

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Система енциклопедій манга»

Виконав студент групи ІПЗк-23-1
ЧУПРИНА Володимир Андрійович

Керівник роботи:
САВІЦЬКИЙ Роман Святославович

Рецензент:
ГОДЛЕВСЬКИЙ Юрій Олександрович

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна Вакалюк
«26» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ЧУПРИНА Володимир Андрійович**

Керівник роботи: **САВІЦЬКИЙ Роман Святославович**

Тема роботи: **«Система енциклопедій манга»**,

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: Об'єктом дослідження є процес створення та функціонування інтерактивних вебплатформ для систематизації та поширення цифрового медіаконтенту. Предметом дослідження є технології та архітектурні підходи для розробки повнофункціональних вебзастосунків, зокрема використання фулстек-фреймворку Next.js, системи управління базами даних PostgreSQL та ORM Prisma для створення масштабованої та орієнтованої на користувача платформи для систематизації та подання контенту.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Савіцький Р. С.	27.02.2025р.	27.02.2025р.
2	Савіцький Р. С.	27.02.2025р.	27.02.2025р.
3	Савіцький Р. С.	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Кваліфікаційна робота освітнього ступеня «бакалавр» складається з вебплатформи «Система енциклопедій манга» та пояснювальної записки. Пояснювальна записка до кваліфікаційної роботи містить 71 сторінку, 48 ілюстрацій, 24 таблиці та 7 сторінок додатків. Загальний обсяг 78 сторінок.

Метою роботи є розробка вебплатформи «Система енциклопедій манга». Система включає модулі для управління контентом, автентифікації та авторизації користувачів, розмежування прав доступу, коментування, створення персоналізованих колекцій та отримання сповіщень.

У роботі поставлені основні завдання на розробку системи. Було проведено порівняльний аналіз аналогічних платформ для виявлення їх переваг та недоліків. Було обрано інтегровану архітектуру на базі фулстек-фреймворку Next.js. Приведені основні сценарії використання системи, логіка взаємодії між модулями, діаграми класів, компонентів, послідовності та розгортання, а також алгоритми роботи ключових модулів. Спроектовано базу даних з використанням СУБД PostgreSQL. Клієнтська та серверна частини розроблені в єдиному середовищі Next.js з використанням мови TypeScript, JavaScript та ORM Prisma.

КЛЮЧОВІ СЛОВА: МАНГА, ВЕБПЛАТФОРМА, NEXT.JS, REACT, POSTGRESQL, TYPESCRIPT, JAVASCRIPT, PRISMA, УПРАВЛІННЯ КОНТЕНТОМ.

					ІПЗ.КР.Б – 121 – 25 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Система енциклопедій манга Пояснювальна записка	Лім.	Арк.	Аркушів
Розроб.		Чуприна В.А.					2	76
Керівник		Савіцький Р.С.						
Рецензент		Годлевський Ю.О.				Житомирська політехнічна група ІПЗк-23-1		
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's qualification thesis consists of the 'Manga Encyclopedia System' web platform and an explanatory note. The explanatory note to the qualification work contains 71 pages, 48 illustrations, 24 tables and 7 pages of appendices. The total volume is 78 pages.

The goal of the work is to develop the 'Manga Encyclopedia System' web platform. The system includes modules for content management, user authentication and authorization, access rights differentiation, commenting, creating personalized collections, and receiving notifications.

The work sets out the main tasks for the system's development. A comparative analysis of similar platforms was conducted to identify their advantages and disadvantages. An integrated architecture based on the Next.js full-stack framework was chosen. The main use case scenarios, the logic of interaction between modules, class, component, sequence, and deployment diagrams, as well as the algorithms for key modules are presented. A database was designed using the PostgreSQL DBMS. The client and server parts were developed in a unified Next.js environment using the TypeScript, JavaScript languages and the Prisma ORM.

KEYWORDS: MANGA, WEB PLATFORM, NEXT.JS, REACT, POSTGRESQL, TYPESCRIPT, JAVASCRIPT, PRISMA, CONTENT MANAGEMENT.

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ	9
1.1. Постановка задачі	9
1.2. Аналіз аналогів програмного продукту	14
1.3. Обґрунтування вибору інструментальних засобів	20
Висновки до першого розділу	24
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ ЕНЦИКЛОПЕДІЙ МАНГА	25
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи	25
2.2 Розробка бази даних системи	29
2.3 Проєктування та реалізація алгоритмів роботи системи	36
2.4 Реалізація системи енциклопедій манга	40
Висновки до другого розділу	51
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУ	53
3.1 Порядок встановлення та налаштування системи	53
3.2 Структура інтерфейсу та порядок взаємодії з системою енциклопедій манга	54
3.3 Тестування роботи програмного продукту	61
Висновки до третього розділу	66
ВИСНОВКИ	67
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	68
ДОДАТКИ	72

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ACID – Atomicity, Consistency, Isolation, Durability (Атомарність, Узгодженість, Ізольованість, Довговічність)

API – Application Programming Interface (Програмний інтерфейс застосунку)

HTTP – HyperText Transfer Protocol (Протокол передачі гіпертексту)

JSON – JavaScript Object Notation (Нотація об'єктів JavaScript)

ORM – Object-Relational Mapping (Об'єктно-реляційне відображення)

SQL – Structured Query Language (Мова структурованих запитів)

SSR – Server-Side Rendering (Рендеринг на стороні сервера)

UI – User Interface (Інтерфейс користувача)

UX – User Experience (Досвід користувача)

ПЗ – Програмне забезпечення

СЕМ – Система енциклопедій манга

СУБД – Система управління базами даних

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми. Перетворившись на потужний глобальний культурний феномен, японські комікси (манга) водночас створили для своєї аудиторії цифровий ландшафт, що характеризується високою фрагментацією. Читачі змушені навігувати між численними, часто непов'язаними, онлайн-ресурсами, що ускладнює не лише пошук контенту, а й відстеження оновлень та доступ до багатомовних версій. Існуючі платформи рідко пропонують ефективні інструменти для персоналізації та соціальної взаємодії, перекладаючи на користувача обов'язок самостійного моніторингу та обмежуючи можливості для формування спільноти.

Ця ситуація породжує попит на інтегроване рішення, яке б не лише агрегувало контент, а й збагачувало читацький досвід через проактивні сповіщення, зручну навігацію та інструменти для обговорення. Розробка такої системи, як «Система енциклопедій манга», пропонує централізоване середовище, що ставить у пріоритет зручність та залученість користувача. Для авторів та перекладачів така платформа стає зручним інструментом для публікації та просування своїх робіт, забезпечуючи прямий зв'язок з аудиторією.

Метою кваліфікаційної роботи є створення програмного продукту «Система енциклопедій манга», який долає проблему фрагментації контенту, надаючи централізовану платформу для його споживання, обговорення та відстеження.

Для досягнення поставленої мети визначено основні завдання, що охоплюють повний цикл розробки. Після проведення аналізу аналогів та обґрунтування вибору технологічного стеку, було сформовано наступні етапи роботи. Ключовим завданням стала розробка програмної архітектури, що включає створення надійної серверної частини на базі фреймворку Next.js з використанням ORM Prisma, та побудову адаптивної клієнтської частини на бібліотеці React. Далі було реалізовано основні функціональні модулі: систему автентифікації з розмежуванням доступу, інструменти управління контентом,

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

механізми соціальної взаємодії (коментування, створення колекцій) та систему персоналізованих сповіщень. Завершальним етапом є комплексне тестування основних потоків виконання системи на основі розроблених тестових випадків для забезпечення її якості та надійності.

Об'єктом дослідження є процес створення та функціонування інтерактивних вебплатформ для систематизації та поширення цифрового медіаконтенту.

Предметом дослідження є технології та архітектурні підходи для розробки повнофункціональних вебзастосунків, зокрема використання фулстек-фреймворку Next.js, системи управління базами даних PostgreSQL та ORM Prisma для створення масштабованої та орієнтованої на користувача платформи для систематизації та подання контенту.

У перспективі розроблена архітектура може бути адаптована для створення спеціалізованих ресурсів, присвячених іншим формам графічної літератури, як-от західні комікси чи вебтуни. Крім того, закладений фундамент дозволяє розширювати функціонал шляхом інтеграції з платформами для команд перекладачів або впровадженням систем глибинної аналітики для генерації більш точних рекомендацій.

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1. Постановка задачі

Метою кваліфікаційної роботи бакалавра є створення інтерактивної веб-платформи для перегляду, публікації та обговорення манги. Вебсистема має об'єднувати шанувальників жанру, авторів та адміністрацію, забезпечуючи зручний доступ до цифрової бібліотеки манги, з можливістю інтерактивної взаємодії між користувачами, рекомендаціями та автоматичним інформуванням про оновлення манги.

Манга — це специфічний стиль японських коміксів, що поєднує графічне мистецтво з глибокими сюжетними лініями [1]. Вона охоплює широкий спектр жанрів — від пригодницьких і фантастичних історій до романтики, драми та філософських роздумів. Іншими словами, манга — це унікальний спосіб художнього вираження, який поєднує візуальне і літературне мистецтво. Її читають мільйони людей у всьому світі, і вона стала невіддільною частиною глобальної популярної культури. Манга вирізняється характерним стилем малювання, побудовою діалогів і структурою сторінок, що має чіткі відмінності від західних коміксів.

Історія сучасної манги почалася після Другої світової війни. У 1940-х роках японський художник Осаму Тедзука створив серію Astro Boy, яка стала основоположною для формування жанру у тому вигляді, в якому ми знаємо його сьогодні [2]. Завдяки кінематографічному стилю подачі та акценту на емоціях персонажів, його роботи мали колосальний вплив на подальший розвиток індустрії. У XXI столітті манга набула глобального поширення. Наприклад, серія One Piece, автором якої є Ейїчиро Ода, стала найбільш продаваною мангою у світі, з понад 500 мільйонами проданих примірників [3].

Розробка системи енциклопедій манги базується на модульній архітектурі, що дозволяє забезпечити гнучкість, масштабованість та керованість системи. Запропонована структура передбачає поділ системи на кілька ключових функціональних модулів, кожен з яких відповідає за

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

реалізацію конкретних бізнес-вимог та взаємодіє з іншими через чітко визначені програмні інтерфейси (API) [4].

Основою взаємодії користувача із системою є модуль автентифікації та авторизації. Його першочерговим завданням є забезпечення контрольованого доступу до ресурсів платформи та захист даних. Модуль реалізує функціонал реєстрації нових облікових записів із валідацією введених даних та безпечний механізм входу для існуючих користувачів. Центральним елементом блоку є система розмежування прав доступу на основі ролей [5]. В рамках системи визначено три основні ролі: «Користувач», «Автор» та «Адміністратор». Кожній ролі присвоюється специфічний набір дозволів, що обмежує або розширює доступ до певних функцій та даних, забезпечуючи таким чином цілісність інформації та відповідність дій користувача від його повноважень.

Ключовим функціональним модулем, що реалізує основне призначення платформи, є модуль управління контентом та його перегляду. З боку авторів, цей модуль надає програмний інструментарій для завантаження, структурування та публікації цифрових творів (манги). Він підтримує організацію контенту за томами, а також додавання необхідних метаданих. З боку читачів, модуль надає інтерфейс для навігації по каталогу та безпосереднього перегляду контенту. Важливою складовою є інтерактивні елементи, зокрема системи рейтингування та коментування, які слугують механізмом збору зворотного зв'язку від аудиторії та є джерелом даних для інших підсистем, наприклад, для системи рекомендацій [6].

Для вирішення задачі ефективної навігації у великому масиві даних проєктується модуль пошуку та персоналізації контенту. Його функціонал поділяється на дві складові. Перша — це багатоаспектний пошук, що дозволяє користувачам виконувати запити за різними атрибутами: назвою, іменем автора, жанром, ключовими словами та іншими метаданими. Друга складова, — це система персоналізованих рекомендацій [7]. Вона функціонує на основі алгоритмів, що аналізують поведінкові патерни користувача: історію переглядів, виставлені оцінки, обрані жанри. Метою даного модуля є

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

підвищення показника виявлення релевантного контенту та, як наслідок, збільшення тривалості користувацьких сесій та рівня залученості аудиторії.

З метою підтримки довготривалої взаємодії з користувачами та їх своєчасного інформування впроваджується підсистема сповіщень. Цей модуль функціонує як механізм асинхронної доставки повідомлень про значущі події, що відбуваються на платформі. Користувачі мають можливість підписуватися на оновлення конкретних творів. При настанні відповідної події, наприклад, публікації нового тому, система генерує та доставляє сповіщення у персональний кабінет користувача. Цей інструмент відіграє критичну роль в утриманні користувачів, стимулюючи їх до регулярних повторних візитів на платформу.

Завершальним, але не менш важливим компонентом архітектури є адміністративна панель (модуль адміністрування). Цей блок є захищеною зоною системи, призначеною виключно для персоналу з рівнем доступу адміністратор та обмеженим доступом для авторів (лише взаємодія з мангою). Він надає вичерпний набір інструментів для забезпечення операційного контролю, модерації контенту та управління спільнотою. До його функцій належить модерація завантажених творів та управління життєвим циклом облікових записів користувачів (включно з блокуванням порушників). Таким чином, даний модуль є гарантом підтримки порядку, якості контенту та загальної стабільності функціонування програмного комплексу.

В результаті повинна бути створена вебплатформа, яка надасть користувачам можливість не лише переглядати та публікувати мангу, але й взаємодіяти з контентом через коментарі, оцінки та рецензії. Важливим аспектом є інтеграція функцій для авторів, що дозволяє публікувати нові роботи, а також можливість користувачів отримувати сповіщення про вихід нових глав чи томів улюблених серій.

Процес створення складної інтерактивної платформи, якою є вебсайт-енциклопедія манги, являє собою багатоетапну діяльність, що вимагає послідовного та системного підходу. Життєвий цикл проєкту охоплює низку

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

ключових фаз, починаючи від проєктування і закінчуючи тестуванням та впровадженням готового продукту. Кожен з цих етапів є критично важливим для досягнення кінцевої мети — створення функціонального, стабільного та зручного у використанні веб-застосунку, що задовольняє потреби цільової аудиторії: як читачів, так і авторів контенту.

Першим етапом у реалізації проєкту є розробка дизайну — це комплексний процес проєктування користувацького досвіду (UX) та користувацького інтерфейсу (UI) [8]. Робота починається з аналізу та формування загальної концепції. На основі цих досліджень створюється візуальна концепція платформи: підбирається кольорова палітра, шрифти, стилістика іконок та ілюстрацій. Далі починається детальна розробка UI-компонентів - це елементи інтерфейсу - кнопки, поля вводу, картки манги, модальні вікна, - які мають бути не лише привабливими, але й інтуїтивно зрозумілими. Особлива увага приділяється дизайну сторінок, призначених для безпосереднього перегляду контенту (рідери), де головним пріоритетом є мінімізація відволікаючих елементів та забезпечення максимального комфорту для очей під час тривалого читання. Результатом цього етапу є набір макетів та інтерактивних прототипів.

Після затвердження візуальної концепції та макетів розпочинається етап побудови архітектури клієнтської частини системи, або ж фронтенду. Це процес перетворення статичних зображень на повноцінний інтерактивний веб-застосунок, з яким безпосередньо взаємодіє користувач. Основою цього етапу є розробка логічної структури сторінок та компонентів. Створюється система роутінгу, що забезпечує плавні та логічні переходи між різними розділами сайту: головною сторінкою, каталогом манги, профілем користувача, сторінками окремих творів та томів. Ключовим викликом тут є оптимізація інтерфейсу для різних пристроїв. Застосовуючи принципи адаптивної верстки, платформа буде однаково коректно і зручно відображатися як на широкоформатних моніторах персональних комп'ютерів, так і на компактних

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

екранах смартфонів і планшетів [9]. Це забезпечує безшовний користувацький досвід незалежно від того, яким пристроєм користується людина.

Паралельно з розробкою клієнтської частини ведеться робота над побудовою архітектури серверної частини, або бекенду [10, 11]. Цей етап включає розробку всієї серверної інфраструктури, яка відповідає за обробку запитів від клієнтської частини, реалізацію бізнес-логіки та взаємодію з базою даних. Одним із центральних завдань є проєктування та реалізація програмного інтерфейсу застосунку (API). Саме через API фронтенд надсилає запити на отримання списку манги, збереження коментаря чи автентифікацію користувача, а бекенд повертає відповідні дані у структурованому форматі. Надійність, безпека та швидкість відповіді цього API є критичними для загальної продуктивності платформи. Правильно спроектована серверна архітектура дозволяє ефективно обробляти багато одночасних запитів та забезпечує можливість подальшого масштабування системи при зростанні навантаження.

Невід'ємною складовою серверної логіки є розробка системи авторизації та управління ролями користувачів. Цей функціонал є критичним для безпеки платформи та для надання різним категоріям користувачів відповідних можливостей. На даному етапі створюється механізм реєстрації, що включає валідацію даних та підтвердження облікового запису, а також захищена процедура авторизації, яка використовує сучасні методи хешування паролів для їх безпечного зберігання [12]. Після успішної автентифікації система ідентифікує роль користувача — «користувач», «автор» чи «адміністратор». На основі цієї ролі система контролю доступу на обмежує або дозволяє виконання певних дій. Наприклад, лише користувач з роллю «автор» може завантажувати нові глави, а лише «адміністратор» — видаляти коментарі чи блокувати користувачів.

Коли базові архітектурні компоненти готові, починається наймасштабніший етап — розробка основного функціоналу. Створюються механізми для публікації манги, її поділу на томи, а також інтерфейси для

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

зручного перегляду. Одночасно проєктується та створюється база даних, яка є централізованим сховищем усієї інформації. Реалізується система коментування та оцінювання контенту, що дозволяє користувачам активно взаємодіяти з платформою та один з одним. Важливою частиною цього етапу є інтеграція системи сповіщень, яка інформує користувачів про вихід нових томів улюблених манг, що значно підвищує залученість аудиторії.

Завершальним, але безперервним протягом усього циклу розробки, є етап тестування системи. Цей процес спрямований на виявлення та усунення помилок, дефектів та вразливостей. Проводиться функціональне тестування, під час якого ретельно перевіряється коректність роботи всіх функцій: від реєстрації та входу до завантаження та перегляду манги [13]. Окремо виконується тестування графічного інтерфейсу, мета якого - переконатися, що всі елементи дизайну відображаються правильно на різних пристроях та у різних браузерах, а взаємодія з ними є зручною та передбачуваною. Лише після успішного проходження всіх циклів тестування та виправлення знайдених недоліків можна вважати платформу готовою до запуску.

Результатом виконання плану дій є створення вебплатформи, яка надасть можливість користувачам переглядати, оцінювати та коментувати мангу, а також відстежувати нові публікації. Автори зможуть публікувати свої твори, а платформа надасть інструменти для ефективного пошуку та рекомендацій. Адміністратори матимуть змогу управляти контентом та забезпечувати зручну модерацію. У підсумку, платформа дозволяє створювати персоналізовані профілі, отримувати сповіщення через особистий кабінет і взаємодіяти з іншими фанатами манги, надаючи доступ до безпечної та сучасної системи для обміну контентом та враженнями.

1.2. Аналіз аналогів програмного продукту

Аналіз аналогів програмного продукту є важливим етапом при розробці платформи для читання та публікації манги. Для цього було проаналізовано кілька популярних онлайн-ресурсів, що надають можливість читання манги, таких як Honey Manga, Zenko Online, Manga.in.ua, bato.to, mangaplaza. Ці

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

платформи мають схожі функціональні можливості та орієнтовані на різних користувачів, проте є певні відмінності, які важливо враховувати при розробці власного продукту.

Одним з таких додатків є платформа Honey Manga (рис. 1.1).

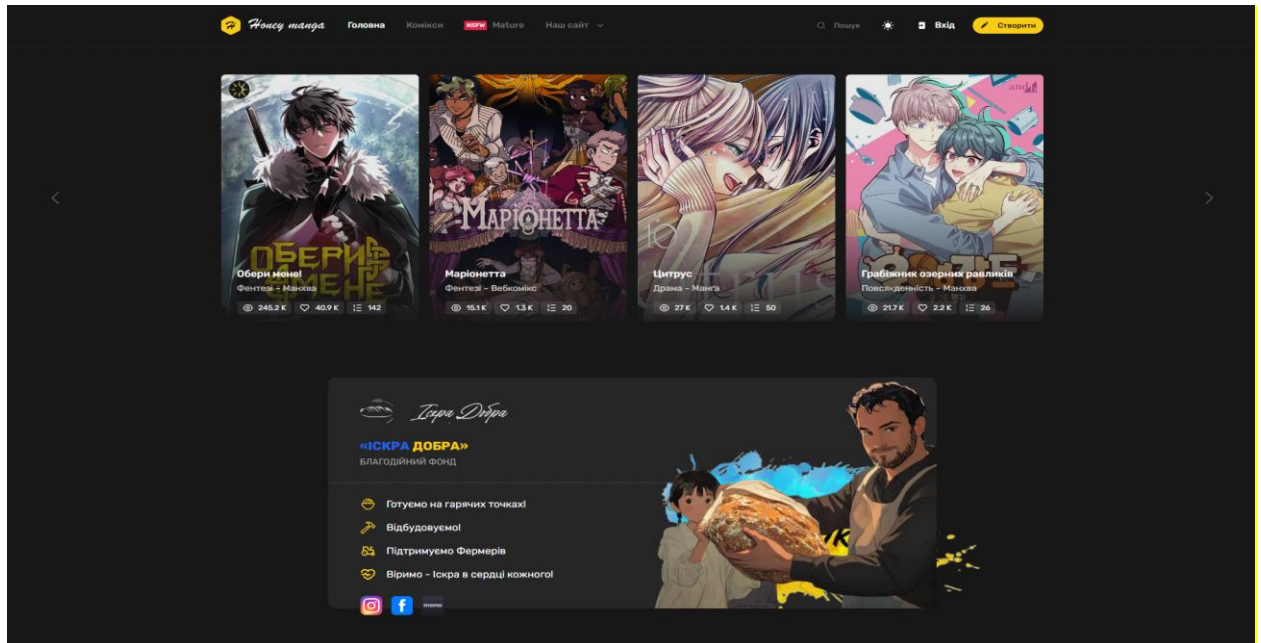


Рис. 1.1. Платформа Honey Manga

Платформа Honey Manga є гарним прикладом ресурсу, орієнтованого на українськомовних користувачів [14]. До її безперечних переваг належать зручний та інтуїтивно зрозумілий інтерфейс, безкоштовний доступ та можливість завантажувати контент для офлайн-читання. Однак для активного читача цей досвід швидко стає неповним. Головним недоліком є відсутність будь-якої системи сповіщень про вихід нових томів, що змушує користувачів самостійно та регулярно перевіряти оновлення улюблених серій. До того ж, орієнтація виключно на українську мову, хоч і є перевагою для частини аудиторії, водночас стає суттєвим обмеженням для тих, хто бажає читати мангу в оригіналі чи іншими мовами, змушуючи їх шукати альтернативні ресурси.

