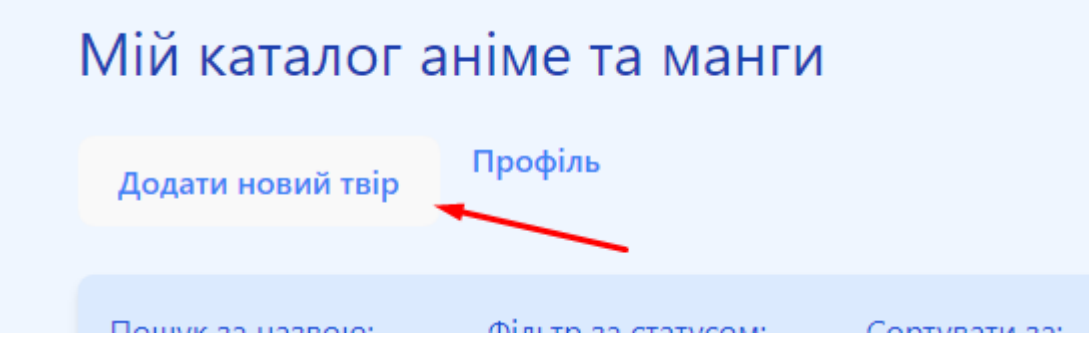

Рисунок 3.9 – Кнопка додавання нового твору

Відкриється форма, де необхідно вказати назву твору, обрати статус із випадаючого списку («В планах», «Читаю/Дивлюсь», «Прочитано/Переглянуто»), додати посилання на обкладинку, оцінку (потім можна змінити), жанр (рис 3.10). Після натискання кнопки «Додати» твір з’явиться у списку, а користувач отримає повідомлення про успішне додавання. Якщо твір із таким ідентифікатором не існує, з’явиться повідомлення про помилку.

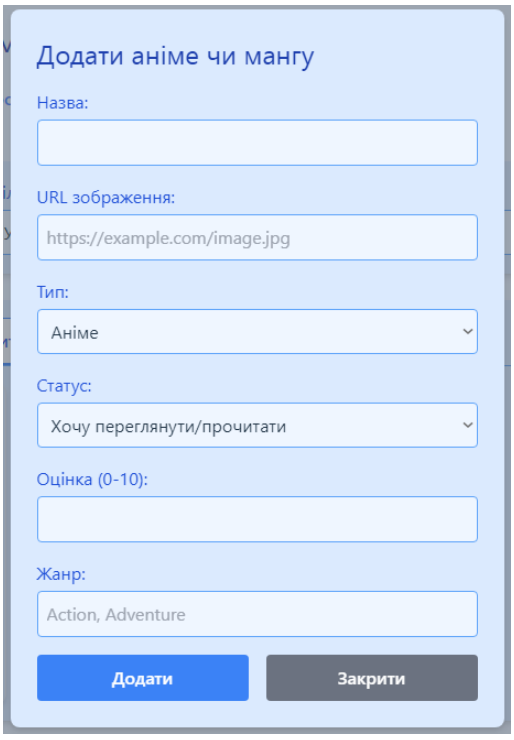

Рисунок 3.10 – Форма додавання твору з полями та кнопкою «Додати»

Для пошуку нових аніме чи манги користувач може перейти до розділу «Пошук» (рис 3.11). Для цього в полі пошуку потрібно ввести запит (наприклад, назви твору).

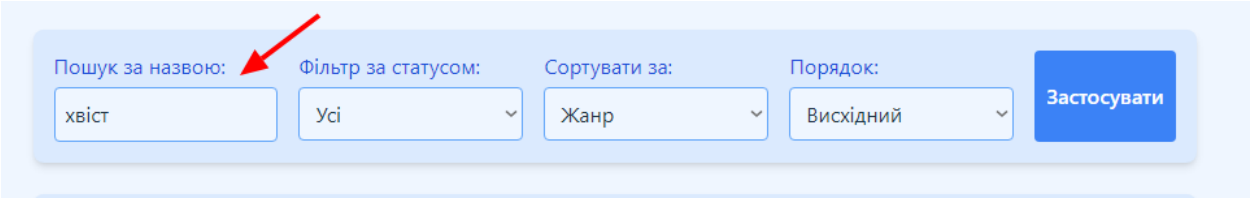

Рисунок 3.11 – Поле пошуку

Після введення тексту та натискання кнопки «Застосувати» система відобразить список відповідних творів із їхніми назвами та типами. Кожен результат можна натиснути, щоб переглянути деталі, або використати його ідентифікатор для додавання до списку (рис 3.12). Якщо результатів немає, відображається повідомлення «Нічого не знайдено».

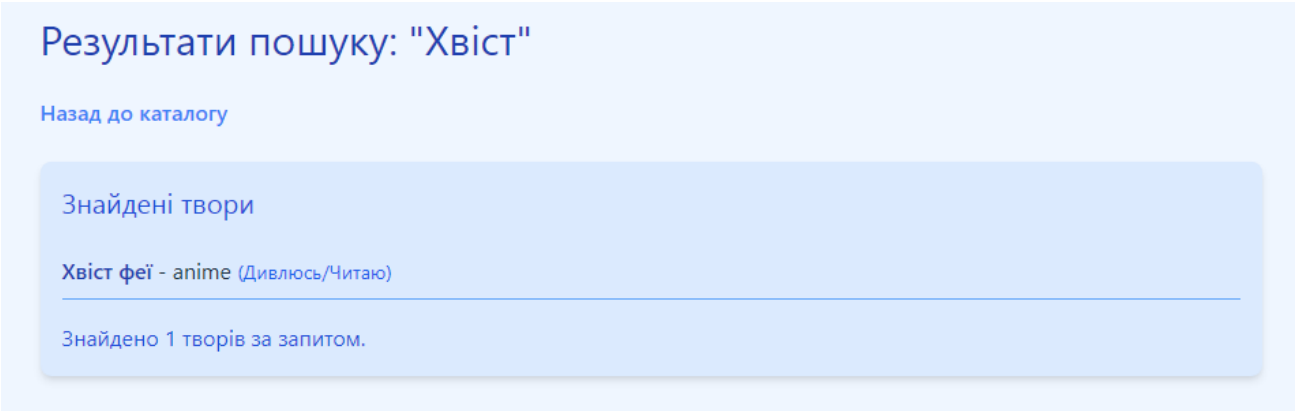

Рисунок 3.12 – Сторінка результату пошуку з полем введення та результатами

Щоб переглянути інформацію про аніме чи мангу із списку потрібно натиснути на назву обраного твору. Після цього відкриється сторінка із детальним описом твору та формою відгуку для даного твору (рис 3.13).