Zenko Online виглядає більш просунутим завдяки добре розробленому дизайну та наявності інтерактивних функцій, як-от коментарі та рейтинги (рис. 1.2) [15]. Це створює відчуття спільноти та дозволяє обмінюватися думками. Проте і ця платформа має суттєві вади. Найбільш відчутним недоліком є

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

відсутність автоматизованої системи сповіщень; навіть маючи акаунт, користувач не отримує жодних повідомлень про вихід нових розділів манги, яку він читає. Також, хоча платформа підтримує контент різними мовами, ця функція не є централізованою: користувачеві доводиться шукати різні версії тайтлу як окремі записи, замість того щоб мати зручний перемикач мов у рамках однієї сторінки манги, що є вкрай незручним.

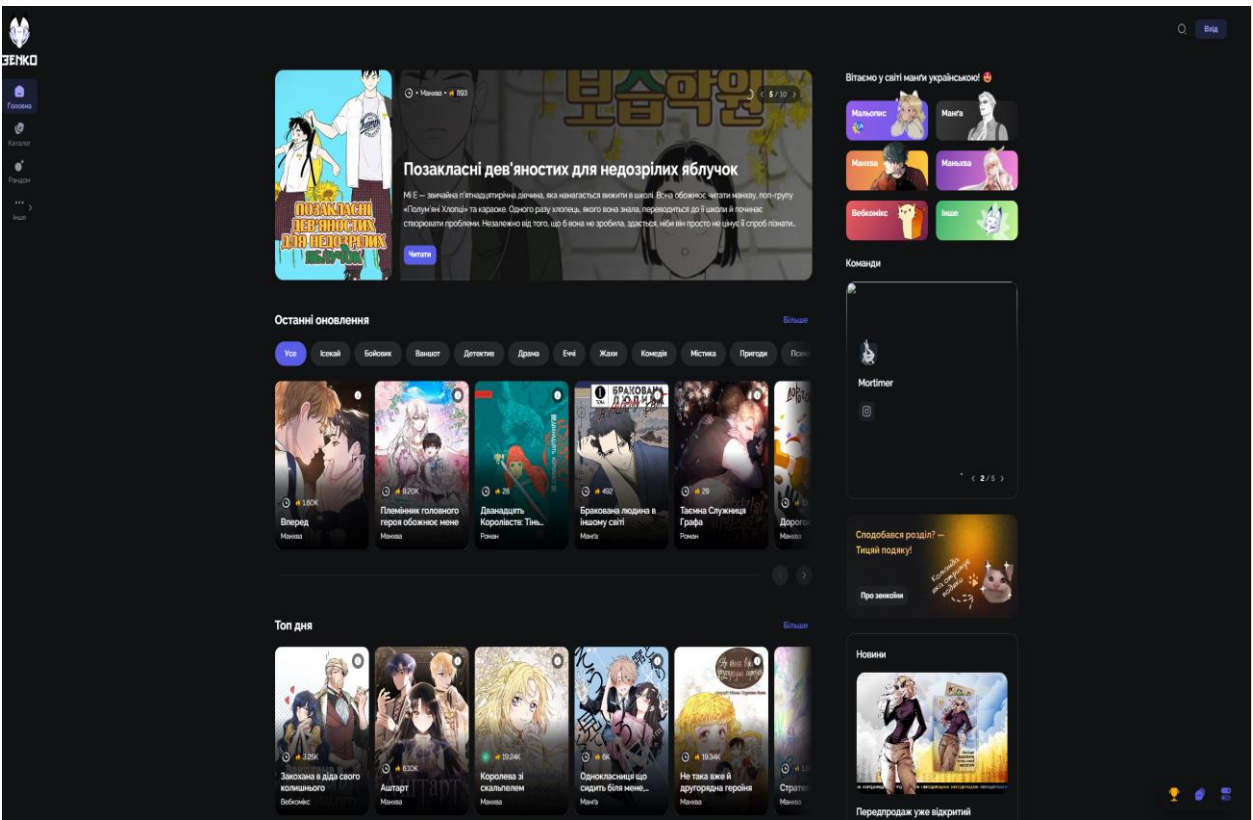


Рис. 1.2. Платформа Zenko Online

Ресурс Manga.in.ua також користується популярністю в Україні завдяки великому вибору манги рідною мовою та простому інтерфейсу (рис. 1.3) [16]. Платформа пропонує безкоштовний доступ та регулярно оновлює свою бібліотеку, що є її сильною стороною. Разом з тим, вона повторює ключові недоліки конкурентів. Тут повністю відсутня система сповіщень, через що відстеження прогресу та виходу нових глав повністю лягає на плечі читача. Це є значною незручністю для тих, хто стежить за десятками онгоїнгів. Крім того, сувора сфокусованість на українському перекладі так само обмежує аудиторію і не надає можливості читати мангу іншими мовами.

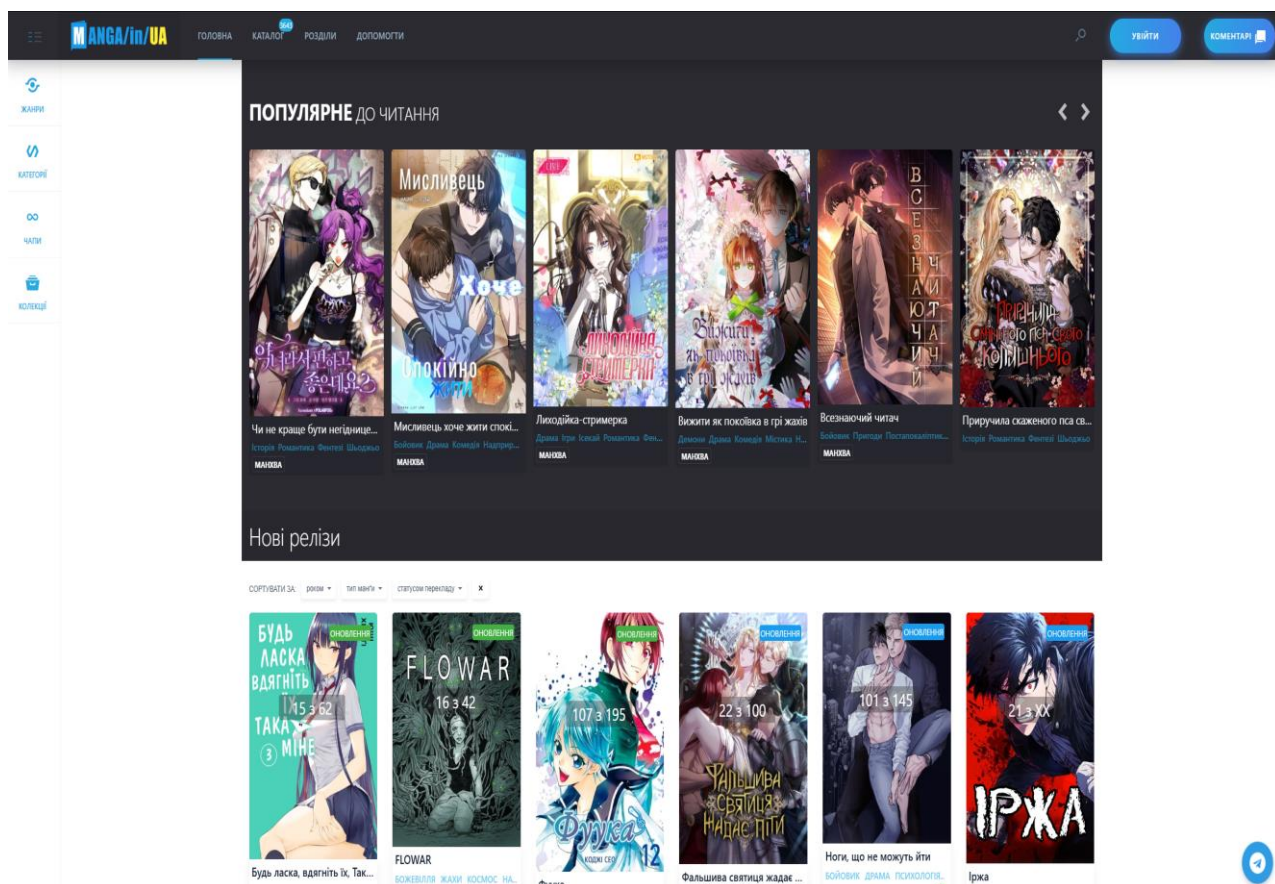


Рис. 1.3. Платформа Manga.in.ua

Bato.to є одним із найбільших міжнародних агрегаторів, перевагою якого є гігантська бібліотека та наявність перекладів від численних команд перекладачів (рис. 1.4) [17]. Проте цей масштаб породжує серйозні проблеми. На сайті відсутня ефективна система сповіщень, яка б активно інформувала читача про нові томи. Головним же недоліком є хаотична подача мультимовного контенту. Різні переклади одного й того ж тайтлу існують як десятки окремих, неупорядкованих записів. Це змушує користувача самотійно пробиратися крізь десятки одних і тих самих манг, замість того щоб запропонувати єдину сторінку з простим і зручним перемиканням між мовами.

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

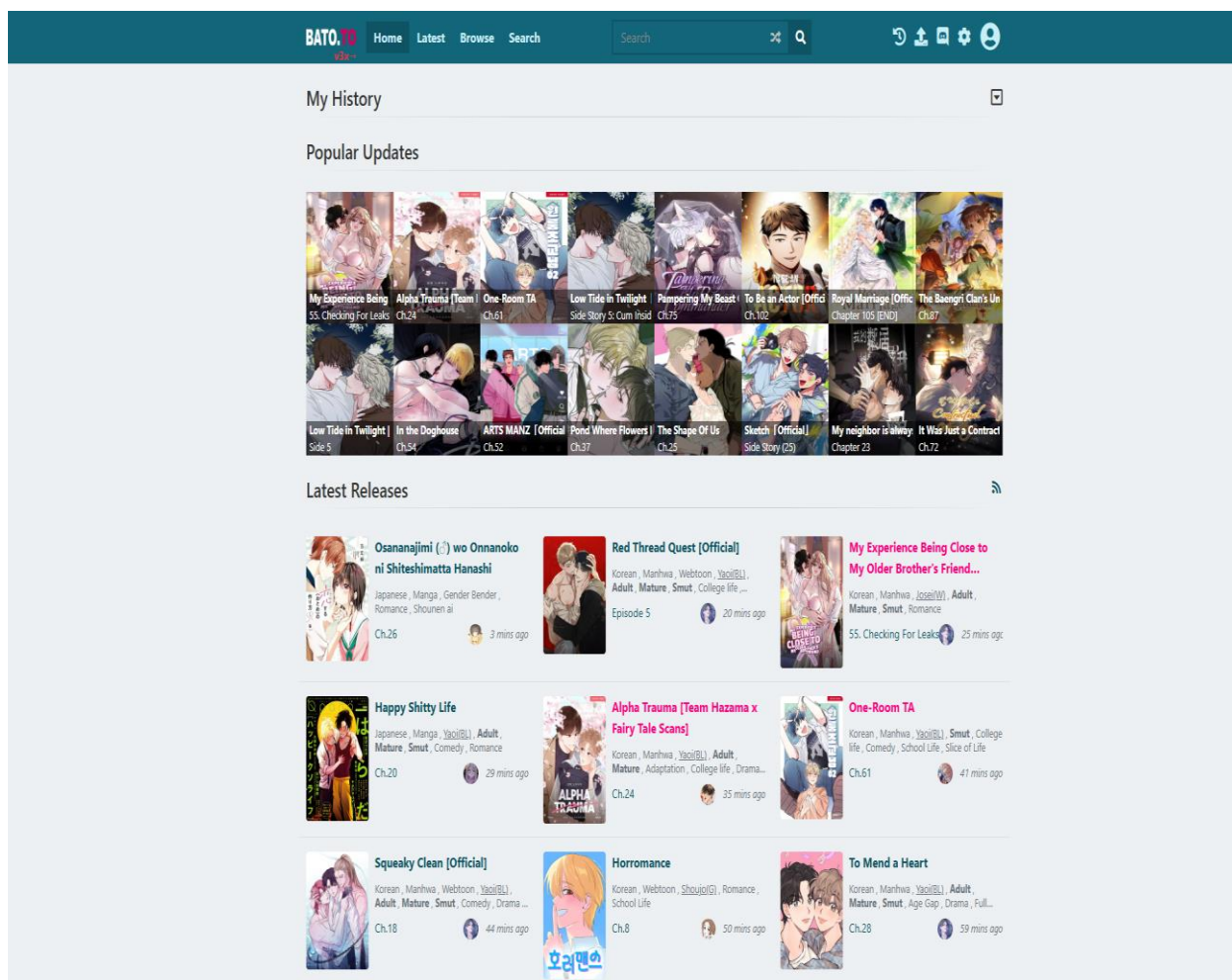


Рис. 1.4. Платформа Bato.to

На противагу агрегаторам, Mangarplaza є офіційною платформою, що гарантує високу якість перекладів та легальність контенту (рис.1.5) [18]. Її перевага — професійний підхід та підтримка авторів. Однак ці плюси нівелюються значними недоліками для широкої аудиторії. Система сповіщень, якщо і існує, то зазвичай обмежується загальними маркетинговими розсилками, а не персоналізованими повідомленнями про вихід глав улюбленої манги. Але найбільш критичним обмеженням є її суворі мономовність. Платформа орієнтована виключно на англomовний ринок, і можливість читати ту саму мангу українською чи будь-якою іншою мовою на сайті просто відсутня, що робить її непридатною для поліглотів та міжнародної аудиторії.

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		
						18

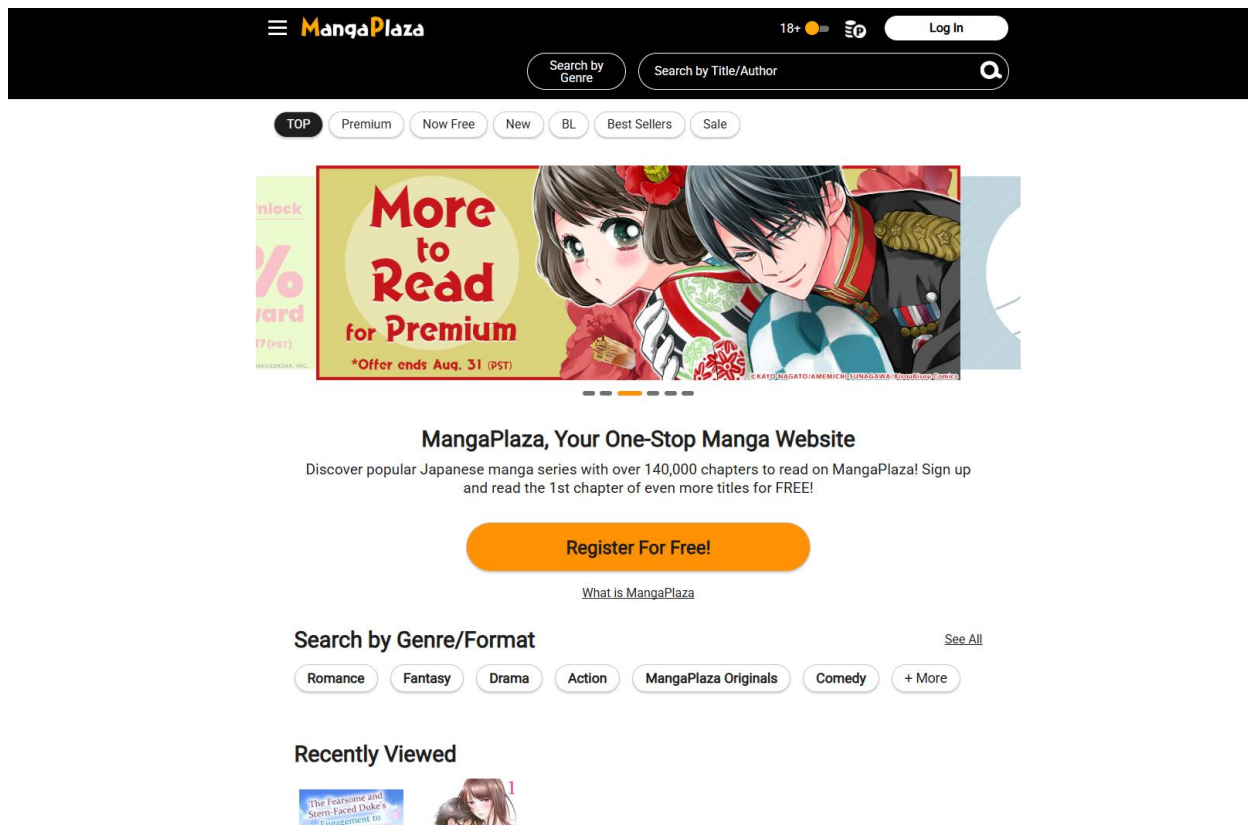


Рис. 1.5. Платформа Mangaplaza

Таблиця 1.1

Таблиця порівняння платформ для перегляду манги

Характеристика	Honey Manga	Zenko Online	Manga.in.ua	Bato.to	Mangaplaza	CEM
Підтримка української мови	+	-	+	+/-	-	+
Інтерфейс та зручність	+	+	+	+/-	+/-	+
Інтерактивні функції (коментарі, рейтинги)	-	+	-	+	+/-	+
Система сповіщень про оновлення манги	-	-	-	-	-	+
Централізована багатомовність	-	-	-	-	-	+
Завантаження для офлайн-читання	+	-	-	-	-	-

Проведений аналіз провідних платформ, таких як Honey Manga, Zenko Online, Manga.in.ua, Bato.to та Mangaplaza, дозволив виявити спільні риси сучасного ринку. Більшість із них пропонують користувачам зручний інтерфейс, доступ до великих бібліотек контенту та базові інтерактивні елементи, як-от коментарі чи рейтинги. Проте за цією зовнішньою функціональністю ховається низка системних недоліків, що створюють значні незручності для активних читачів.

Ключова проблема полягає у відсутності двох критично важливих, орієнтованих на користувача функцій. По-перше, жоден із ресурсів не має ефективної системи автоматичних сповіщень, яка повідомляє про вихід нових глав улюбленої манги, перекладаючи обов'язок постійного моніторингу на самого читача. По-друге, не вирішено проблему багатомовності: платформи або суворо обмежені однією мовою, або пропонують різні переклади в хаотичному та незручному для навігації вигляді.

Враховуючи недоліки, було прийнято рішення про створення платформи «Система енциклопедій манга». Її мета — не просто об'єднати переваги існуючих аналогів, а й запропонувати принципово новий рівень комфорту, усунувши їхні головні слабкі місця. В основі нашого продукту лежать дві інновації: система сповіщень, що звільняє читача від рутинного відстеження, та централізований мультимовний функціонал для зручного перемикання між мовними версіями манги на одній сторінці. У поєднанні з розширеними можливостями для взаємодії та публікації власного контенту, наша платформа покликана стати комплексним та найбільш зручним рішенням для сучасної спільноти поціновувачів манги.

1.3. Обґрунтування вибору інструментальних засобів

Розробка сучасної, високопродуктивної та масштабованої вебплатформи, якою є «Система енциклопедій манга», вимагає ретельного підходу до вибору технологічного стеку. Архітектура програмного продукту базується на сучасних принципах комплексної розробки, де єдине середовище та мова програмування, а саме TypeScript, використовуються для реалізації як

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

клієнтської, так і серверної складових [19]. Такий уніфікований підхід значно підвищує швидкість розробки, спрощує подальшу підтримку кодової бази та покращує взаємодію в команді розробників.

Після детального аналізу вимог проєкту було прийнято рішення зупинитися на зв'язці інструментальних засобів, що включає фулстек-фреймворк Next.js, реляційну систему управління базами даних PostgreSQL та інструмент для об'єктно-реляційного відображення (ORM) Prisma. Цей набір технологій було обрано через їхню виняткову синергію, високу продуктивність, значну зручність та ефективність процесу розробки, а також широкі можливості для майбутнього масштабування. Кожен компонент цього стеку вирішує специфічні завдання, а їхня спільна інтеграція дозволяє створити надійний, швидкий та функціональний продукт, що повністю відповідає поставленим цілям.

Хоча бібліотека React є фундаментальною для створення користувацьких інтерфейсів, для побудови повноцінного продукту, як «Система енциклопедій манга», потрібен більш комплексний та структурований інструмент [20]. Next.js є виробничим фреймворком, побудованим на базі React, що розширює його можливості та надає безліч критично важливих функцій, інтегрованих за замовчуванням [21, 22].