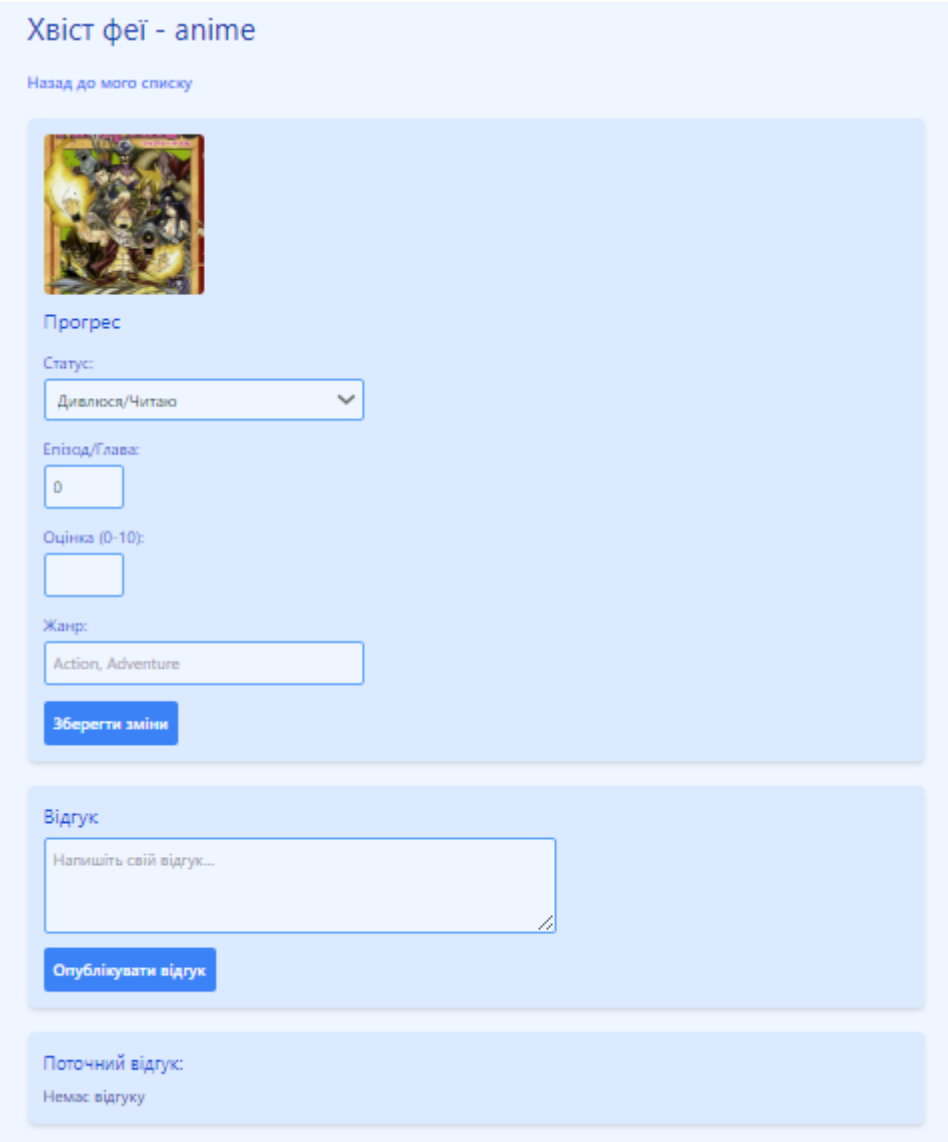

Рисунок 3.13 – Детальна сторінка твору

Шоб внести дані про прогрес роботи із твором потрібно внести зміни у відповідні поля і натиснути зберегти зміни (рис. 3.14).

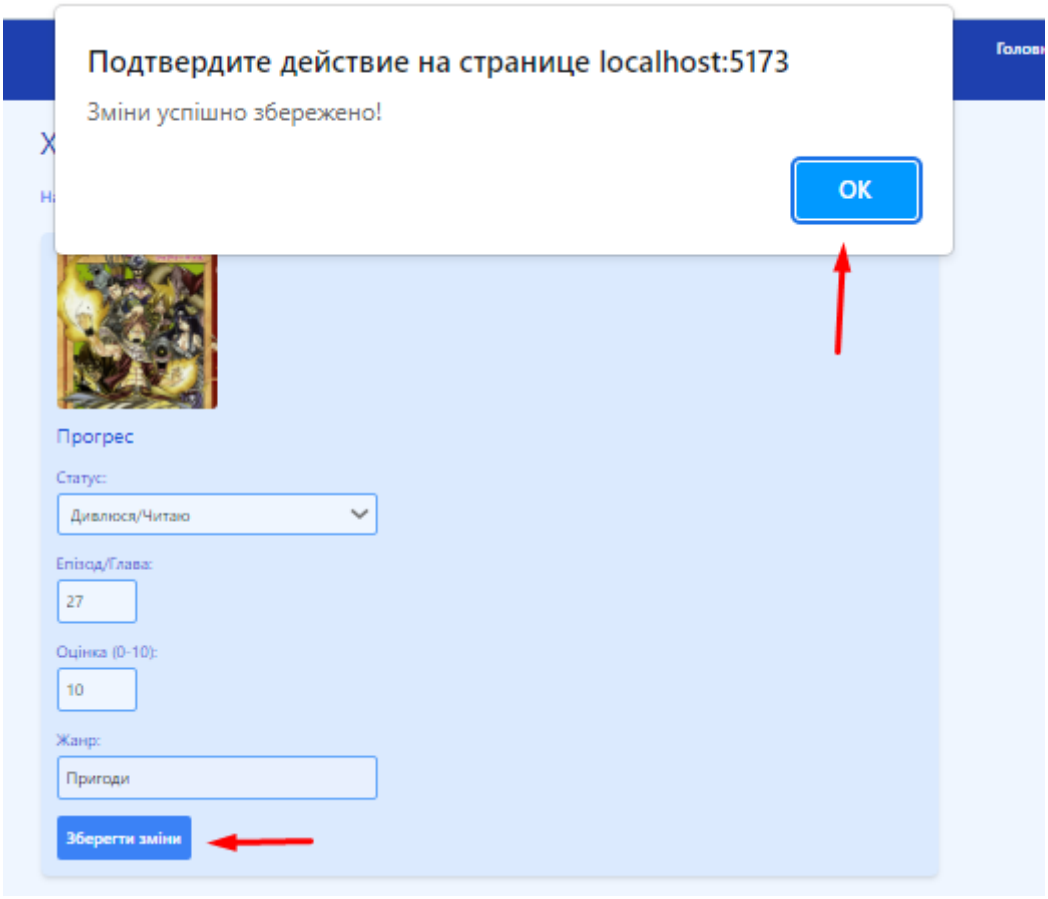

Рисунок 3.14 – Внесення змін у прогрес твору

Після збереження змін вони з’являться в інформації про твір у списку (рис. 3.15).

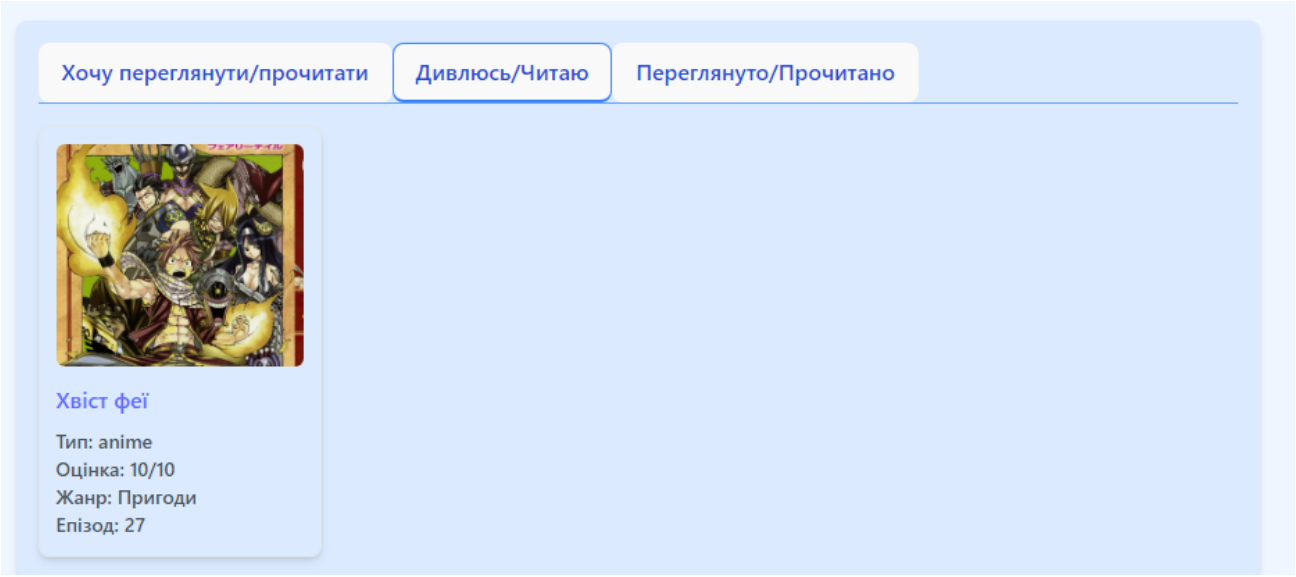

Рисунок 3.15 – Збережені зміни прогресу твору

У розділі «Профіль», доступному через навігаційну панель, користувач може переглянути свої дані, такі як ім’я користувача чи електронна пошта (рис 3.16).