Ключовою перевагою Next.js є його гнучкість у виборі стратегій відображення сторінок, що дозволяє оптимізувати кожен частину додатка для досягнення високої продуктивності та індексації пошуковими системами. Зокрема, рендеринг на стороні сервера (SSR) є оптимальним для динамічних сторінок, вміст яких залежить від даних конкретного користувача або часто змінюється, наприклад, для сторінок профілю, списків улюблених творів чи історії читання [23]. Завдяки SSR, HTML-сторінка генерується на сервері при кожному запиті, що гарантує миттєве відображення актуального контенту та забезпечує чудову індексацію пошуковими системами. Сторінки з мангою будуть легко знаходитись у пошукових системах, що є критичним для залучення нової аудиторії.

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

Next.js використовує файлову систему для побудови маршрутизації, де створення нової сторінки зводиться до створення файлу у відповідній директорії, що є інтуїтивно зрозумілим і значно прискорює розробку. Фреймворк також стандартно інтегрує механізми оптимізації, такі як автоматичне розділення коду, що дозволяє завантажувати лише той програмний код, який необхідний для конкретної сторінки. Це суттєво зменшує початковий час завантаження додатка. Для проєкту «Система енциклопедій манга», де будуть використовуватись тисячі зображень, вбудований компонент оптимізації зображень забезпечує їх лінійне завантаження, зміну розміру та конвертацію у сучасні формати, що кардинально покращує швидкість завантаження сторінок. Крім того, Next.js надає можливість створювати повноцінні серверні ендпоінти для програмного інтерфейсу в межах того ж проєкту. Це усуває необхідність у розгортанні та підтримці окремого серверного додатку, оскільки вся логіка для взаємодії з базою даних, обробки запитів на сповіщення та аутентифікації користувачів може бути реалізована безпосередньо в Next.js, що спрощує архітектуру та забезпечує безшовну інтеграцію між клієнтською та серверною частинами.

Для зберігання даних проєкту, що включають інформацію про користувачів, величезний каталог манги, томи, коментарі, рейтинги та складні зв'язки між ними, необхідна надійна, гнучка та масштабована база даних. PostgreSQL, як об'єктно-реляційна система управління базами даних з відкритим вихідним кодом, є одним із кращих рішень у своїй галузі та ідеально відповідає цим вимогам [24].

Головною перевагою PostgreSQL є її безкомпромісна надійність. Повна підтримка транзакцій та відповідність принципам ACID (атомарність, узгодженість, ізольованість, довговічність) гарантує цілісність даних [25]. У контексті нашого проєкту це означає, що операції, як-от додавання манги до списку улюблених користувача або збереження прогресу читання, будуть виконані коректно, навіть у випадку системних збоїв чи високого навантаження, забезпечуючи збереження даних користувачів.

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

PostgreSQL славиться своєю розширюваністю. Для проєкту є корисною підтримка таких типів даних, як JSONB, що дозволяє зберігати гнучкі, неструктуровані дані (наприклад, список альтернативних назв манги чи перелік персонажів) в одній колонці та при цьому ефективно їх індексувати [26]. Крім того, PostgreSQL має потужні вбудовані можливості для повнотекстового пошуку. Це дозволить реалізувати швидкий та релевантний пошук по каталогу манги, враховуючи морфологію та можливі помилки в назвах. Можливість встановлення спеціалізованих розширень робить її ідеальним інструментом для створення інтелектуальної пошукової системи.

Платформа «Система енциклопедій манга» розрахована на постійне зростання як кількості користувачів, так і обсягу контенту. PostgreSQL чудово зарекомендувала себе у роботі з великими обсягами даних та високими навантаженнями. Вона надає широкий спектр інструментів для оптимізації продуктивності, включаючи різні типи індексів, можливість партиціонування великих таблиць та механізми реплікації для розподілу навантаження та забезпечення високої доступності системи.

Для ефективного поєднання потужності Next.js та надійності PostgreSQL потрібен сучасний інструмент доступу до даних. Prisma — це інструмент для об'єктно-реляційного відображення нового покоління, який вирішує багато проблем традиційних ORM та забезпечує виняткову зручність розробки завдяки своїй типобезпечності [27].

Ключовою перевагою Prisma є забезпечення повної узгодженості типів даних на всьому шляху від бази даних до клієнтської частини. На відміну від традиційних підходів, де типи даних часто доводиться визначати вручну, Prisma автоматично генерує клієнт TypeScript на основі декларативної схеми бази даних. Це означає, що будь-який запит до бази даних повертає повністю типізований об'єкт. Якщо в схемі бази даних змінюється поле, компілятор TypeScript миттєво виявить усі місця в коді, які потребують оновлення. Такий підхід усуває цілий клас потенційних помилок під час виконання програми, значно підвищує надійність коду та прискорює його рефакторинг.

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

Prisma використовує єдиний файл схеми, що слугує читабельним та інтуїтивно зрозумілим описом моделі даних і зв'язків між таблицями. Він є центральним джерелом інформації про архітектуру бази даних, що спрощує роботу. На основі цієї схеми Prisma керує міграціями бази даних — процесом внесення змін до її структури. Інструмент Prisma автоматично генерує SQL-скрипти для міграції, що дозволяє легко змінювати схеми разом із кодом додатку. Сгенерований Prisma Client надає простий, але водночас потужний програмний інтерфейс для виконання операцій з даними. Складні запити з фільтрацією, сортуванням та включенням пов'язаних даних записуються лаконічно та читабельно, що не тільки підвищує продуктивність розробника, але й робить код більш зрозумілим та легким у підтримці.

Висновки до першого розділу

Вибір технологічного стеку Next.js, PostgreSQL та Prisma є обґрунтованим рішенням, що забезпечує ідеальний баланс між продуктивністю, надійністю та ефективністю розробки. Next.js надає гнучку та потужну основу для створення швидких, оптимізованих для пошукових систем веб-інтерфейсів та інтегрованої серверної частини. PostgreSQL служить надійним фундаментом для зберігання та обробки великих обсягів складних даних, гарантуючи їх цілісність та безпеку. Prisma виступає ідеальним мостом між додатком та базою даних, забезпечуючи неперевершену типобезпечність та значно спрощуючи роботу з даними. Разом ці інструменти створюють ідеальну синергію, дозволяючи побудувати високоякісний, масштабований та легкий у підтримці програмний продукт «Система енциклопедій манга», що зможе задовольнити потреби користувачів та успішно розвиватися в довгостроковій перспективі.

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ ЕНЦИКЛОПЕДІЙ МАНГА

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи

Для забезпечення повноцінного функціоналу роботи з мангою, система включає модулі управління каталогом творів, аналізу читацьких вподобань, обробки рецензій та автоматизації рекомендацій, з можливістю формувати персоналізовані підбірки. Також передбачено зберігання даних про твори, авторів, жанри, томи та історію переглядів, що спрощує пошук контенту, покращує взаємодію з користувачами та автоматизує процес створення унікального читацького досвіду.

Внутрішні користувачі мають різні обов'язки відповідно до їх ролей, що приведено нижче (табл. 2.1).

Таблиця 2.1

Внутрішні користувачі

Актор	Короткий опис
Користувач	Користувач системи, який може переглядати та читати мангу, залишати коментарі та рецензії.
Адміністратор	Відповідає за управління системою, додавання нової манги, томів і глав, а також модерацію коментарів і користувачів.
Автор	Користувач який може публікувати манги, має обмежений доступ до адмін-панелі в якій може лише додавати та редагувати мангу.

На (рис. 2.1) зображено побудовану діаграму базових варіантів використання відповідно до кожної ролі.

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

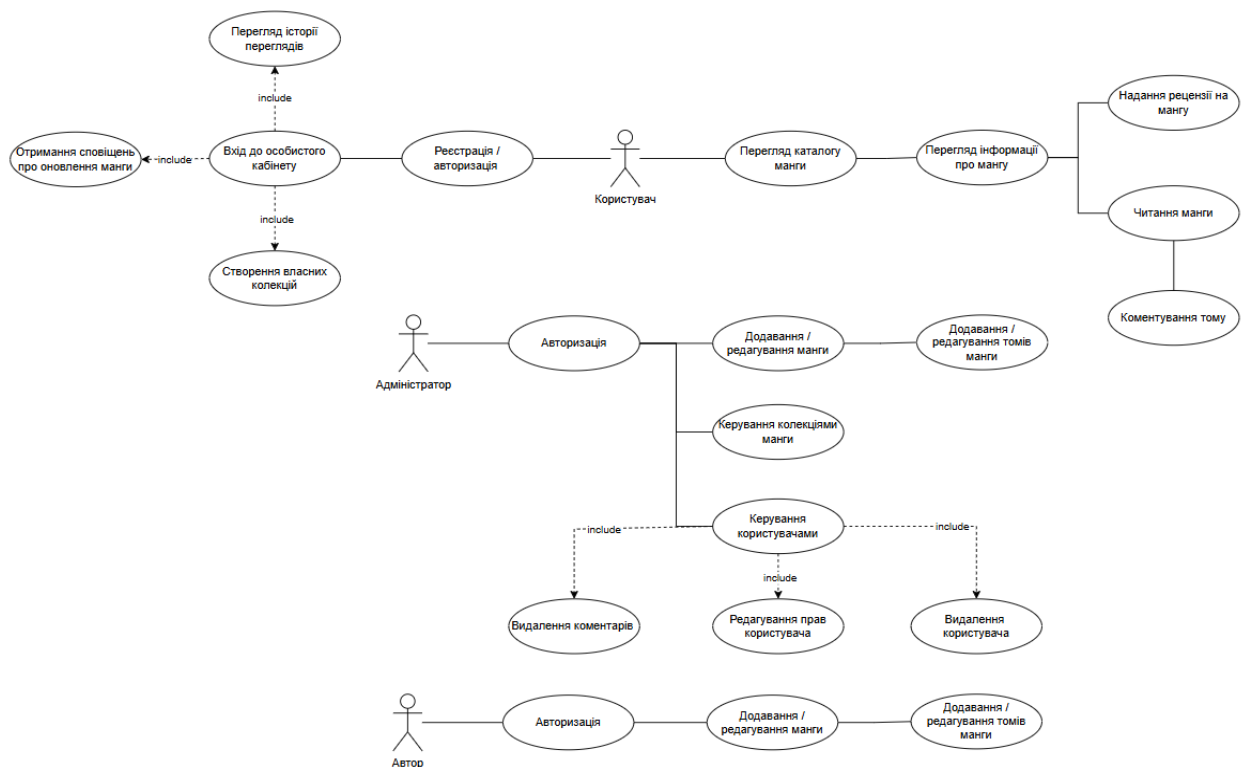


Рис 2.1 Діаграма базових варіантів використання

Специфікація основних прецедентів для кожної ролі приведена в додатку А, узагальнені функції користувачів відповідно до їх ролей представлені в таблиці 2.2.

Таблиця 2.2

Основні функції користувачів

Роль	Основні функції
Користувач	• Перегляд каталогу манг з фільтрацією за тегами. • Додавання творів до особистих колекцій. • Публікація рецензій на манги. • Отримання персоналізованих рекомендацій. • Взаємодія з іншими користувачами через коментарі. • Отримання сповіщень про оновлення улюбленої манги.
Адміністратор	• Додавання нових творів до каталогу. • Редагування інформації про манги, томи та теги. • Модерування коментарів, рецензій та блокування користувачів. • Створення глобальних колекцій.
Автор	• Додавання нових творів до каталогу. • Редагування інформації про власні манги, томи та теги.

На основі визначених функцій були сформульовані детальні функціональні та нефункціональні вимоги до програмного забезпечення (таблиця 2.3).

Таблиця 2.3

Вимоги до ПЗ

Тип вимоги	Опис
Функціональні вимоги	• Управління контентом: Додавання, перегляд, фільтрування та редагування інформації про мангу та її томи. • Модерація: Можливість видалення коментарів та блокування користувачів для дотримання правил спільноти. • Персоналізація: Генерування рекомендацій на основі історії переглядів та вподобань. • Сповіщення: Формування сповіщень про вихід або оновлення томів манги. • Особистий кабінет: Управління профілем, перегляд історії, створення власних колекцій манг.
Нефункціональні вимоги	• Простота використання: Час навчання для нових користувачів — до 30 хв, для адміністраторів — до 1 год. • Продуктивність: Час реакції на стандартні запити — до 1 сек; на складні — до 3 сек. • Надійність: Стабільна робота при одночасній активності до 1000 користувачів; доступність бази даних 99.9%; не більше 1 критичної помилки на 5000 запитів. • Системні вимоги: ОП для клієнта — не менше 2 ГБ; для сервера — від 8 ГБ. • Масштабованість: Підтримка обробки понад 10 000 творів. • Безпека: Захист даних через шифрування та автентифікацію. • Сумісність: Підтримка сучасних браузерів (Chrome, Firefox, Safari, Edge) та адаптація для мобільних пристроїв.

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

Можемо виділити наступні сутності, які зображені на діаграмі нижче, виходячи з аналізу вимог функціональності, що забезпечують реалізацію системи енциклопедій манг. Об'єктно-орієнтована структура системи у вигляді діаграми класів приведена нижче (рис. 2.2), яка відображає зв'язки між сутностями [28].

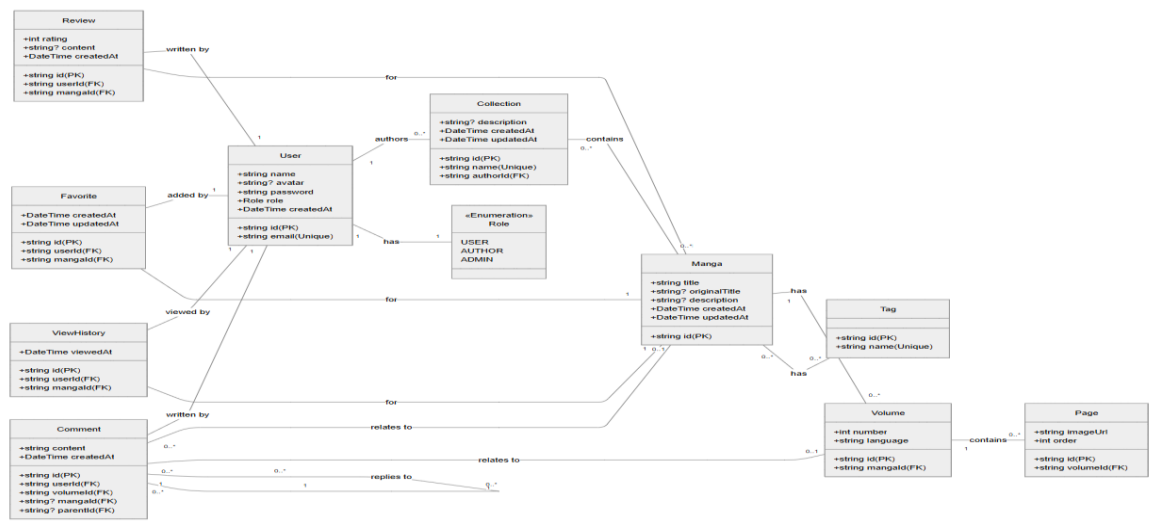


Рис 2.2 Діаграма класів

Представлена діаграма класів описує модель даних, в центрі якої знаходяться дві основні сутності: User та Manga. Сутність User моделює користувача системи, містить поля для ідентифікації, такі як email та password, а також пов'язана відношенням "один-до-одного" з перерахуванням Role, яке визначає тип користувача.

Сутність Manga виступає як основний об'єкт контенту, що характеризується атрибутами title та description. Структура контенту є композиційною: діаграма показує, що Manga має зв'язок "один-до-багатьох" з сутністю Volume, яка, в свою чергу, аналогічно пов'язана з Page. Це відображає ієрархію, де манга складається з томів, а томи — зі сторінок.

Для категоризації контенту реалізовано зв'язок "багато-до-багатьох" між сутностями Manga та Tag. Це дозволяє одній манзі присвоювати декілька тегів, а одному тегу відповідати багатьом творам. Подібним чином організовано зв'язок між Collection та Manga, що дає змогу групувати різні твори у довільні колекції. При цьому сама Collection має зв'язок "один-до-багатьох" з User, що позначає авторство колекції.

Взаємодія між користувачами та контентом моделюється через окремі асоціативні класи. Сутності Review, Favorite та ViewHistory слугують для зв'язку між User та Manga, фіксуючи відповідно відгуки, додавання до обраного та історію переглядів. Кожна з цих сутностей має власні атрибути, що деталізують взаємодію, наприклад, rating у Review.

Найбільш складною є сутність Comment. Діаграма ілюструє, що коментар завжди пов'язаний з одним User (автором), але може відноситись або до Manga в цілому, або до конкретного Volume. Крім того, наявність рекурсивного зв'язку "один-до-багатьох" на самій сутності Comment моделює можливість ієрархічних відповідей, створюючи гілки обговорень.

2.2 Розробка бази даних системи

Дані в системі зберігаються в базі даних PostgreSQL, яка складається з 12 таблиць. Ці таблиці забезпечують зберігання всієї інформації про користувачів, контент та їхню взаємодію, що є необхідним для виконання основного функціоналу системи (табл. 2.4).

Таблиця 2.4

Загальний опис таблиць бази даних.

Назва таблиці	Призначення
User	Таблиця для зберігання даних користувачів.
Manga	Таблиця, що містить інформацію про мангу.
Tag	Довідник тегів для категоризації манги.
MangaTag	Сполучна таблиця для зв'язку між мангою та тегами.
Volume	Таблиця для зберігання томів, певної манги.
Page	Таблиця сторінок, що входять до складу томів.
Favorite	Таблиця для зберігання обраних робіт користувачів.
Review	Таблиця для зберігання відгуків та рейтингів користувачів.
Comment	Таблиця для зберігання коментарів до контенту.
ViewHistory	Таблиця для відстеження історії переглядів манги користувачами.
Collection	Таблиця для зберігання користувацьких колекцій.
CollectionManga	Сполучна таблиця для зв'язку між колекціями та мангою.

Структура бази даних висвітлена на (рис 2.3).

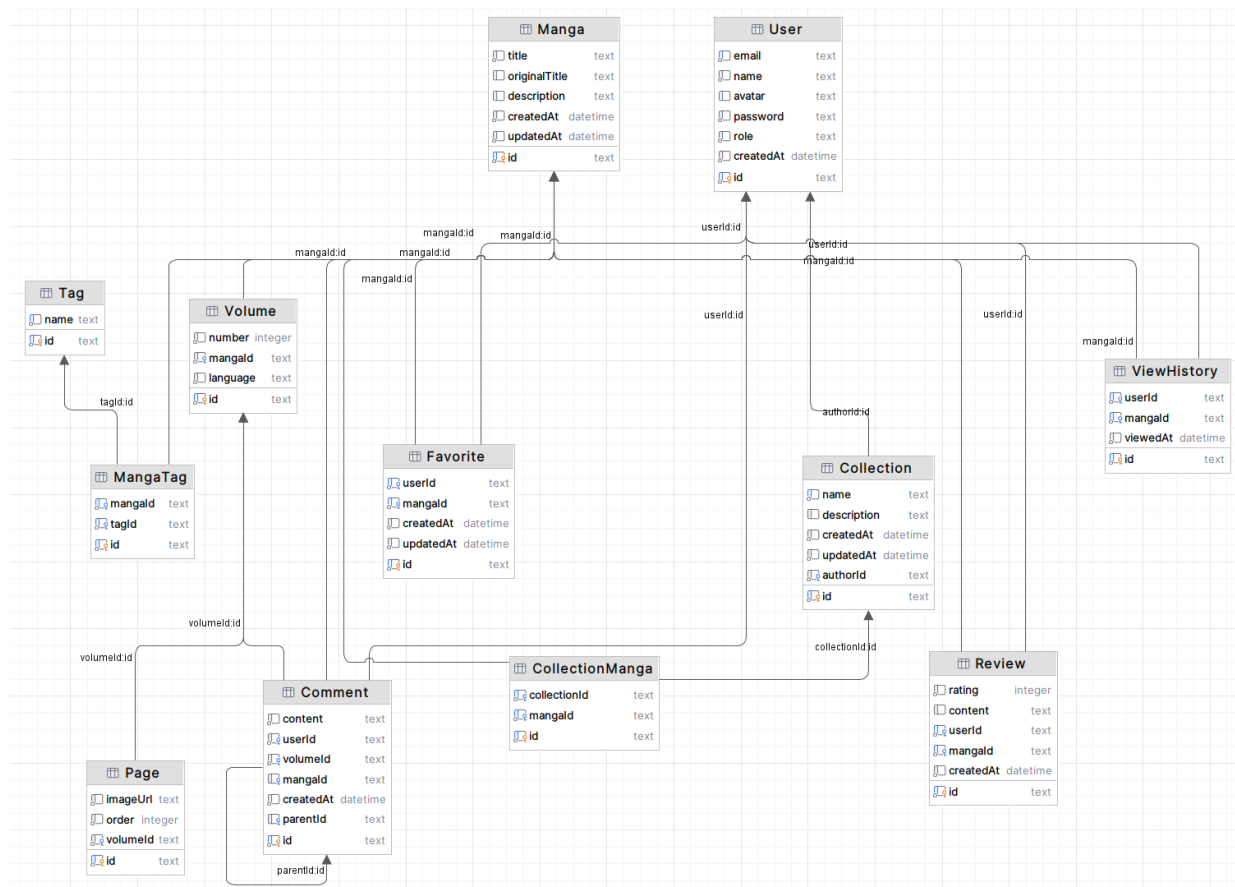


Рис 2.3 Структура бази даних

Таблиця «User» слугує для збереження даних користувача. Опис полів таблиці наведений нижче (табл. 2.5).