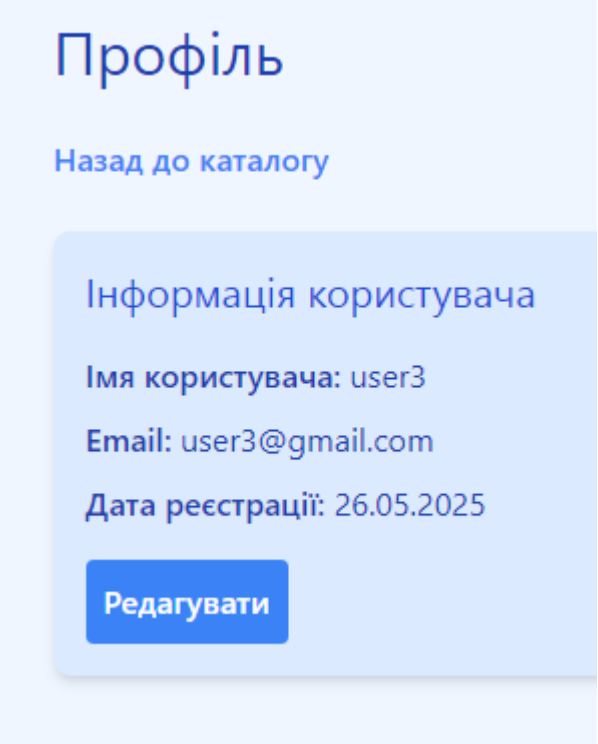

Рисунок 3.16 – Сторінка профілю з даними користувача та кнопками

Щоб редагувати свої дані, такі як ім’я користувача чи електронна пошта, потрібно натиснути кнопку «Редагувати» і у відкритому модальному вікні внести потрібні зміни і зберегти (рис 3.17).

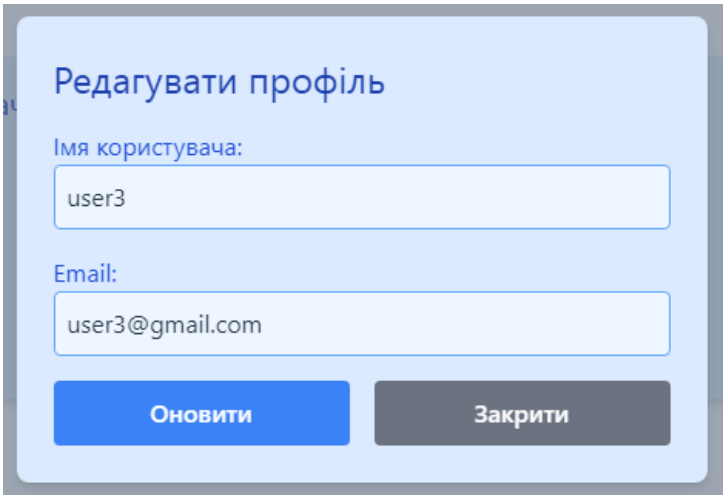

Рисунок 3.17 – Редагування профілю

Щоб вийти з облікового запису потрібно натиснути кнопку «Вийти» у навігаційній панелі, після чого користувач повертається на головну сторінку (рис 3.18).

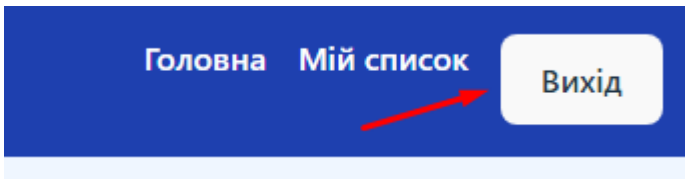

Рисунок 3.18 – Кнопка виходу

Сервіс включає навігаційну панель, яка завжди доступна у верхній частині сторінки. Вона містить посилання на «Головна», в залежності чи авторизований користувач «Реєстрація», «Авторизація» чи «Мій список», «Вийти», дозволяючи швидко переходити між розділами (рис 3.19). Усі сторінки адаптовані для різних пристроїв (комп'ютерів, планшетів, смартфонів) завдяки використанню Tailwind CSS. Якщо під час роботи виникають помилки (наприклад, проблеми з сервером), користувач отримує відповідне повідомлення на екрані.

Рисунок 3.19 – Приклад меню із навігацією

Ця інструкція охоплює всі етапи роботи з сервісом, від реєстрації до використання додаткових функцій. Користувачу достатньо слідувати описаним крокам, щоб ефективно вести облік своїх аніме та манги.

3.3 Заходи щодо поліпшення вебзастосунку

Розроблений вебсервіс для каталогізації та управління переглянутими аніме та мангою є функціональним рішенням, яке відповідає основним потребам користувачів. Однак, враховуючи потенціал для розширення аудиторії та підвищення якості користувацького досвіду, існує низка заходів, які можуть поліпшити вебзастосунок.