Таблиця 2.5

Опис таблиці «User»

Назва	Тип даних	PK	FK	Опис поля
id	TEXT	+	-	Унікальний ідентифікатор користувача.
email	TEXT	-	-	Електронна пошта користувача (унікальне значення).
name	TEXT	-	-	Ім'я користувача.
avatar	TEXT	-	-	Аватар користувача.
password	TEXT	-	-	Захешований пароль користувача.
role	TEXT	-	-	Роль користувача в системі.
createdAt	TIMESTAMP	-	-	Час створення запису про користувача.

Таблиця «Manga» містить основну інформацію про твори манги. Опис полів таблиці наведений нижче (табл. 2.6).

Таблиця 2.6

Опис таблиці «Manga»

Назва	Тип даних	PK	FK	Опис поля
id	TEXT	+	-	Унікальний ідентифікатор манги.
title	TEXT	-	-	Основна назва манги.
originalTitle	TEXT	-	-	Оригінальна назва манги (може бути NULL).
description	TEXT	-	-	Опис або синопсис манги (може бути NULL).
createdAt	TIMESTAMP	-	-	Час створення запису.
updatedAt	TIMESTAMP	-	-	Час останнього оновлення запису.

Таблиця «Tag» є довідником тегів. Опис полів таблиці наведений нижче (табл. 2.7).

Таблиця 2.7

Опис таблиці «Tag»

Назва	Тип даних	PK	FK	Опис поля
id	TEXT	+	-	Унікальний ідентифікатор тега (CUID).
name	TEXT	-	-	Назва тега (унікальне значення).

Таблиця «MangaTag» є сполучною таблицею для реалізації зв'язку "багато-до-багатьох" між мангою та тегами (табл. 2.8).

Таблиця 2.8

Опис таблиці «MangaTag»

Назва	Тип даних	PK	FK	Опис поля
id	TEXT	+	-	Унікальний ідентифікатор запису.
mangaId	TEXT	-	+	Зовнішній ключ, що посилається на id в таблиці Manga.
tagId	TEXT	-	+	Зовнішній ключ, що посилається на id в таблиці Tag.

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

Таблиця «Volume» зберігає інформацію про томи манги (табл. 2.9).

Таблиця 2.9

Опис таблиці «Volume»

Назва	Тип даних	PK	FK	Опис поля
id	TEXT	+	-	Унікальний ідентифікатор тому (CUID).
number	INTEGER	-	-	Порядковий номер тому.
mangaId	TEXT	-	+	Зовнішній ключ, що посилається на id в таблиці Manga.
language	TEXT	-	-	Мова тому (за замовчуванням "UA").

Таблиця «Page» зберігає посилання на зображення сторінок (табл. 2.10).

Таблиця 2.10

Опис таблиці «Page»

Назва	Тип даних	PK	FK	Опис поля
id	TEXT	+	-	Унікальний ідентифікатор сторінки.
imageUrl	TEXT	-	-	URL-адреса зображення сторінки.
order	INTEGER	-	-	Порядковий номер сторінки в томі.
volumeId	TEXT	-	+	Зовнішній ключ, що посилається на id в таблиці Volume.

Таблиця «Favorite» фіксує, яку мангу користувачі додали до списку обраного (табл. 2.11).

Таблиця 2.11

Опис таблиці «Favorite»

Назва	Тип даних	PK	FK	Опис поля
id	TEXT	+	-	Унікальний ідентифікатор запису.
userId	TEXT	-	+	Зовнішній ключ, що посилається на id в таблиці User.
mangaId	TEXT	-	+	Зовнішній ключ, що посилається на id в таблиці Manga.
createdAt	TIMESTAMP	-	-	Час додавання до обраного.
updatedAt	TIMESTAMP	-	-	Час оновлення запису.

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

Таблиця «Review» зберігає відгуки та рейтинги користувачів (табл. 2.12).

Таблиця 2.12

Опис таблиці «Review»

Назва	Тип даних	PK	FK	Опис поля
id	TEXT	+	-	Унікальний ідентифікатор відгука.
rating	INTEGER	-	-	Числова оцінка (рейтинг).
content	TEXT	-	-	Текстовий вміст відгука (може бути NULL).
userId	TEXT	-	+	Зовнішній ключ, що посилається на id в таблиці User.
mangaId	TEXT	-	+	Зовнішній ключ, що посилається на id в таблиці Manga.
createdAt	TIMESTAMP	-	-	Час створення відгука.

Таблиця «Comment» зберігає коментарі користувачів (табл. 2.13).

Таблиця 2.13

Опис таблиці «Comment»

Назва	Тип даних	PK	FK	Опис поля
id	TEXT	+	-	Унікальний ідентифікатор коментаря.
content	TEXT	-	-	Текстовий вміст коментаря.
userId	TEXT	-	+	Зовнішній ключ на автора коментаря (User).
volumeId	TEXT	-	+	Зовнішній ключ на том (Volume), до якого залишено коментар.
mangaId	TEXT	-	+	Зовнішній ключ на мангу (Manga), до якої залишено коментар (може бути NULL).
parentId	TEXT	-	+	Зовнішній ключ на батьківський коментар для реалізації відповідей (може бути NULL).
createdAt	TIMESTAMP	-	-	Час створення коментаря.

Таблиця «ViewHistory» відстежує історію переглядів (табл. 2.14).

Таблиця 2.14

Опис таблиці «ViewHistory»

Назва	Тип даних	РК	ФК	Опис поля
id	TEXT	+	-	Унікальний ідентифікатор запису історії.
userId	TEXT	-	+	Зовнішній ключ на користувача (User).
mangaId	TEXT	-	+	Зовнішній ключ на мангу (Manga).
viewedAt	TIMESTAMP	-	-	Час останнього перегляду.

Таблиця «Collection» зберігає дані про користувацькі колекції (табл. 2.15).

Таблиця 2.15

Опис таблиці «Collection»

Назва	Тип даних	РК	ФК	Опис поля
id	TEXT	+	-	Унікальний ідентифікатор колекції.
name	TEXT	-	-	Назва колекції (унікальне значення).
description	TEXT	-	-	Опис колекції (може бути NULL).
authorId	TEXT	-	+	Зовнішній ключ на автора колекції (User).
createdAt	TIMESTAMP	-	-	Час створення колекції.
updatedAt	TIMESTAMP	-	-	Час останнього оновлення.

Таблиця «CollectionManga» є сполучною для зв'язку "багато-до-багатьох" між колекціями та мангою (табл. 2.16).

Опис таблиці «CollectionManga»

Назва	Тип даних	PK	FK	Опис поля
id	TEXT	+	-	Унікальний ідентифікатор запису.
collectionId	TEXT	-	+	Зовнішній ключ, що посилається на id в таблиці Collection.
mangaId	TEXT	-	+	Зовнішній ключ, що посилається на id в таблиці Manga.

Спроектowana реляційна структура бази даних є комплексним та масштабованим рішенням, що дозволяє побудувати багатофункціональну платформу, орієнтовану на спільноту читачів манги [29]. Структура бази даних поділяється на три основні логічні блоки: управління основним контентом, організація та категоризація даних, а також забезпечення соціальної взаємодії та персоналізації користувацького досвіду.

Перший блок, що базується на таблицях User, Manga, Volume та Page, реалізує ядро системи. Завдяки ієрархічній структурі, де манга складається з томів, а томи — зі сторінок, досягається ефективне зберігання та логічна організація контенту. Таблиця User слугує основою для автентифікації, авторизації та розмежування прав доступу.

Другий блок значно розширює можливості системи, виходячи за межі простого зберігання даних. Завдяки таблицям Tag та MangaTag реалізовано гнучку систему класифікації, що дозволяє здійснювати глибоку категоризацію контенту та забезпечує ефективні механізми пошуку і фільтрації. У свою чергу, таблиці Collection та CollectionManga надають користувачам інструменти для кураторської діяльності — створення персоналізованих добірок. Це не лише покращує навігацію, але й стимулює активність спільноти.

Третій, соціально-орієнтований блок, є ключовим для утримання аудиторії. Таблиці Review, Favorite, ViewHistory та Comment забезпечують персоналізацію та соціальну взаємодію. Таблиця ViewHistory є основою для

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

реалізації функціоналу "продовжити читання" та для майбутніх систем персоналізованих рекомендацій.

Таким чином, розроблена структура бази даних є цілісною та логічною. Послідовне використання первинних та зовнішніх ключів забезпечує високий рівень цілісності даних. Модульність схеми дозволяє в майбутньому легко розширювати функціонал, додаючи нові сутності без необхідності кардинальної перебудови існуючих зв'язків. Система ефективно інтегрує управління контентом, інструменти для його організації та потужні механізми соціальної взаємодії, що є запорукою для побудови успішного онлайн-проєкту.

2.3 Проєктування та реалізація алгоритмів роботи системи

Оскільки для розробки програмного додатку було обрано фреймворк Next.js, архітектура системи є інтегрованою, що поєднує в собі як клієнтську, так і серверну логіку в єдиному проєкті. Цей підхід дозволяє створювати швидкі та сучасні веб-застосунки, використовуючи рендеринг як на стороні сервера, так і на стороні клієнта.

Комунікація між клієнтською та серверною частинами реалізована за допомогою API Routes - вбудованого механізму Next.js. Цей підхід функціонує аналогічно до традиційного RESTful API та виступає інтерфейсом для взаємодії з даними.

Основні принципи роботи API в даному проєкті полягають у наступному. По-перше, він базується на концепції ресурсів, де кожен тип даних, наприклад, користувачі, манга чи коментарі, має унікальну URL-адресу. По-друге, API використовує стандартні HTTP-методи для виконання операцій над цими ресурсами [30]. Метод GET застосовується для отримання даних. Для створення нового об'єкта використовується метод POST. Оновлення інформації реалізовано через методи PUT/PATCH, а для видалення використовується метод DELETE.

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

Таким чином, клієнтська частина, що складається з компонентів, написаних на React, взаємодіє з серверною логікою через уніфікований інтерфейс HTTP-запитів, отримуючи дані у форматі JSON.

Процес обробки запиту в системі відбувається за наступним алгоритмом. Коли користувач виконує дію в інтерфейсі, клієнтська частина надсилає HTTP-запит до відповідного API Route. На стороні сервера цей маршрут отримує запит і, використовуючи Prisma Client, формує запит до бази даних PostgreSQL. Prisma виступає як ORM, що трансліює виклики методів у мові TypeScript у безпечні SQL-запити. База даних виконує запит, повертає результат, який Prisma передає назад до API Route. Серверна логіка обробляє ці дані та повертає клієнту відповідь у вигляді JSON-об'єкта, на основі якого оновлюється користувацький інтерфейс.

Також, для зображення алгоритму роботи додатку розроблено структурну діаграму (рис 2.4), де зображені основні потоки виконання логіки для користувачів з певними ролями.

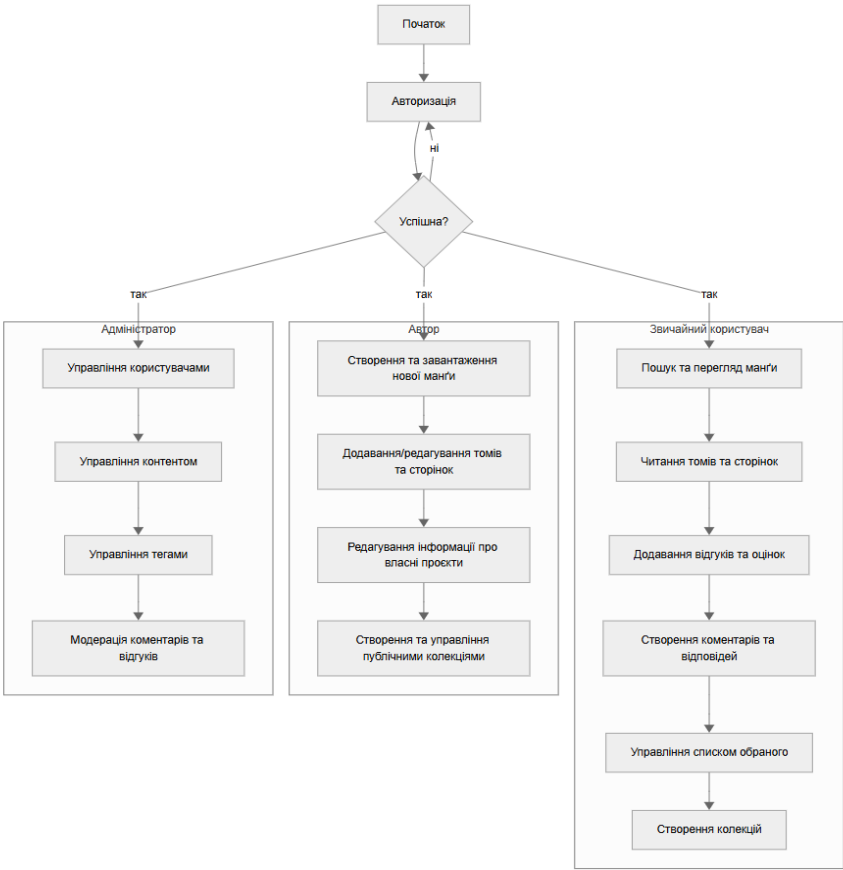


Рис 2.4 Структурна діаграма системи

На діаграмі ми бачимо, що робота системи починається з етапу авторизації. У випадку успішного входу користувач переходить до використання системи, а в разі невдачі - повертається до початку.

Успішна авторизація надає користувачеві доступ до функціоналу, визначеного його роллю. Звичайні користувачі можуть шукати й читати контент, залишати відгуки та керувати своїми списками. Авторам доступне створення та редагування власних проєктів і колекцій. Адміністратор, у свою чергу, отримує повні права на управління всіма аспектами системи: користувачами, контентом, тегами та модерацією.

Робота користувача в межах сесії завершується виконанням необхідних дій та, в кінцевому підсумку, виходом із системи.

Безпосередньо застосунок можна умовно поділити на певні модулі. Для цього побудуємо діаграму компонентів (рис 2.5) додатку для декількох модулів.

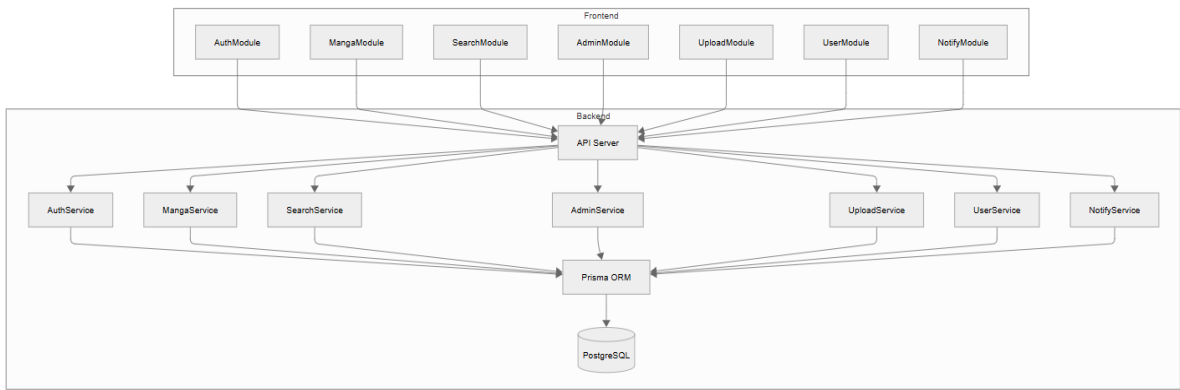


Рис 2.5 Діаграма компонентів додатку

Діаграма компонентів відображає архітектуру системи, що складається з двох основних частин: Frontend та Backend. Frontend є модульним і містить окремі компоненти для автентифікації, управління мангою, пошуку, адміністрування та інших функцій. Усі ці модулі взаємодіють з Backend через відповідні Арі роути.

На стороні Backend, API роути приймають запити від клієнта і направляють їх до відповідних сервісів, кожен з яких відповідає за свою частину бізнес-логіки. Усі сервіси використовують Prisma ORM для взаємодії

з базою даних PostgreSQL, що забезпечує централізований та стандартизований доступ до даних.

Детальніше розглянемо алгоритм роботи модуля керування мангою. Для цього було побудовано діаграму послідовності (рис. 2.6) [31].

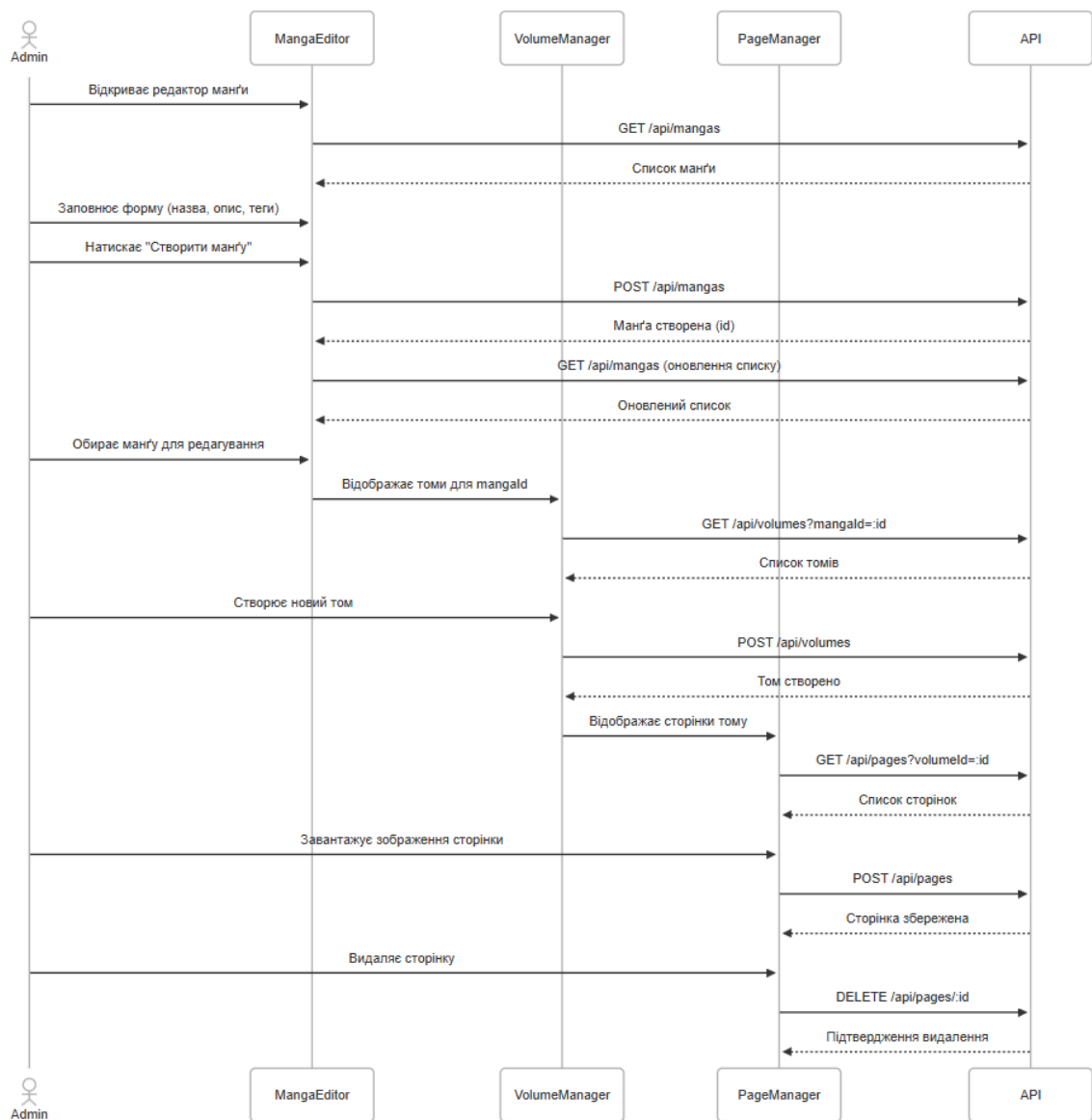


Рис 2.6 Діаграма послідовності для модуля керування мангою

Ця діаграма ілюструє послідовність дій адміністратора під час створення та наповнення контенту на платформі. Процес починається, коли адміністратор заходить до редактора манги. У відповідь система одразу ж звертається до API, щоб завантажити та відобразити повний перелік наявних робіт.

Адміністратор може створити новий твір, заповнивши необхідні дані, такі як назва, опис та теги. Після підтвердження цієї дії редактор надсилає запит на створення нової манги до API, і після успішного збереження оновлює список, щоб відобразити щойно додану роботу.

Подальша робота відбувається в контексті конкретної манги. Коли адміністратор обирає один із творів, система переходить до управління його томами. На цьому етапі завантажується і показується список томів, що належать саме цій манзі. Адміністратор може додати новий том, і ця зміна так само зберігається через API.

Аналогічний процес відбувається і на рівні сторінок: після вибору тому завантажуються всі його сторінки. Адміністратор може додавати нові зображення, які зберігаються через API, або ж видаляти непотрібні сторінки, надсилаючи відповідний запит на видалення. Таким чином, діаграма демонструє покроковий ієрархічний процес управління контентом: від створення загальної сутності манги до детального редагування її окремих сторінок.

2.4 Реалізація системи енциклопедій манга

Розглянемо бізнес логіку одного з модулів системи, а саме модуля керування мангою. Його архітектура побудована на трьох взаємопов'язаних компонентах, що відповідають за різні рівні ієрархії: MangaEditor для управління мангою, VolumeManager для роботи з томами, і PageManager для управління сторінками. Розглянемо логіку роботи цього модуля поетапно, детально аналізуючи кожен крок.

Процес починається з компонента MangaEditor, який слугує основною точкою входу для адміністратора. При першому завантаженні цей компонент ініціює запит до серверу для отримання списку всіх існуючих робіт, що дозволяє адміністратору одразу побачити поточний стан контенту. Основна логіка реалізована через стан React, де зберігається форма для створення чи редагування, поточний режим роботи та список завантажених творів.

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

Робота починається з ініціалізації стану форми в компоненті MangaEditor.jsx (рис.2.7). Для зберігання даних нової манги використовується об'єкт form, що містить поля для назви, оригінальної назви, опису та масиву тегів.

```
const [form, setForm] = useState({
  title: '',
  originalTitle: '',
  description: '',
  tags: [],
});
```

Рис 2.7 Форма для створення нової манги

Коли адміністратор заповнює форму і натискає кнопку "Зберегти", викликається асинхронна функція. Ця функція збирає дані з об'єкта form, перетворює їх у формат JSON і надсилає на сервер за допомогою POST-запита на ендпоінт /api/mangas (рис. 2.8).

```
const handleAdd = async () => {
  setLoading(true);
  const res = await fetch('/api/mangas', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(form),
  });
  if (res.ok) {
    const created = await res.json();
    setMangas(m => [created, ...m]);
    reset();
  } else {
    const err = await res.json();
    alert(err.error || 'Помилка');
  }
  setLoading(false);
};
```

Рис 2.8 Запит створення манги

Наступний етап - обробка запиту на сервері. На сервері запит приймається відповідним API Route. Першим кроком є валідація отриманих даних, це є базовим захистом від створення неповних даних [32].

Далі, використовуючи Prisma, виконується операція prisma.manga.create. Робота з тегами реалізована так: операція connectOrCreate вкладеного запиту дозволяє уникнути дублювання. Після успішного створення запису в базі

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

даних, сервер повертає клієнту JSON-об'єкт щойно створеної манги зі статусом 201 Created (рис. 2.9).

```
export async function POST(req) {
  const { title, originalTitle, description, tags } = await req.json()

  if (!title.trim()) {
    return NextResponse.json({ error: 'Потрібен заголовок' }, { status: 400 })
  }

  try {
    const created = await prisma.manga.create({
      data: {
        title,
        originalTitle,
        description,
        tags: {
          create: (tags || []).map((name) => ({
            tag: {
              connectOrCreate: {
                where: { name },
                create: { name },
              },
            },
          )))
        },
      },
      include: {
        tags: {
          include: { tag: true },
        },
      },
    })

    // Удобный формат: возвращаем просто массив имён тегов
    const result = {
      id: created.id,
      title: created.title,
      originalTitle: created.originalTitle,
      description: created.description,
      createdAt: created.createdAt,
      updatedAt: created.updatedAt,
      tags: created.tags.map((mt) => mt.tag.name),
    }

    return NextResponse.json(result, { status: 201 })
  } catch (err) {
    console.error('[POST /api/mangas]', err)
    return NextResponse.json({ error: 'Не вдалося створити мангу' }, { status: 500 })
  }
}
```

Рис 2.9 Обработка запроса

Редагування є не менш важливою операцією, що дозволяє оновлювати інформацію про твір. Коли адміністратор обирає існуючу мангу зі списку, клієнтський компонент MangaEditor переходить у режим редагування (рис. 2.10). Для цього ідентифікатор обраної манги зберігається у стані editId, а дані цієї манги завантажуються у той самий об'єкт стану form, який

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

використовувався для створення. Це дозволяє повторно використовувати ту саму форму як для створення, так і для редагування.

```
const handleEdit = manga => {
  setEditId(manga.id);
  setForm({
    title: manga.title,
    originalTitle: manga.originalTitle || '',
    description: manga.description || '',
    tags: manga.tags?.map(tag => tag) || [],
  });
  setMode('edit');
};
```

Рис 2.10 Режим редагування

Після внесення змін адміністратор натискає кнопку "Зберегти зміни". Це ініціює виклик асинхронної функції, яка надсилає PUT-запит на ендпоінт `/api/mangas/[id]`, де `[id]` — це ідентифікатор манги, що редагується. У тілі запиту передаються оновлені дані з форми у форматі JSON (рис 2.11).

```
const handleSave = async () => {
  setLoading(true);
  const res = await fetch(`/api/mangas/${editId}`, {
    method: 'PUT',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(form),
  });
  if (res.ok) {
    const updated = await res.json();
    setMangas(m => m.map(x => (x.id === updated.id ? updated : x)));
    reset();
  } else {
    const err = await res.json();
    alert(err.error || 'Помилка');
  }
  setLoading(false);
};
```

Рис 2.11 Запит на оновлення

На сервері PUT-запит обробляється відповідним динамічним маршрутом. Після оновлення запису в базі даних сервер повертає оновлений об'єкт клієнту (рис 2.12).

```

export async function PUT(req, { params }) {
  const { id } = await params;
  const { title, originalTitle, description, tags } = await req.json();
  if (!title.trim()) {
    return NextResponse.json({ error: 'Потрібен заголовок' }, { status: 400 });
  }
  try {
    const updated = await prisma.manga.update({
      where: { id },
      data: {
        title,
        originalTitle,
        description,
        tags: {
          deleteMany: {},
          create: (tags || []).map(name => ({
            tag: {
              connectOrCreate: {
                where: { name }, // шукаємо унікальний тег
                create: { name }, // або створюємо новий
              },
            },
          })),
        },
      },
      include: {
        tags: {
          include: {
            tag: true, // щоб в масиві tags було поле 'tag' з об'єктом Tag
          },
        },
      },
    });
    const result = {
      ...updated,
      tags: updated.tags.map(mt => mt.tag.name),
    };
    return NextResponse.json(result);
  } catch (err) {
    console.error(['PUT /api/mangas/:id'], err);
    return NextResponse.json({ error: 'Не вдалося оновити мангу' }, { status: 500 });
  }
}

```

Рис 2.12 Обробка запиту на оновлення

У режимі редагування адміністратору доступна кнопка "Видалити". Перед відправкою запиту на сервер система показує діалогове вікно для підтвердження дії, щоб запобігти випадковому видаленню. Після підтвердження надсилається DELETE-запит на той самий ендпоінт, що й для редагування (рис. 2.13).

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савицький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

```
const handleDelete = async () => {
  if (!confirm('Ви впевнені, що хочете видалити цю мангу?')) return;
  setLoading(true);
  const res = await fetch(`/api/mangas/${editId}`, {
    method: 'DELETE',
  });
  if (res.ok) {
    setMangas(m => m.filter(x => x.id !== editId));
    reset();
  } else {
    const err = await res.json();
    alert(err.error || 'Помилка видалення');
  }
  setLoading(false);
};
```

Рис 2.13 Запит на видалення

Обробка DELETE-запиту є значно складнішою, ніж просто видалення запису з бази даних. Оскільки з мангою пов'язані файли сторінок, що зберігаються у файловій системі, їх необхідно видалити, щоб уникнути накопичення "сміття".

Серверна логіка спочатку знаходить усі томи та сторінки, що належать манзі, яку видаляють. Далі вона збирає шляхи до всіх файлів зображень і послідовно видаляє їх з диска за допомогою модуля fs. Лише після успішного очищення файлової системи Prisma виконує операцію prisma.manga.delete. Завдяки налаштованим у схемі Prisma каскадним правилам, видалення запису про мангу автоматично спричинить видалення всіх пов'язаних з нею записів у інших таблицях (томів, сторінок, відгуків, коментарів тощо) (рис. 2.14).

```
export async function DELETE(req, { params }) {
  const { id } = await params;
  try {
    const volumes = await prisma.volume.findMany({
      where: { mangaId: id },
      select: {
        pages: {
          select: { imageUrl: true }
        }
      }
    });
    const imagePaths = volumes.flatMap(volume =>
      volume.pages.map(p => p.imageUrl)
    );
    await Promise.all(imagePaths.map(async (url) => {
      const filePath = path.join(process.cwd(), 'public', 'manga_pages', url); //
      try {
        fs.unlinkSync(filePath);
      } catch (err) {
        console.warn(`Файл не знайдено або не видалено: ${filePath}`);
      }
    }));
    await prisma.manga.delete({ where: { id } });
    return new Response(null, { status: 204 });
  } catch (err) {
    console.error('Помилка видалення манги:', err);
    return Response.json({ error: 'Не вдалося видалити мангу' }, { status: 500 });
  }
}
```

Рис 2.14 Обробка видалення манги

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

Після створення манги наступним кроком є додавання до неї томів. Цей функціонал знаходиться у компоненті VolumeManager.jsx. Компонент VolumeManager отримує mangaId як властивість від свого батьківського компонента MangaEditor. При першому рендерингу він використовує цей ID, щоб зробити GET-запит і завантажити список уже наявних томів для даної манги. Для створення нового тому ініціалізуються окремі стани для номера та мови (рис. 2.15).

```
const [volumes, setVolumes] = useState([]);
const [newNumber, setNewNumber] = useState('');
const [newLanguage, setNewLanguage] = useState('UA');
const [loading, setLoading] = useState(false);

const load = () => {
  fetch(`/api/mangas/${mangaId}/volumes`)
    .then(res => res.json())
    .then(setVolumes);
};
useEffect(load, [mangaId]);
```

Рис 2.15 Ініціалізація станів

Після введення номера та мови адміністратор ініціює створення тому. Клієнтська частина перетворює номер тому на ціле число, перевіряє, чи заповнені обидва поля, і надсилає POST-запит на відповідний ендпоінт, передаючи дані в тілі запиту (рис. 2.16)

```
const addVolume = async () => {
  const num = parseInt(newNumber, 10);
  if (!num || !newLanguage.trim()) return alert('Введіть номер  мову тому');
  setLoading(true);
  const res = await fetch(`/api/mangas/${mangaId}/volumes`, {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ number: num, language: newLanguage.trim() }),
  });
  if (res.ok) {
    setNewNumber('');
    setNewLanguage('UA');
    load();
  } else {
    const err = await res.json();
    alert(err.error || 'Помилка');
  }
  setLoading(false);
};
```

Рис 2.16 Ініціалізація станів

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				46
Змн.	Арк.	№ докум.	Підпис	Дата		

На сервері API Route приймає запит. Виконується кілька рівнів валідації. Лише після проходження всіх перевірок виконується `prisma.volume.create`, і запис про новий том додається до бази даних, пов'язаний із відповідною мангою (рис. 2.17).

```
export async function POST(request, { params }) {
  const { id } = await params;
  if (!id) {
    return NextResponse.json({ error: 'Немає айді манги' }, { status: 400 });
  }
  let body;
  try {
    body = await request.json();
  } catch {
    return NextResponse.json({ error: 'Невірний JSON' }, { status: 400 });
  }
  const { number, language } = body;
  if (typeof number !== 'number' || typeof language !== 'string') {
    return NextResponse.json(
      { error: 'Поля "number" (number) і "language" (string) обов'язкові' },
      { status: 400 }
    );
  }
  const mangaExists = await prisma.manga.findUnique({ where: { id } });
  if (!mangaExists) {
    return NextResponse.json(
      { error: `Манга з айді ${id} не знайдена` },
      { status: 404 }
    );
  }
  const exists = await prisma.volume.findFirst({
    where: { mangaId: id, number, language },
  });
  if (exists) {
    return NextResponse.json(
      { error: `Том ${number} із мовою "${language}" вже існує для манги ${id}` },
      { status: 400 }
    );
  }
  const volume = await prisma.volume.create({
    data: { mangaId: id, number, language },
  });
  await prisma.manga.update({
    where: { id },
    data: { updatedAt: new Date() },
  });
  return NextResponse.json(volume, { status: 201 });
}
```

Рис 2.17 Обробка запиту на сервері

Видалення тому, як і видалення манги, є незворотною операцією, що потребує очищення пов'язаних файлів. В інтерфейсі VolumeManager біля кожного тому є кнопка "Видалити". Після підтвердження дії клієнт надсилає DELETE-запит на динамічний ендпоінт (рис. 2.18)

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

```
const deleteVolume = async (id) => {
  if (!confirm('Видалити цей том?')) return;
  await fetch(`/api/mangas/${mangaId}/volumes/${id}`, { method: 'DELETE' });
  load();
};
```

Рис 2.18 Запит на видалення

Серверна логіка для видалення тому схожа на логіку видалення манги. Спочатку система знаходить усі сторінки, що належать цьому тому. Далі відбувається видалення відповідних файлів з диска. Після цього виконується `prisma.volume.delete`, що завдяки каскадним правилам автоматично видаляє всі пов'язані з томом записи (сторінки, коментарі до тому) з бази даних (рис. 2.19).

```
export async function DELETE(req, { params }) {
  const { volumeId } = params;
  try {
    const pages = await prisma.page.findMany({
      where: { volumeId },
      select: { imageUrl: true }
    });
    await Promise.all(
      pages.map(async ({ imageUrl }) => {
        const filePath = path.join(process.cwd(), 'public', 'manga_pages', imageUrl);
        try {
          fs.unlinkSync(filePath);
        } catch {
          console.warn('Не вдалося знайти або видалити файл: ${filePath}');
        }
      })
    );
    await prisma.volume.delete({
      where: { id: volumeId }
    });
    return NextResponse.json({ success: true }, { status: 200 });
  } catch (err) {
    console.error('Помилка видалення тому:', err);
    return NextResponse.json({ error: 'Не вдалося видалити том' }, { status: 500 });
  }
}
```

Рис 2.19 Обробка запиту на видалення

Завершальним етапом є наповнення томів сторінками. Цю відповідальність несе компонент `PageManager`. Компонент `PageManager` активується для конкретного тому і отримує `volumeId`. За аналогією з попередніми компонентами, він завантажує список вже існуючих сторінок. Для додавання нових сторінок використовується поле введення типу `file`, яке дозволяє множинний вибір. Обрані файли зберігаються у стані `newFiles`.

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

Процес завантаження сторінок є двохетапним. Спочатку кожен файл послідовно відправляється на спеціальний ендпоінт для завантаження файлів, де він зберігається на сервері. У відповідь сервер повертає унікальне ім'я файлу. Після цього клієнтська частина робить другий запит — POST, передаючи вже не сам файл, а отримане ім'я файлу (рис. 2.20).

```
const uploadNew = async () => {
  if (!volumeId) return;
  setLoading(true);
  try {
    for (const file of newFiles) {
      const filename = await handleFileUpload(file);
      const imageUrl = filename;
      const res = await fetch(
        `/api/mangas/${mangaId}/volumes/${volumeId}/pages`,
        {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify({ imageUrl }),
        }
      );
      if (!res.ok) {
        const err = await res.json().catch(() => ({}));
        throw new Error(err.error || 'Помилка збереження сторінки');
      }
      const created = await res.json();
      setPages(p => [...p, created]);
    }
    setNewFiles([]);
  } catch (e) {
    alert(e.message);
  } finally {
    setLoading(false);
  }
};
```

Рис 2.20 Запит на завантаження сторінок

На сервері API Route для створення сторінки отримує volumeId та imageUrl. Важливою частиною бізнес-логіки є автоматичне визначення порядкового номера сторінки (order). Для цього Prisma виконує агрегуючий запит, щоб знайти максимальне значення order серед існуючих сторінок даного тому. Новій сторінці присвоюється значення maxOrder + 1, що гарантує її розміщення в кінці списку (рис. 2.21).

```

export async function POST(req, { params }) {
  const { volumeId } = await params;
  const { imageUrl } = await req.json();
  const maxOrder = await prisma.page.aggregate({
    where: { volumeId },
    _max: { order: true },
  });
  const nextOrder = (maxOrder._max.order ?? 0) + 1;
  const page = await prisma.page.create({
    data: { volumeId, imageUrl, order: nextOrder },
  });
  return NextResponse.json(page, { status: 201 });
}

```

Рис 2.21 Обробка запиту на завантаження сторінок

Якщо ж користувач хоче видалити сторінку, користувач натискає кнопку "Видалити" біля конкретної сторінки. Після підтвердження клієнт надсилає DELETE-запит на ендпоінт, що включає ID сторінки (рис. 2.22).

```

const handleDelete = async (pageId) => {
  if (!confirm('Видалити сторінку?')) return;
  setLoading(true);
  try {
    const res = await fetch(
      `/api/mangas/${mangaId}/volumes/${volumeId}/pages/${pageId}`,
      { method: 'DELETE' }
    );
    if (!res.ok) {
      const err = await res.json().catch(() => ({}));
      throw new Error(err.error || 'Не вдалося видалити сторінку');
    }
    setPages(pages.filter(p => p.id !== pageId));
  } catch (e) {
    alert(e.message);
  } finally {
    setLoading(false);
  }
};

```

Рис 2.22 Запит на видалення сторінки

Серверна логіка знаходить запис про сторінку за її pageId, щоб отримати imageUrl. Після цього вона видаляє відповідний файл з диска. На завершення, запис про сторінку видаляється з бази даних (рис. 2.23).

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

```

export async function DELETE(request, context) {
  const { pageId } = await context.params;
  const page = await prisma.page.findUnique({ where: { id: pageId } });
  if (!page) {
    return NextResponse.json({ error: 'Такої сторінки немає' }, { status: 404 });
  }
  try {
    const filepath = path.join(
      process.cwd(),
      'public',
      'manga_pages',
      path.basename(page.imageUrl)
    );
    await fs.unlink(filepath);
  } catch {
  }
  await prisma.page.delete({ where: { id: pageId } });
  return new NextResponse(null, { status: 204 });
}

```

Рис 2.23 Обробка запиту на видалення сторінки

В висновку ми маємо те, що бізнес-логіка всього модуля публікації манги побудована за чіткою ієрархічною структурою, де кожен компонент вирішує своє вузькоспеціалізоване завдання. Взаємодія між клієнтом та сервером через чітко визначені API роутами та використанням Prisma ORM для безпечної роботи з базою даних, ця взаємодія створює надійну, модульну та масштабовану архітектуру.

Висновки до другого розділу

У другому розділі було проведено комплексне проєктування системи енциклопедій манга. На основі визначених ролей користувачів - «Користувач», «Автор» та «Адміністратор» - були розроблені ключові варіанти використання системи, що візуалізовано за допомогою діаграми прецедентів. Це дозволило детально визначити функціональні та нефункціональні вимоги, заклавши основу для подальшого проєктування архітектури.

На базі цих вимог була спроектована структура системи, представлена у вигляді діаграми класів, яка відображає основні сутності та зв'язки між ними. Відповідно до цієї структури, була розроблена реляційна модель бази даних для PostgreSQL, що складається з 12 таблиць. Ця модель забезпечує надійне зберігання даних про користувачів, мангу, томи, сторінки, а також

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

інформацію, необхідну для реалізації соціальної взаємодії та персоналізації, як-от коментарі, відгуки та колекції.

Також у розділі було описано архітектуру програмного продукту, побудовану на базі фреймворку Next.js, та принципи взаємодії між клієнтською та серверною частинами через API роути. Для ілюстрації логічних потоків роботи системи було створено структурну діаграму та діаграму компонентів. В останньому пункті було детально проаналізовано практичну реалізацію бізнес-логіки на прикладі модуля управління контентом. Були розглянуті ключові операції CRUD для сутностей "манга", "том" та "сторінка", що включають валідацію даних, взаємодію з базою даних через Prisma ORM та управління файловою системою.

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУ

3.1 Порядок встановлення та налаштування системи

Для наочного представлення фізичної архітектури системи використовується діаграма розгортання, наведена на рисунку 3.1 [33]. Цей тип структурних діаграм показує, як саме програмні компоненти системи будуть розподілені по фізичних вузлах обробки, після її встановлення та запуску. Таким чином, вона моделює конфігурацію системи в реальному робочому середовищі.

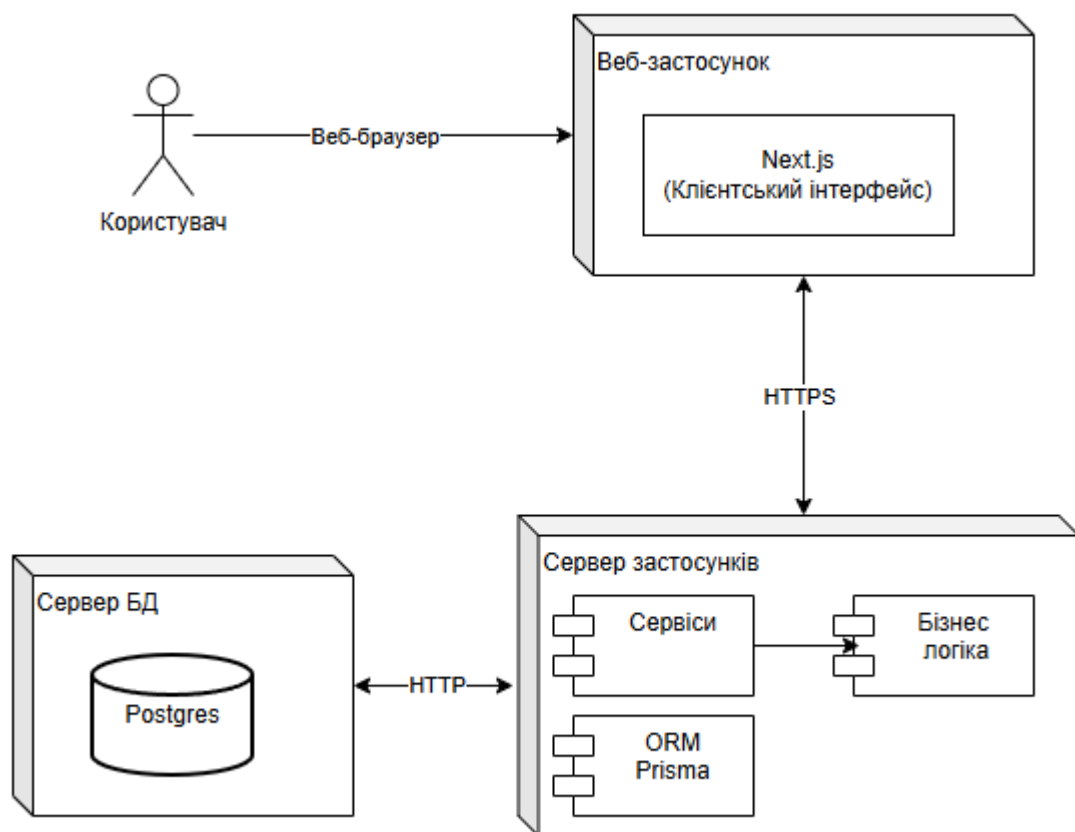


Рис 3.1 Діаграма розгортання додатку

Надана діаграма розгортання ілюструє фізичну архітектуру системи.