Першим заходом є розширення соціальних функцій. Додавання можливості створювати спільні списки для групового перегляду чи читання, а також інтеграція чату чи форуму для обговорення творів, підвищить залученість спільноти. Наприклад, можна реалізувати функцію «Друзі», яка дозволить користувачам додавати одне одного в список контактів і отримувати сповіщення про оновлення їхніх списків.

Другим напрямком є створення рекомендаційної системи. Аналізуючи оцінки, прогрес і вподобання користувачів, система могла б пропонувати точніші рекомендації. Інтеграція з зовнішніми джерелами, такими як MyAnimeList, для автоматичного імпорту даних про популярні твори також підвищила б якість пропозицій.

Третім заходом є оптимізація продуктивності. Зі збільшенням кількості користувачів і записів у базі даних PostgreSQL можуть виникати затримки при обробці запитів. Для цього пропонується додати кешування даних із

використанням Redis, що прискорить доступ до часто запитаних списків чи результатів пошуку. Крім того, асинхронна обробка запитів на сервері FastAPI може бути розширена для зменшення часу очікування, особливо при генерації рекомендацій чи завантаженні великих списків.

Четвертим напрямком є поліпшення інтерфейсу користувача. Хоча Tailwind CSS забезпечує адаптивний дизайн, додавання темної теми як опції за замовчуванням із можливістю перемикання підвищить комфорт для користувачів, які віддають перевагу такому стилю. Також варто реалізувати функцію перетягування (drag-and-drop) для сортування творів у списку, що зробить управління більш інтуїтивним. Додавання анімацій для переходів між сторінками чи відображення списків покращить візуальний досвід.

П’ятим заходом є підвищення безпеки. Наразі авторизація базується на JWT-токенах, але можна додати двофакторну автентифікацію (2FA) через електронну пошту чи SMS для захисту облікових записів від несанкціонованого доступу. Також варто впровадити захист від SQL-ін’єкцій і XSS-атак, якщо цього ще не зроблено, шляхом ретельної валідації вхідних даних на сервері FastAPI і використання бібліотек для очищення введених користувачем даних на клієнтській частині React.

Шостим напрямком є додавання офлайн-функціональності. Наразі сервіс працює лише онлайн, але впровадження локального кешу через Service Workers у React дозволить користувачам переглядати свої списки без підключення до Інтернету. Синхронізація даних із сервером відбуватиметься автоматично після відновлення зв’язку, що зробить сервіс зручнішим у ситуаціях із нестабільним інтернетом.

Сьомим заходом є розширення функціональності пошуку. Додавання фільтрів за жанром, роком випуску чи кількістю епізодів/глав у пошуковій системі дозволить користувачам швидше знаходити потрібний контент. Також можна реалізувати автодоповнення запитів на основі популярних творів, що спростить введення пошукових фраз.

Нарешті, для зручності користувачів пропонується додати функцію експорту списків у форматах PDF чи Excel, а також можливість імпорту даних із інших сервісів (наприклад, CSV-файлів). Це дозволить легко переносити списки між платформами та зберігати їх у зрозумілому вигляді поза сервісом.

Реалізація цих заходів потребуватиме додаткового часу та ресурсів, але значно підвищить цінність вебзастосунку. Пріоритетність поліпшень залежатиме від зворотного зв’язку користувачів і стратегії подальшого розвитку проєкту. Впровадження хоча б частини цих змін зробить сервіс більш привабливим для любителів аніме та манги, забезпечить його конкурентоспроможність і відкриє можливості для масштабування в майбутньому.

ВИСНОВКИ

У рамках кваліфікаційної роботи було створено вебзастосунок «Персональна бібліотека аніме та манги», який слугує зручним інструментом для каталогізації та управління переглянутими аніме та мангою. Цей проєкт охопив повний цикл розробки програмного продукту – від теоретичного аналізу предметної області до практичної реалізації та планування подальших удосконалень, що дозволило досягти цілей і завдань, окреслених у вступі. Розробка сервісу стала відповіддю на потреби любителів аніме та манги, забезпечивши їх функціональним рішенням для організації переглянутого контенту.