На вузлі Web-application розгортається клієнтська частина застосунку. Вона включає інтерфейс користувача, створений на базі Next.js, а також відповідні модулі та бібліотеки. Цей вузол відповідає за візуалізацію даних та комунікацію з серверною частиною, надсилаючи запити та отримуючи дані.

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

Користувачі взаємодіють із системою саме через цей вузол за допомогою веббраузера.

Application Server є центральним вузлом, де розгорнуто серверну частину додатку. Його призначення — обробка даних, отриманих від клієнтської частини, та виконання основної бізнес-логіки системи. На цьому сервері розміщені компоненти "Services", "Business Logic" та "ORM Prisma", які спільно забезпечують функціонування серверної логіки та взаємодію з базою даних. Зв'язок з клієнтською частиною відбувається через безпечний протокол HTTPS.

На окремому вузлі Database Server розгорнуто базу даних Postgres. Цей сервер відповідає виключно за зберігання та управління даними. Application Server взаємодіє з ним через протокол HTTP для запису та читання інформації, забезпечуючи надійне збереження всіх оброблених даних системи.

3.2 Структура інтерфейсу та порядок взаємодії з системою енциклопедій манга

Продемонструємо інтерфейс системи на прикладі основного сценарію виконання додатку. Коли користувач авторизується в системі, його зустрічає головна сторінка з рекомендаціями та найновішими мангами (рис. 3.1).

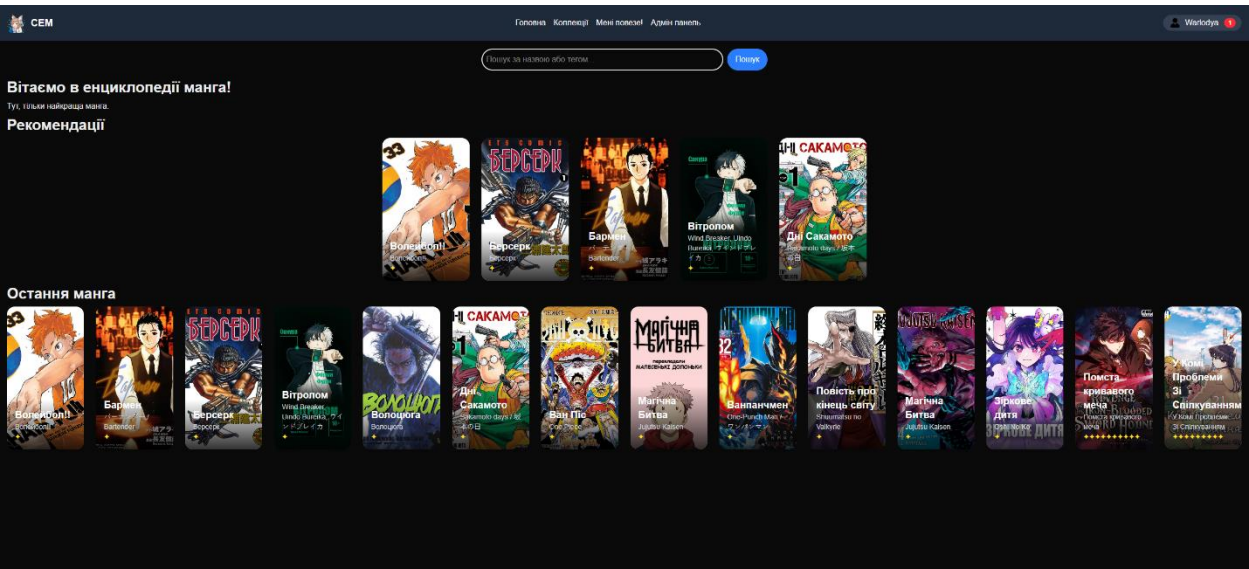


Рис 3.1 Інтерфейс головної сторінки

Коли користувач відкриває сторінку певної манги, йому показується вся інформація про неї, включно з описом, тегами, списком томів та можливістю залишити та прочитати відгуки (рис. 3.2).

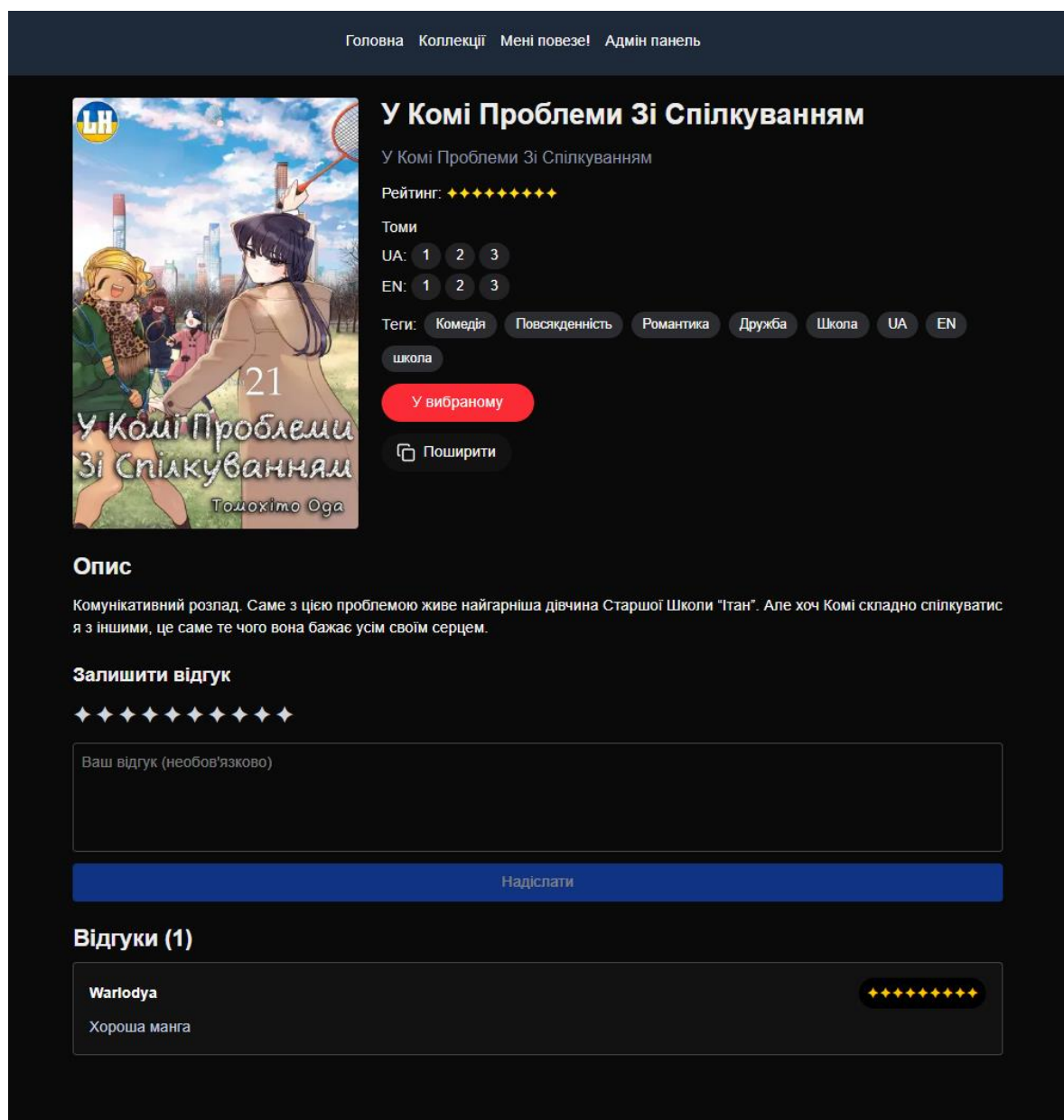


Рис 3.2 Інтерфейс сторінки манги

Для початку читання користувач переходить у «рідер», який має мінімалістичний дизайн для комфортного перегляду. Основну частину екрана займає сторінка манги, а зверху розташовані елементи навігації для повернення до опису, переходу до коментарів або до наступного тому (рис. 3.3).

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савицький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		55



Рис 3.3 Інтерфейс рідеру манги

Також авторизований користувач може взаємодіяти з секцією коментарів, призначеною для обговорення контенту користувачами (рис. 3.4).

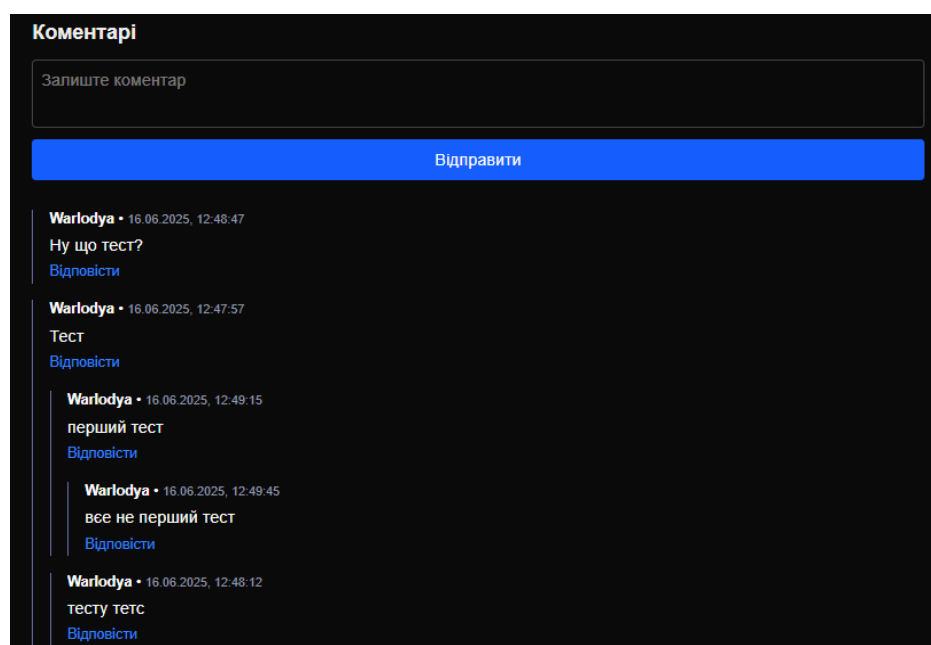


Рис 3.4 Інтерфейс коментарів

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				56
Змн.	Арк.	№ докум.	Підпис	Дата		

Розділ "Колекції" надає користувачам доступ до перегляду добірок, створених спільнотою. Окрім цього, реалізовано функціонал, що дозволяє ділитися посиланням на будь-яку з представлених колекцій (рис. 3.5 – 3.6).

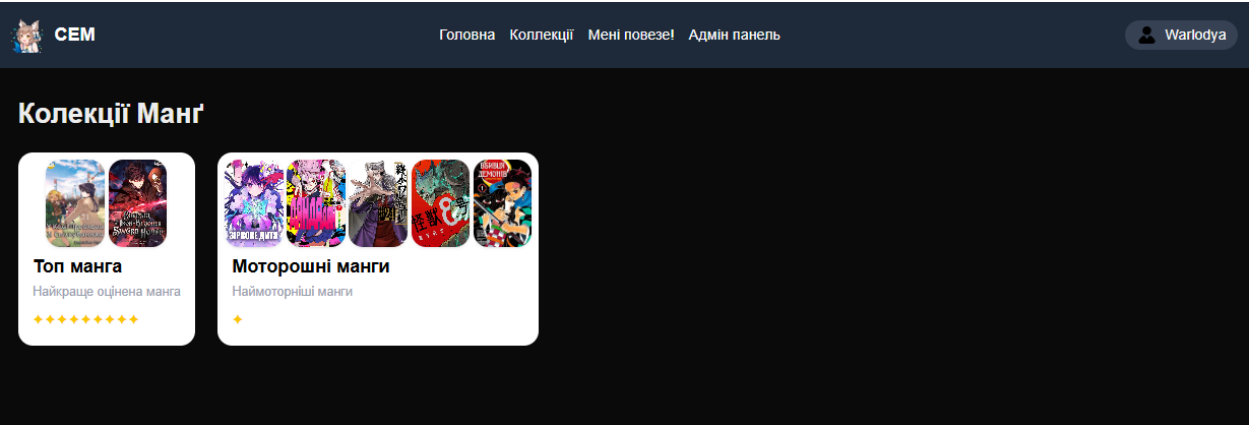


Рис 3.5 Сторінка колекцій

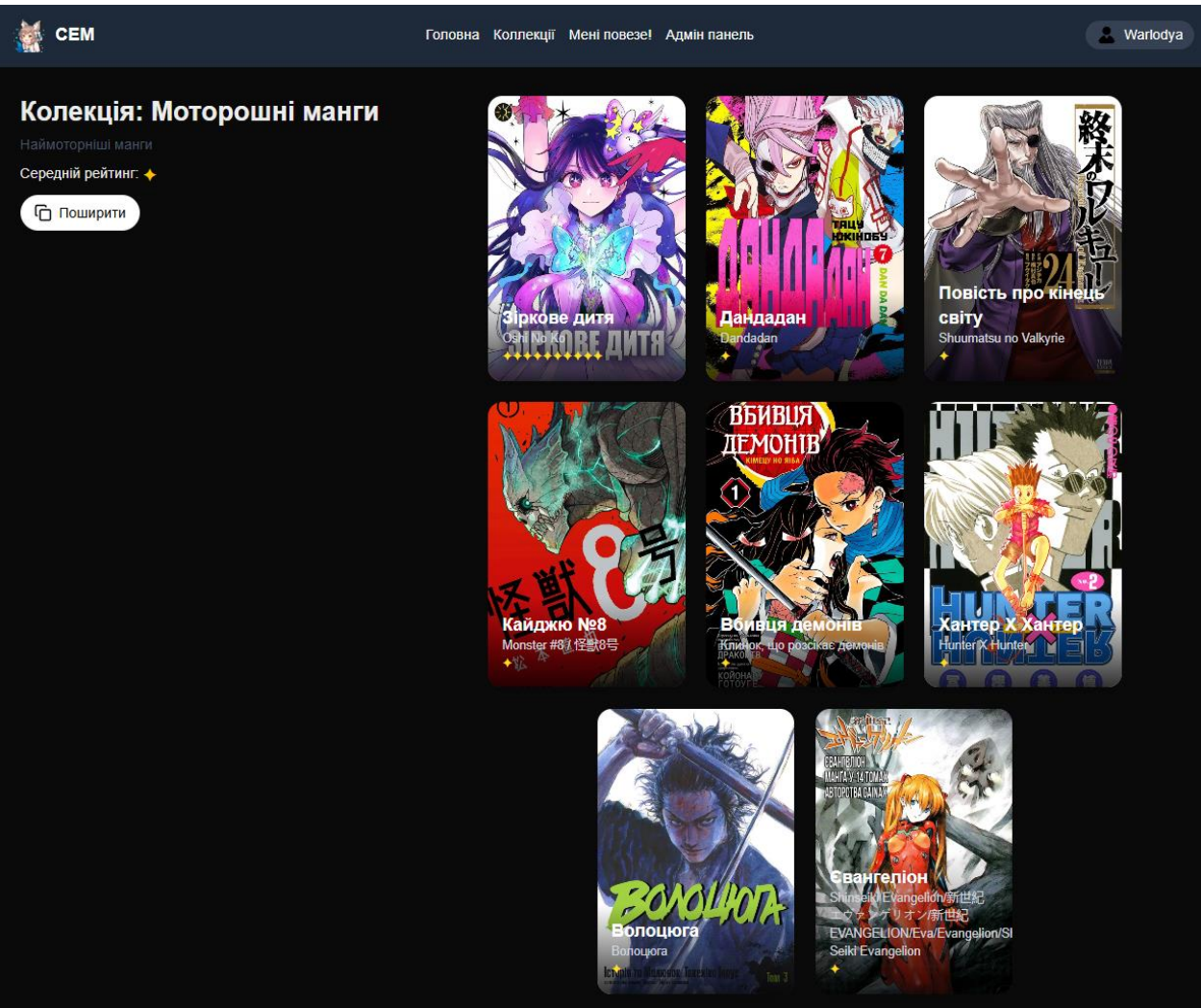


Рис 3.6 Вигляд колекції

Коли користувач відкриває сторінку свого профілю, йому відображається вся персоналізована інформація, а саме: його особисті дані, кнопка для виходу з акаунту, інструменти для управління своїми колекціями, а також візуальні добірки улюбленої манги, історії переглядів, сповіщення про оновлення улюбленої манги (рис. 3.7 – 3.10).

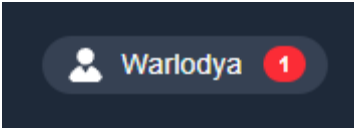


Рис 3.7 Вигляд профілю з сповіщенням про оновлення манги

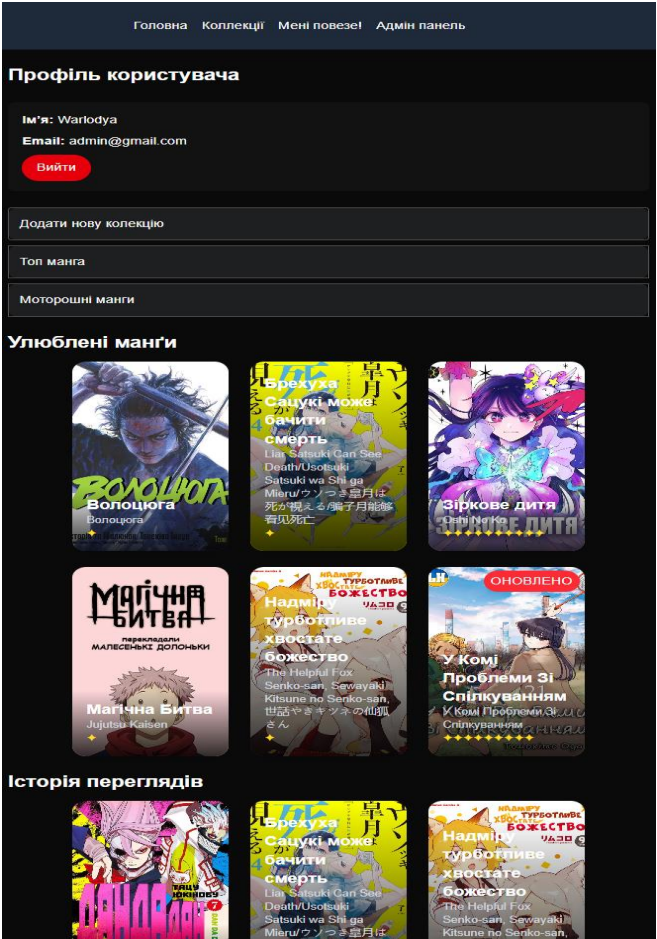


Рис 3.8 Вигляд профілю користувача



Рис 3.9 Вигляд оновленої манги

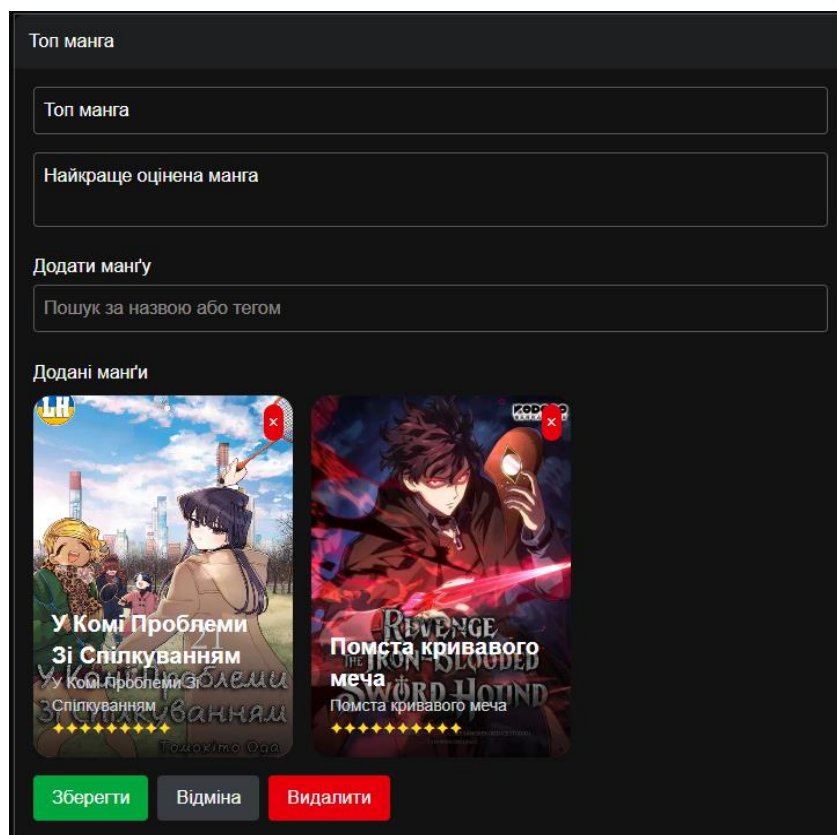


Рис 3.10 Меню редагування колекції

Якщо користувач має роль «Автор» чи «Адміністратор» він має доступ до адмін-панелі, в якій присутні 3 редактори: редактор манги, редактор колекцій та редактор користувачів («Автор» має доступ лише до редактора манги).