Перший розділ роботи був присвячений ретельному аналізу методів і засобів, необхідних для створення застосунку. Дослідження перспективності розробки вебсервісу показало, що популярність аніме та манги у світі стрімко зростає, а разом із цим збільшується попит на спеціалізовані платформи для ведення обліку такого контенту. Огляд існуючих платформ, таких як MyAnimeList, AniList і Kitsu, дав змогу оцінити їхні можливості та виявити як сильні сторони – наприклад, великі бази даних і розвинені соціальні функції, так і слабкі – зокрема, перевантаженість інтерфейсу чи недостатня гнучкість налаштувань. Ці спостереження стали основою для формування концепції власного сервісу, який би усував виявлені недоліки та пропонував унікальні переваги. Аналіз інструментарію підтвердив доцільність вибору сучасного стеку технологій: React для створення динамічного та інтерактивного інтерфейсу, FastAPI для розробки швидкого й асинхронного API, PostgreSQL для надійного й структурованого зберігання даних і Tailwind CSS для реалізації адаптивного та естетичного дизайну. Постановка задачі чітко визначила мету проєкту – створити зручний, інтуїтивно зрозумілий і функціональний вебсервіс, який відповідає потребам цільової аудиторії.

Другий розділ зосередився на проєктуванні системи, заклавши міцний фундамент для подальшої реалізації. Розробка архітектури та визначення етапів створення дозволили організувати процес роботи структуровано й послідовно, врахувавши всі ключові аспекти майбутнього сервісу. Визначення вимог до системи дало змогу сформувати чітке бачення її функціональності, продуктивності та зручності для користувачів, що стало орієнтиром для наступних етапів. Опис сторінок і компонентів інтерфейсу детально розкрив структуру сервісу, включаючи головну сторінку, особистий список творів, пошук та профіль користувача, що забезпечило продуманий підхід до користувацького досвіду. Проєктування бази даних стало важливим кроком для створення реляційної моделі, яка включає таблиці для користувачів, творів і персональних списків із відповідними зв’язками, що гарантує ефективне зберігання й обробку даних про аніме та мангу. Обґрунтування вибору Visual Studio Code як середовища розробки підкреслило його переваги, такі як легкість, гнучкість і підтримка обраного стеку технологій, що зробило його оптимальним інструментом для реалізації проєкту.

Третій розділ став практичним втіленням усіх попередніх напрацювань, забезпечивши програмну реалізацію застосунку. Серверна частина, створена на основі FastAPI, реалізувала REST API для обробки запитів від клієнта, підключення до бази даних PostgreSQL і механізми авторизації через JWT-токени, що забезпечило безпечний доступ до системи. Клієнтська частина, розроблена з використанням React і стилізована за допомогою Tailwind CSS, створила інтерактивний і адаптивний інтерфейс, який дозволяє користувачам реєструватися, входити в систему, управляти списками творів, шукати контент і переглядати рекомендації. Інструкція користувача надала вичерпний посібник із використання сервісу, детально описавши кожен крок — від реєстрації до роботи з основними функціями, що полегшує освоєння застосунку навіть для новачків. Пропозиції щодо поліпшення вебзастосунку окреслили перспективи його розвитку, включаючи впровадження соціальних

функцій, таких як спільні списки чи чат, удосконалення рекомендаційної системи за допомогою машинного навчання, оптимізацію продуктивності через кешування, підвищення безпеки з двофакторною автентифікацією та додавання офлайн-режиму для роботи без підключення до Інтернету.

Розроблений вебсервіс «Персональна бібліотека аніме та манги» став повноцінним рішенням для ведення обліку переглянутого контенту, відповідаючи поставленим завданням і потребам цільової аудиторії. Використання сучасного стеку технологій забезпечило високу продуктивність, адаптивність і зручність використання, що робить сервіс конкурентоспроможним серед аналогів. У майбутньому впровадження запропонованих удосконалень, таких як розширення соціальних можливостей, інтеграція з зовнішніми платформами чи додавання нових функцій, може значно підвищити його привабливість і функціональність. Таким чином, створений вебзастосунок не лише виконує своє основне призначення, але й має потенціал для розвитку, що відкриває перспективи для його масштабування та залучення ширшої аудиторії любителів аніме та манги.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Власов Д. Я. Популярність жанру аніме в украннній культурі: український контекст. Актуальні проблеми сучасності. Молодіжна наука КНУКіМ 2024: матеріали Всеукраїнської науково-практичної конференції здобувачів вищої освіти і молодих учених, Київ, 08–09 листопада 2024 р. Вид.центр КНУКіМ, 2024. с. 155-157.

2. Довженко І., Олейнікова І., Лисова О., Левченко В. Інтеграція стилю аніме в українську культуру. Актуальні проблеми сучасного дизайну: Матеріали IV Міжнародна наук.-практ. конф., Київ 27 квіт. 2022р. С 259–262.

3. MyAnimeList vs. Other Platforms: Unique Benefits for Online Visibility. URL: https://vocal.media/education/my-anime-list-vs-other-platforms-unique-benefits-for-online-visibility

4. Березовська Л.І. Вплив аніме-культури на формуванні Я-концепції особистості. Вісник Національного унівеситету оборони України. Вип.3. 2023. С.27 – 33.

5. MyAnimeList. URL: https://myanimelist.net

6. MyAnimeList. URL: https://uk.wikipedia.org/wiki/MyAnimeList

7. AniList. URL: https://anilist.co

8. Introduction | AniList API Docs. URL: https://docs.anilist.co/guide/introduction

8. Kitsu. URL: https://kitsu.app/explore/anime

10. Kitsu.io is using App Fixers. URL: https://medium.com/app-fixers/kitsu-io-is-using-app-fixers-b20032d43bac

11. MyAnimeList. URL: https://www.linkedin.com/company/myanimelist/

12. Which is better? Anilist vs MyAnimeList. URL: https://www.reddit.com/r/anime/comments/zoe3vq/which_is_better_anilist_vs_myanimelist/

13. Kitsu connects anime and manga fans from all over the world. URL: https://nextpittsburgh.com/business-tech-news/kitsu-a-website-based-in-blairsville-connects-anime-and-manga-fans-from-all-over-the-world/

14. Alex Banks, Eve Porcello. Learning React: Modern Patterns for Developing React Apps 2nd Edition. O'Reilly Media. 2020. c. 310.

15. Robin Wieruch. The Road to React: The React.js 19 with Hooks in JavaScript Book (2025 Edition) Kindle Edition. 2017. c. 410.

16. ReactJS - Advantages & Disadvantages. URL: https://medium.com/@muhamedsalihseyedibrahim/reactjs-advantages-disadvantages-16f479b3aa47

17. Giunio De Luca. FastAPI Cookbook: Develop high-performance APIs and web applications with Python. Packt Publishing. 2024. c. 358.

18. FastAPI. Tutorial - User Guide. URL: https://fastapi.tiangolo.com/tutorial/

19. FastAPI - Advantages and Disadvantages. URL: https://dev.to/shariqahmed525/fastapi-advantages-and-disadvantages-197o

20. Luca Ferrari, Enrico Pirozzi. Learn PostgreSQL - Second Edition: Use, manage and build secure and scalable databases with PostgreSQL 2nd ed. Edition. Packt Publishing. 2023. c. 744.

21. PostgreSQL. URL: https://www.postgresql.org

22. PostgreSQL Advantages and Disadvantages. URL: https://www.aalpha.net/blog/pros-and-cons-of-using-postgresql-for-application-development/

23. Roberto Rescigno. Tailwind CSS: a guide to using the popular utility-first CSS framework. Kindle Edition. 2023. c. 49.

24. A Comprehensive Introduction to Tailwind CSS. URL: https://medium.com/@alifm2101/a-comprehensive-introduction-to-tailwind-css-36bc9cb81a1c

25. Pros and Cons of using Tailwind CSS. URL: https://www.codewalnut.com/learn/pros-and-cons-of-using-tailwind-css

26. What is PostgreSQL?. URL: https://www.ibm.com/think/topics/postgresql

27. Visual Studio Code. URL: https://code.visualstudio.com

28. What is Visual Studio Code? Microsoft’s extensible code editor. URL: https://www.infoworld.com/article/2335960/what-is-visual-studio-code-microsofts-extensible-code-editor.html

29. What are the advantages and disadvantages of using Visual Studio Code or Atom?. URL: https://medium.com/@ssc.ahmed.926748/what-are-the-advantages-and-disadvantages-of-using-visual-studio-code-or-atom-d3132bf1af85

30. Visual Studio vs Visual Studio Code: What’s the Difference?. URL: https://codeop.tech/visual-studio-vs-visual-studio-code-whats-the-difference/