Коли користувач з відповідною роллю заходить в редактор манги він може створити нову мангу або ж редагувати наповнення вже існуючих манг (рис. 3.11 – 3.12).

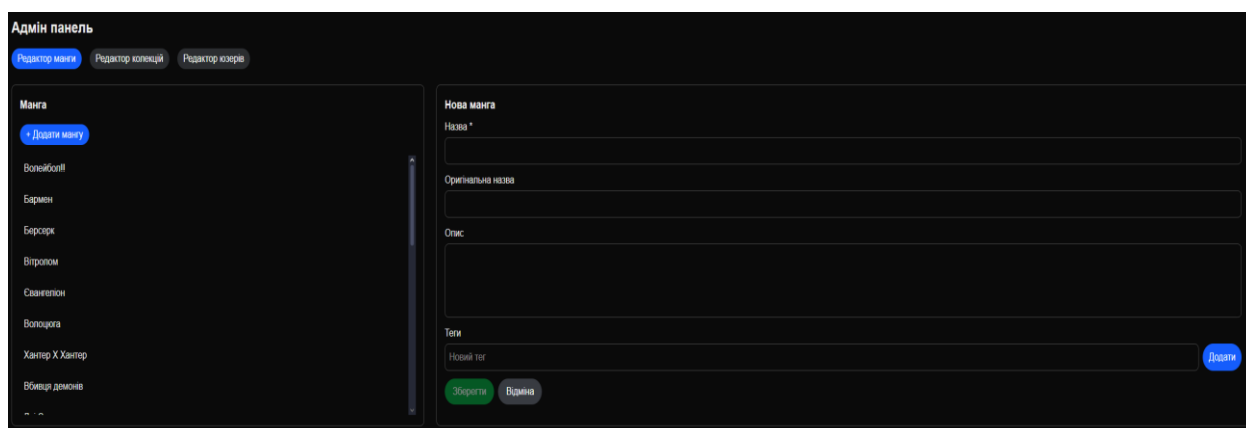


Рис 3.11 Меню редактора манги (створення нової манги)

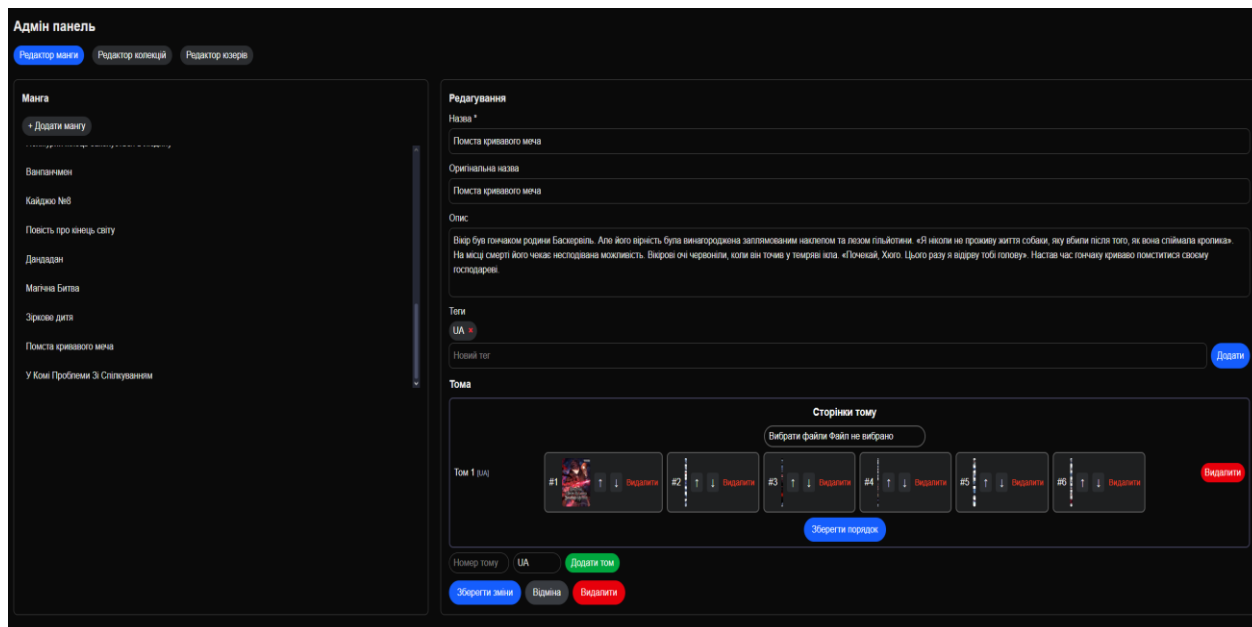


Рис 3.12 Меню редактора манги (редагування / наповнення манги)

В режимі редагування манги користувач може редагувати наповнення манги таке, як опис, назву та теги, так і створити новий том з наповненням сторінок.

В режимі редактору колекцій користувач може керувати будь якими колекціями та їх наповненням (рис. 3.13).

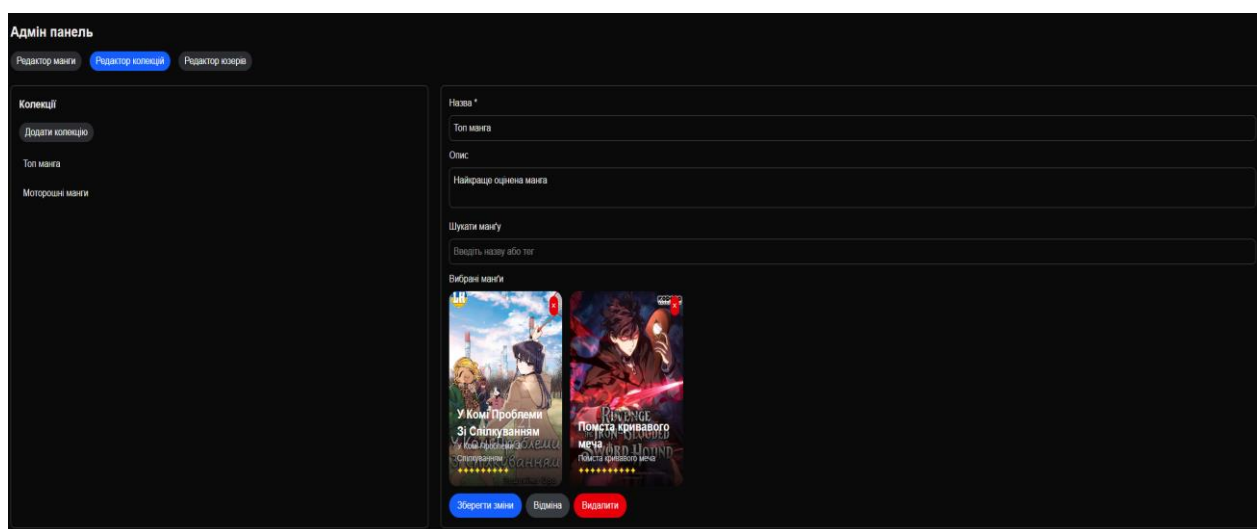


Рис 3.13 Меню редактора колекцій

В режимі редактору користувачів адміністратор може переглянути користувачів та заблокувати їх(рис. 3.14).

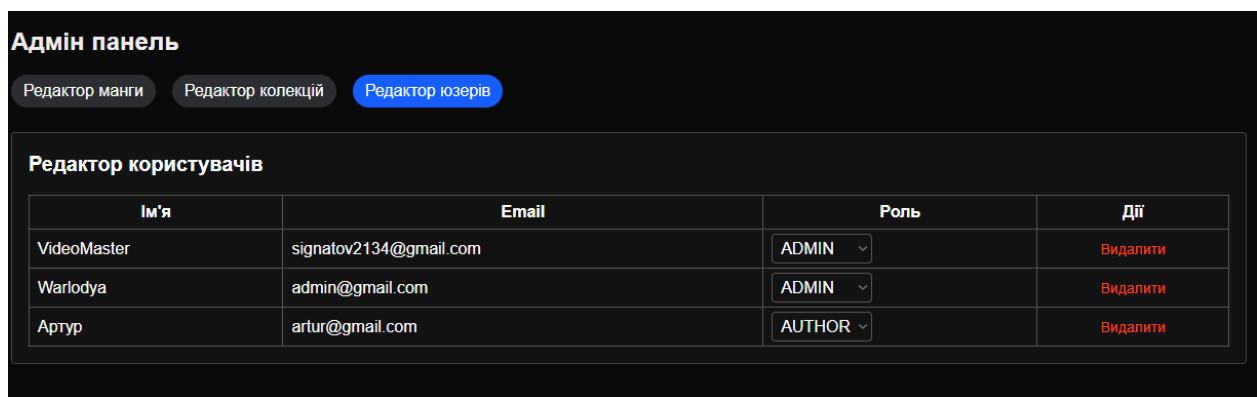


Рис 3.14 Меню редактора користувачів

3.3 Тестування роботи програмного продукту

Розробка будь-якого програмного продукту - це складний процес, у якому неможливо одразу врахувати всі сценарії використання, тому виникнення помилок є неминучим. Саме тому тестування є ключовим етапом для забезпечення високої якості додатку. В ході тестування перевіряється коректність обробки даних, робота функціоналу, стійкість до збоїв та багато інших аспектів. Головна мета - виявити та усунути максимальну кількість дефектів до того, як продукт потрапить до кінцевих користувачів.

Для систематизації процесу тестування були розроблені тестові випадки, або test cases [34]. Кожен тест-кейс є набором кроків та умов для перевірки роботи певного функціоналу програмної системи. Створені для проєкту тестові випадки представлені у вигляді таблиці (табл. 3.1).

Таблиця 3.1

Таблиця Test Cases

Опис Тестового Випадку	Кроки Тестування	Тестові Дані	Очікуваний Результат	Фактичний Результат	Статус
Перевірка авторизації з дійсними даними	1. Перейти на сторінку входу. 2. Ввести email. 3. Ввести пароль. 4. Натиснути "Увійти".	Email: admin@gmail.com Пароль: admin	Користувач успішно входить у систему.	Як і очікувалося	Pass

Продовження таблиці 3.1

Опис Тестового Випадку	Кроки Тестування	Тестові Дані	Очікуваний Результат	Фактичний Результат	Статус
Перевірка авторизації з недійсним паролем	1. Перейти на сторінку входу. 2. Ввести email. 3. Ввести недійсний пароль. 4. Натиснути "Увійти".	Email: admin@gmail.com Пароль: 123admin	Користувач не повинен увійти. З'являється повідомлення про помилку "Неправильний логін або пароль".	Як і очікувалося	Pass
Перегляд сторінки манги	1. Авторизуватися в системі. 2. Перейти на головну сторінку. 3. Натиснути на будь-яку мангу з каталогу.	-	Відкривається сторінка з детальним описом манги, її томами та відгуками.	Як і очікувалося	Pass
Додавання коментаря до тому	1. Авторизуватися в системі. 2. Відкрити будь-яку мангу та перейти до читання тому. 3. Перейти до секції коментарів. 4. Ввести текст у поле коментаря. 5. Натиснути "Відправити".	Текст: "Чудовий том!"	Коментар з'являється у списку під полем вводу.	Як і очікувалося	Pass
Додавання манги до улюблених	1. Авторизуватися в системі. 2. Відкрити сторінку будь-якої манги. 3. Натиснути на іконку "Додати до улюбленого". 4. Перейти на сторінку свого профілю.	-	Обрана манга з'являється у секції "Улюблені манги".	Як і очікувалося	Pass

Закінчення таблиці 3.1

Опис Тестового Випадку	Кроки Тестування	Тестові Дані	Очікуваний Результат	Фактичний Результат	Статус
Створення нової манги (Адміністратор)	1. Авторизуватися під обліковим записом адміністратора. 2. Перейти до "Адмін панелі". 3.Відкрити "Редактор манги" та натиснути "Додати мангу". 4. Заповнити всі обов'язкові поля. 5.Натиснути "Зберегти".	Назва: "Тестова манга" Опис: "Опис для тесту"	Нова манга створюється і з'являється у загальному списку в редакторі.	Як і очікувалося	Pass

Також, необхідно перевірити функціонал проекту за допомогою Postman [35].

На (рис. 3.15) зображено результат успішного GET-запиту на отримання списку всієї манги з бази даних.

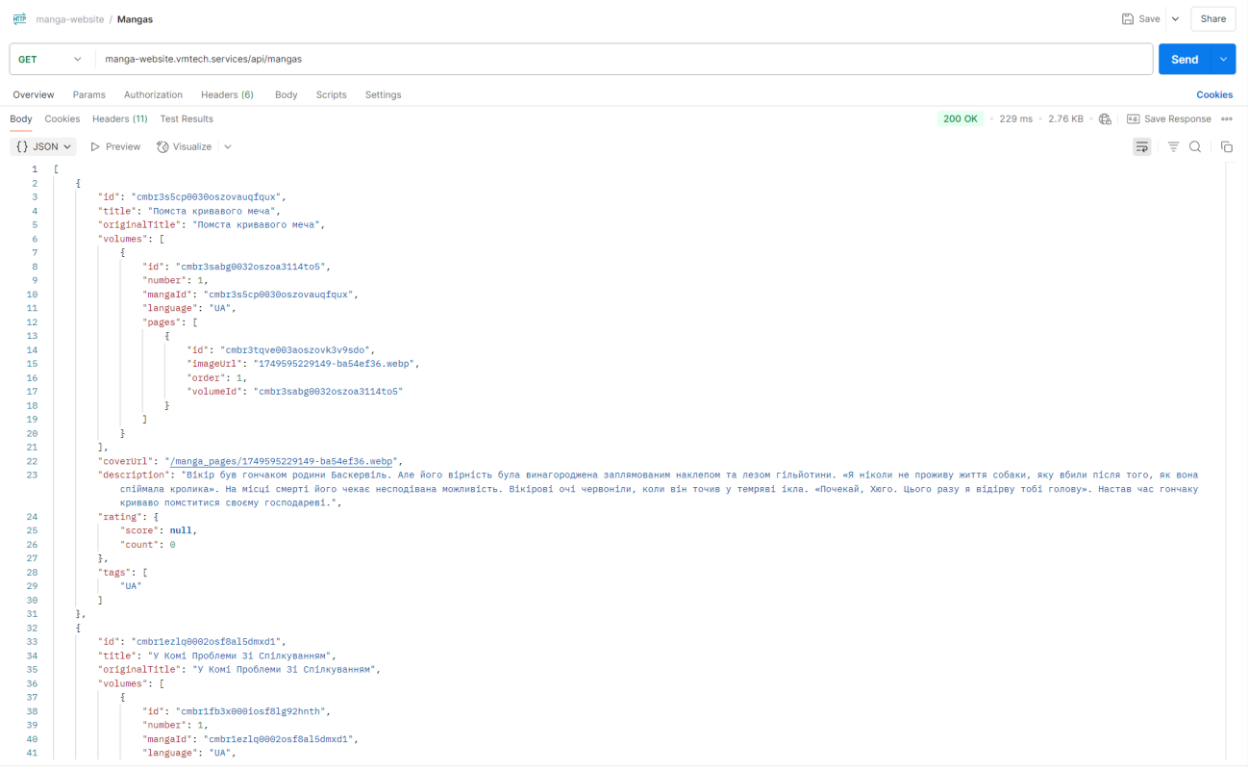


Рис 3.15 Спроба отримання списку всієї манги

Як і очікувалося, дія виконана успішно. Сервер повернув статус 200 ОК та масив об'єктів, де кожен об'єкт представляє окрему мангу з її основними даними та вкладеними томами.

Далі було перевірено отримання даних по конкретній манзі за її ідентифікатором (рис. 3.16).

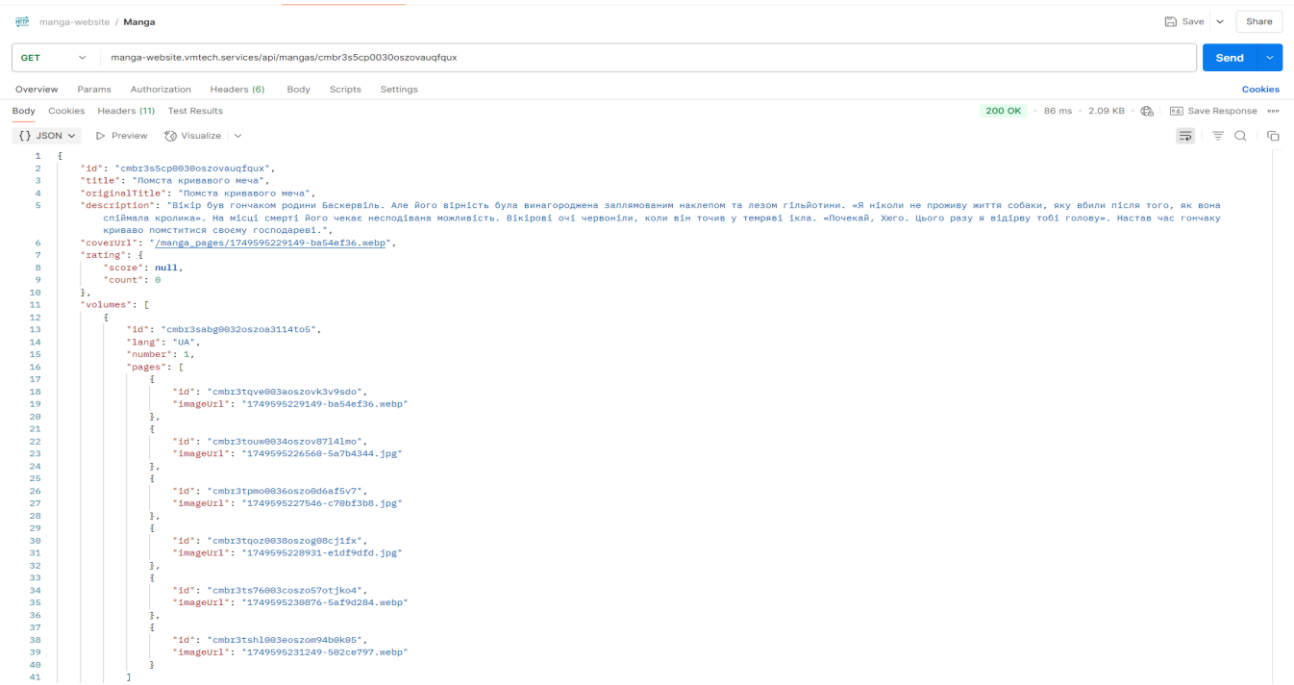


Рис 3.16 Спроба отримання конкретної манги за ID

Ця операція також завершилася успішно. Сервер повернув об'єкт з повною інформацією про запитувану мангу, включно з її описом, рейтингом та структурою томів.

Наступним кроком була перевірка отримання списку томів для конкретної манги (рис. 3.17) та списку сторінок для конкретного тому (рис. 3.18).

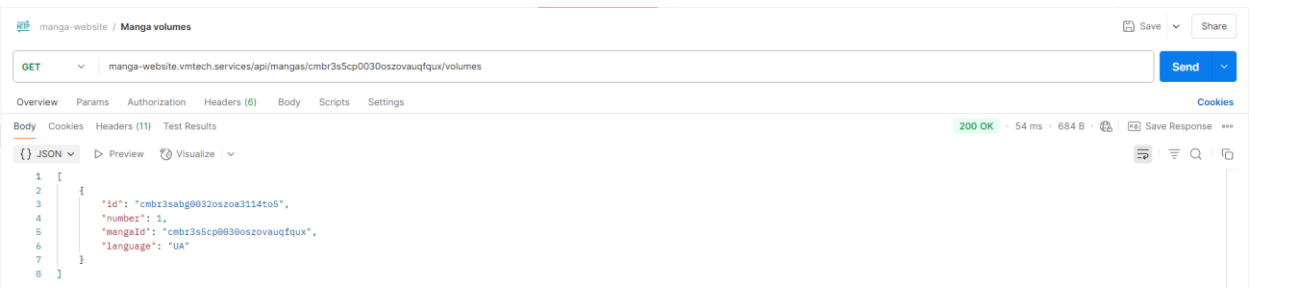


Рис 3.17 Спроба отримання списку томів для манги

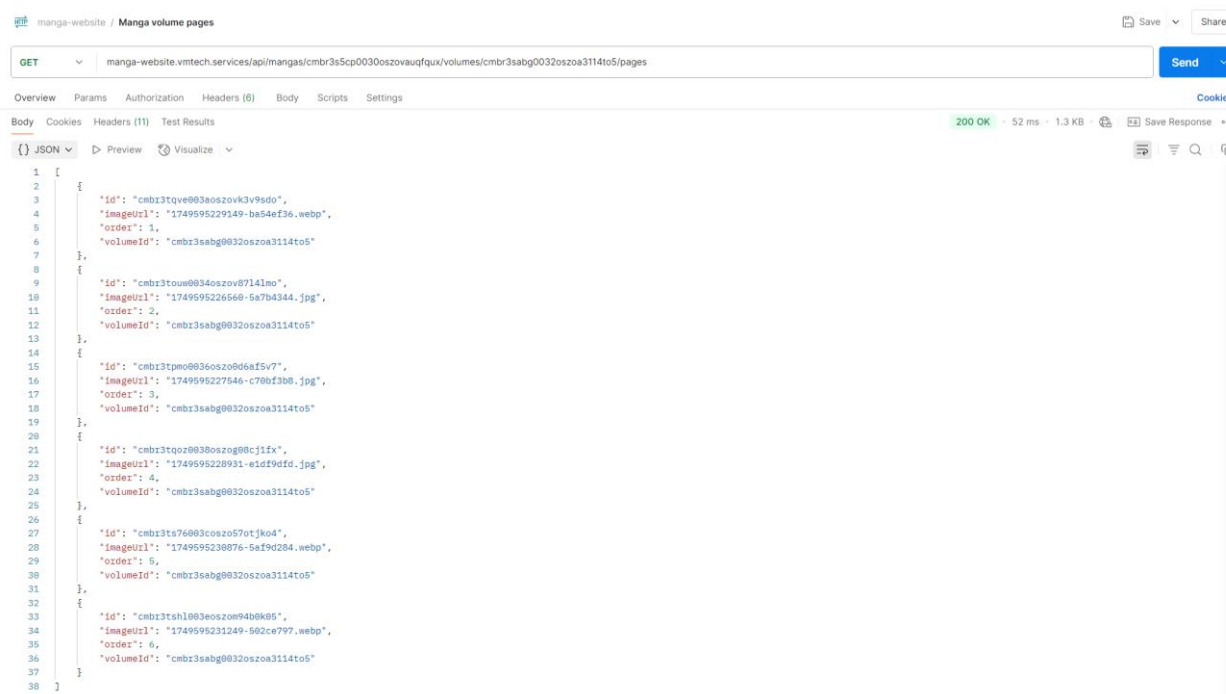


Рис 3.18 Спроба отримання списку сторінок для тому

Обидва запити були виконані коректно. Сервер повернув відповідні масиви даних зі статусом 200 OK, що підтверджує правильну роботу ієрархічних зв'язків у базі даних.

Перевіримо роботу з колекціями. На (рис. 3.19) показано результат запиту на отримання всіх наявних колекцій.

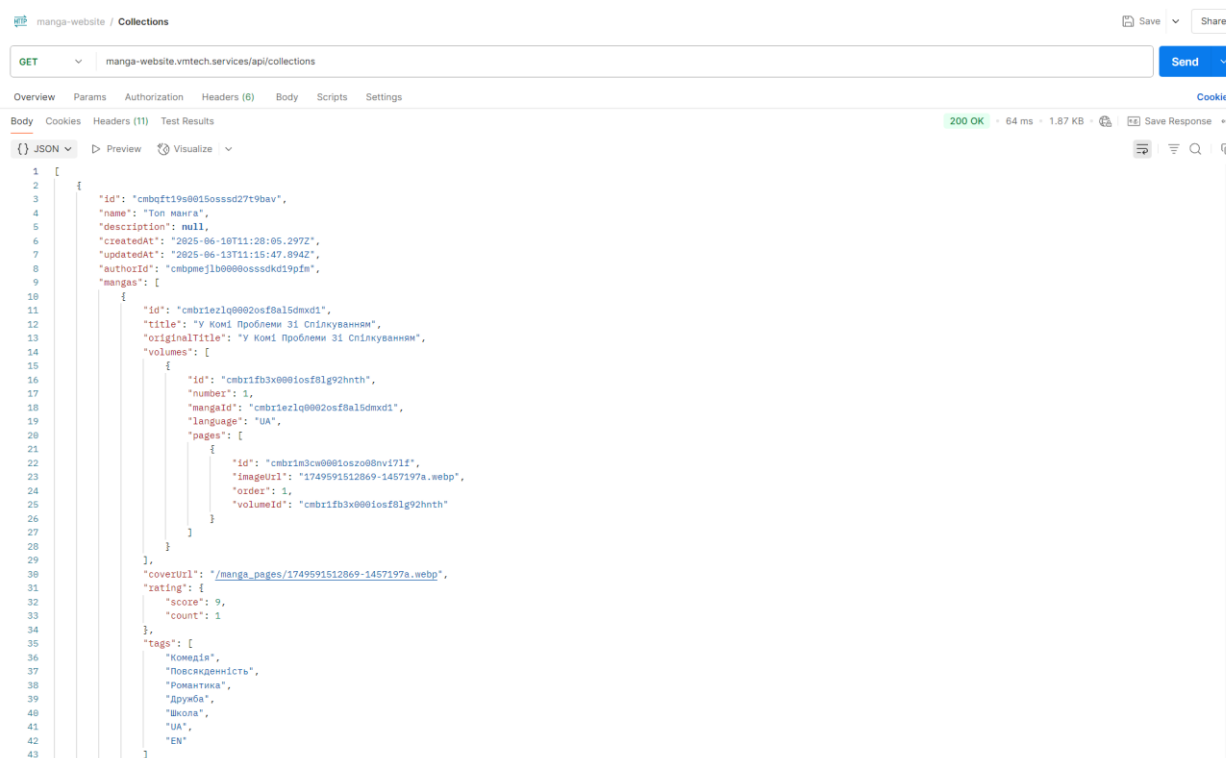


Рис 3.19 Спроба отримання списку колекцій

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

Як і очікувалося, сервер повернув масив колекцій, де кожна містить свою назву, опис та вкладений список манги, що до неї входить.

Наостанок було протестовано функціонал пошуку (рис. 3.20).

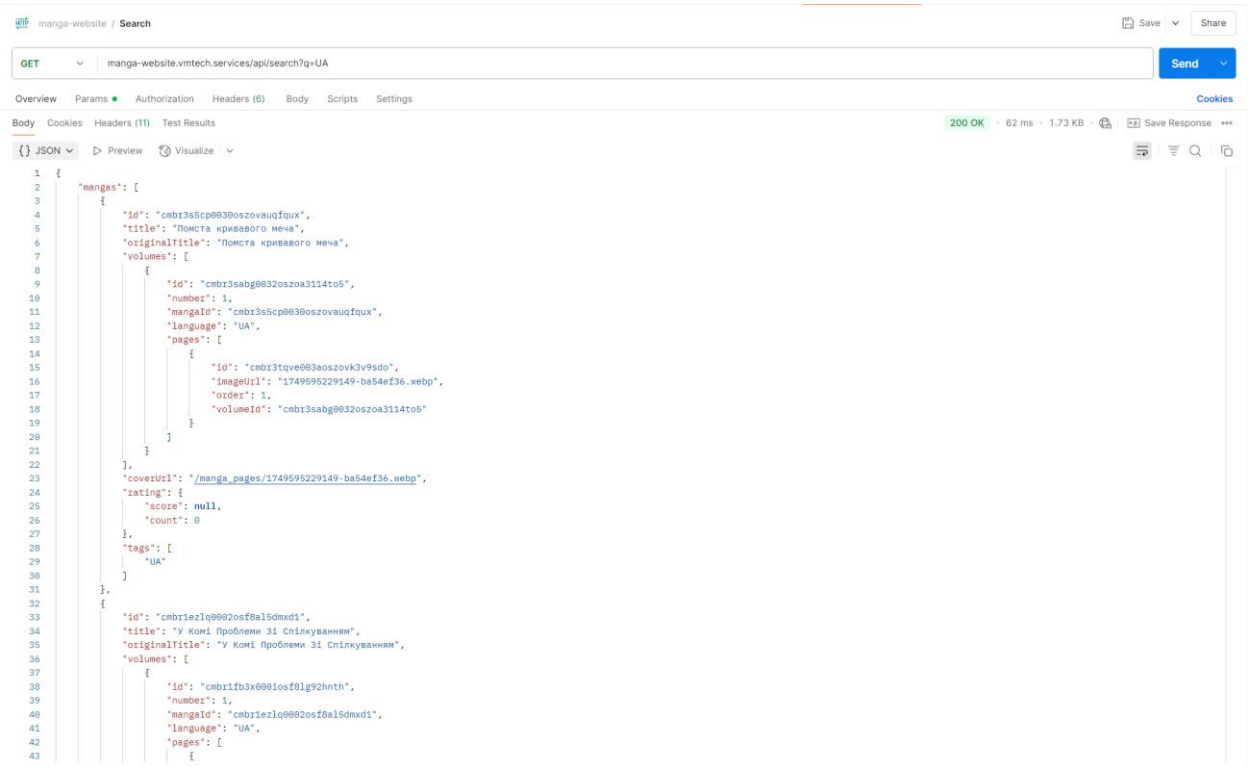


Рис 3.20 Результат роботи пошукового запиту

Ця дія також виконана успішно. На запит із ключовим словом сервер повернув масив манги, що відповідає критеріям пошуку, демонструючи коректну роботу пошукового алгоритму.

Висновки до третього розділу

В цьому розділі було побудовано та показано діаграму розгортання системи. Також було продемонстровано інтерфейс та порядок взаємодії з системою енциклопедій манга. Було докладно описано функціональні можливості додатку в залежності від ролі користувача. А також проведено тестування як серверної так і клієнтської частини розробленої системи за допомогою Test Cases та Postman. В ході проведення тестування та перевірок було виявлено та виправлено знайдені помилки. В результаті отримано застосунок відповідної якості.

ВИСНОВКИ

Під час виконання випускної кваліфікаційної роботи бакалавра було спроектовано та розроблено вебплатформу «Система енциклопедій манга». В ході роботи було проведено детальне дослідження предметної області, що включало порівняльний аналіз існуючих аналогів. На основі виявлених недоліків конкурентів, зокрема відсутності ефективних систем сповіщень та зручної багатомовності, були сформульовані ключові функціональні та нефункціональні вимоги до системи.

Проектування системи включало аналіз основних варіантів використання платформи для різних ролей користувачів, а також розробку архітектури системи на основі сучасного технологічного стеку Next.js, PostgreSQL та Prisma. Було спроектовано реляційну базу даних, що забезпечує надійне зберігання інформації про всі сутності системи.

У ході роботи було продемонстровано та детально описано основний потік взаємодії з системою. Було розглянуто ключовий функціонал, що включає перегляд та читання манги, створення персоналізованих колекцій, коментування, а також розширені можливості для авторів та адміністраторів з управління контентом та користувачами. Для візуалізації архітектури та логіки роботи додатку були розроблені та описані відповідні діаграми. Наведено приклади з детальним описом інтерфейсу розробленої системи для кожної ролі.

Розроблена система вирізняється на тлі аналогів завдяки реалізації двох ключових інновацій: централізованої системи сповіщень про оновлення манги та зручного мультимовного функціоналу. Інтерфейс платформи є інтуїтивно зрозумілим, сучасним та адаптованим для різних пристроїв.

Результатом кваліфікаційної роботи є готова до експлуатації «Система енциклопедій манга», яка надає комплексне рішення для спільноти поціновувачів манги, забезпечуючи зручні інструменти для читання, публікації та соціальної взаємодії, і вирішує основні проблеми, наявні в аналогічних продуктах на ринку.

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Manga History & Critiques [Електронний ресурс]. – Режим доступу до ресурсу: <https://libguides.calstatela.edu/c.php?g=1141475&p=8364977> (дата звернення: 18.06.2025).
2. What Did Osamu Tezuka Change in Manga? [Електронний ресурс]. – Режим доступу до ресурсу: <https://artsandculture.google.com/story/twVxHS43JkMsKg> (дата звернення: 18.06.2025).
3. One Piece Manga Sets Guinness World Record With Over 500 Million Published [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.animenewsnetwork.com/news/2022-08-04/one-piece-manga-sets-guinness-world-record-with-over-500-million-published/.188352> (дата звернення: 18.06.2025).
4. What is an API? [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.oracle.com/cloud/cloud-native/api-management/what-is-api/> (дата звернення: 18.06.2025).
5. Role-based access control: 11 top tips [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.cerbos.dev/blog/role-based-access-control-best-practices> (дата звернення: 18.06.2025).
6. Rating System and Rating Architecture: A Rating Infrastructure [Електронний ресурс]. – Режим доступу до ресурсу: <https://fastercapital.com/content/Rating-System--Rating-System-and-Rating-Architecture--A-Rating-Infrastructure.html> (дата звернення: 18.06.2025).
7. What is a Recommendation Engine? [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.ibm.com/think/topics/recommendation-engine> (дата звернення: 18.06.2025).
8. The Fundamentals of UI and UX Design: A Comprehensive Guide [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.thecuneiform.com/insights/the-fundamentals-of-ui-and-ux-design-a-comprehensive-guide/> (дата звернення: 18.06.2025).

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

9. Adaptive design: a comprehensive guide [Електронний ресурс]. – Режим доступу до ресурсу: <https://blog.depositphotos.com/adaptive-design.html> (дата звернення: 18.06.2025).
10. What is Backend Development? [Електронний ресурс]. – Режим доступу до ресурсу: <https://roadmap.sh/backend> (дата звернення: 18.06.2025).
11. What Does a Back-End Developer Do? [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.coursera.org/articles/back-end-developer> (дата звернення: 18.06.2025).
12. OWASP Node.js Authentication, Authorization and Cryptography Practices [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.nodejs-security.com/blog/owasp-nodejs-authentication-authorization-cryptography-practices> (дата звернення: 18.06.2025).
13. What is Functional Testing? Types & Examples [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.opentext.com/what-is/functional-testing> (дата звернення: 18.06.2025).
14. Офіційний сайт Honey Manga [Електронний ресурс]. – Режим доступу до ресурсу: <https://honey-manga.com.ua/> (дата звернення: 18.06.2025).
15. Офіційний сайт Zenko Online [Електронний ресурс]. – Режим доступу до ресурсу: <https://zenko.online/> (дата звернення: 18.06.2025).
16. Офіційний сайт Manga.in.ua [Електронний ресурс]. – Режим доступу до ресурсу: <https://manga.in.ua/> (дата звернення: 18.06.2025).
17. Офіційний сайт Bato.to [Електронний ресурс]. – Режим доступу до ресурсу: <https://bato.to/> (дата звернення: 18.06.2025).
18. Офіційний сайт Mangaplaza [Електронний ресурс]. – Режим доступу до ресурсу: <https://mangaplaza.com/> (дата звернення: 18.06.2025).
19. TypeScript: JavaScript With Syntax For Types [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.typescriptlang.org/> (дата звернення: 18.06.2025).
20. Getting started with React [Електронний ресурс]. – Режим доступу до ресурсу: <https://developer.mozilla.org/en->

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

[US/docs/Learn_web_development/Core/Frameworks/libraries/React_getting_started](https://reactjs.org/docs/learn-web-development/Core/Frameworks/libraries/React_getting_started) (дата звернення: 18.06.2025).

21. Learn Next.js [Електронний ресурс]. – Режим доступу до ресурсу: <https://nextjs.org/learn/pages-router> (дата звернення: 18.06.2025).
22. Next.js by Vercel - The React Framework [Електронний ресурс]. – Режим доступу до ресурсу: <https://devdocs.io/nextjs/> (дата звернення: 18.06.2025).
23. Server-side rendering (SSR) [Електронний ресурс]. – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Glossary/SSR> (дата звернення: 18.06.2025).
24. PostgreSQL Tutorial [Електронний ресурс]. – Режим доступу до ресурсу: <https://neon.com/postgresql/tutorial> (дата звернення: 18.06.2025).
25. ACID Transactions in Databases [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.databricks.com/glossary/acid-transactions> (дата звернення: 18.06.2025).
26. Mastering PostgreSQL JSONB type in one article [Електронний ресурс]. – Режим доступу до ресурсу: <https://blog.logto.io/mastering-postgresql-jsonb> (дата звернення: 18.06.2025).
27. Next-generation Node.js and TypeScript ORM [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.prisma.io/orm> (дата звернення: 18.06.2025).
28. How to Make a UML Class Diagram (and Others) With Examples [Електронний ресурс]. – Режим доступу до ресурсу: <https://slickplan.com/blog/how-to-make-a-uml-diagram> (дата звернення: 18.06.2025).
29. Complete Guide to Database Schema Design [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.integrate.io/blog/complete-guide-to-database-schema-design-guide/> (дата звернення: 18.06.2025).

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

30. HTTP request methods [Електронний ресурс]. – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods> (дата звернення: 18.06.2025).
31. What is a UML Sequence Diagram? [Електронний ресурс]. – Режим доступу до ресурсу: <https://miro.com/diagramming/what-is-a-uml-sequence-diagram/> (дата звернення: 18.06.2025).
32. 5 Best Practices for Data Validation [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.dsstream.com/post/5-best-practices-for-data-validation> (дата звернення: 18.06.2025).
33. Deployment diagrams in UML modeling [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-deployment> (дата звернення: 18.06.2025).
34. How to Write Test Cases for Software Testing [Електронний ресурс]. – Режим доступу до ресурсу: <https://luxequality.com/blog/how-to-write-test-cases-for-software-testing/> (дата звернення: 18.06.2025).
35. Postman | The Collaboration Platform for API Development [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.postman.com/> (дата звернення: 18.06.2025).

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

ДОДАТКИ

		Чуприна В.А.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

Специфікація прецедентів

Таблиця А.1

Ім'я варіанту використання:	Реєстрація та Автентифікація в системі
Рівень важливості:	Критичний
Основний актор:	Користувач
Інші учасники:	Відсутні
Короткий опис:	Надання користувачу можливості створити обліковий запис та увійти в систему для доступу до персоналізованого функціоналу.
Передумови:	Користувач перебуває на головній сторінці або сторінці входу/реєстрації.
Основний потік:	1. Користувач переходить на сторінку реєстрації, заповнює форму (email, ім'я, пароль) та підтверджує її. 2. Система перевіряє унікальність email та валідність даних. 3. Після успішної реєстрації користувач може перейти на сторінку входу. 4. Користувач вводить свій email та пароль для авторизації в системі.
Альтернативні/виключні потоки:	• Якщо email вже існує, система виводить повідомлення про помилку. • Якщо введені дані невалідні (неправильний формат email, короткий пароль), система підсвічує помилкові поля. • У разі неправильного логіна або пароля під час входу, система інформує про це користувача.

Таблиця А.2

Ім'я варіанту використання:	Перегляд та пошук манги
Рівень важливості:	Високий
Основний актор:	Користувач
Інші учасники:	Відсутні
Короткий опис:	Надання користувачу можливості знаходити та читати мангу в каталозі.
Передумови:	Користувач авторизований або не авторизований у системі.
Основний потік:	1. Користувач переходить до каталогу манги. 2. Використовує рядок пошуку або фільтри (за тегами, жанрами) для знаходження потрібного твору. 3. Обирає мангу зі списку для переходу на її детальну сторінку. 4. Обирає потрібний том та переходить до режиму читання («рідер»).
Альтернативні/виключні потоки:	• Якщо за результатами пошуку нічого не знайдено, система виводить відповідне повідомлення.

Таблиця А.3

Ім'я варіанту використання:	Управління профілем та колекціями
Рівень важливості:	Високий
Основний актор:	Користувач
Інші учасники:	Відсутні
Короткий опис:	Надання користувачу можливості переглядати свою активність, додавати мангу до улюблених та створювати власні колекції.
Передумови:	Користувач авторизований у системі.
Основний потік:	1. Користувач переходить на сторінку свого профілю. 2. Переглядає історію читання та список улюбленої манги. 3. На сторінці манги натискає «Додати до улюбленого». 4. У профілі натискає «Створити нову колекцію», заповнює назву, опис та додає до неї мангу.
Альтернативні/виключні потоки:	• Якщо користувач намагається додати до колекції мангу, яка вже там є, система може вивести відповідне сповіщення.

Таблиця А.4

Ім'я варіанту використання:	Публікація та редагування манги
Рівень важливості:	Критичний
Основний актор:	Автор
Інші учасники:	Адміністратор (модерація)
Короткий опис:	Надання автору можливості додавати нові твори та керувати їхнім вмістом.
Передумови:	Користувач має роль «Автор» та авторизований у системі.
Основний потік:	1. Автор переходить до «Адмін-панелі» (обмеженої). 2. У «Редакторі манги» натискає «Додати мангу». 3. Заповнює форму з метаданими (назва, опис, теги). 4. Після створення манги додає до неї томи. 5. Для кожного тому завантажує сторінки. 6. Автор може редагувати метадані та вміст своєї манги.
Альтернативні/виключні потоки:	• Якщо під час завантаження файлів виникає помилка, система повідомляє про це автора. • Якщо спробувати зберегти форму з незаповненими обов'язковими полями, система виводить помилку валідації.

Таблиця А.5

Ім'я варіанту використання:	Управління користувачами
Рівень важливості:	Критичний
Основний актор:	Адміністратор
Інші учасники:	Користувачі
Короткий опис:	Надання адміністратору можливості керувати обліковими записами та ролями користувачів.
Передумови:	Користувач має роль «Адміністратор» та авторизований у системі.
Основний потік:	1. Адміністратор переходить до «Адмін-панелі». 2. Відкриває «Редактор користувачів». 3. Переглядає список всіх користувачів системи. 4. Може змінити роль будь-якого користувача (наприклад, надати права «Автора»). 5. Може видалити (заблокувати) обліковий запис користувача.
Альтернативні/виключні потоки:	• При спробі видалення власного облікового запису система може блокувати цю дію.

Таблиця А.6

Ім'я варіанту використання:	Модерація контенту
Рівень важливості:	Високий
Основний актор:	Адміністратор
Інші учасники:	Користувачі, Автори
Короткий опис:	Забезпечення порядку та якості контенту на платформі шляхом перевірки та видалення невідповідних матеріалів.
Передумови:	Користувач має роль «Адміністратор» та авторизований у системі.
Основний потік:	1. Адміністратор переглядає нові коментарі, рецензії або завантажену мангу. 2. У разі виявлення порушення правил спільноти (спам, образливий контент), адміністратор видаляє відповідний коментар/рецензію. 3. Адміністратор може редагувати або видаляти будь-яку мангу в системі.
Альтернативні/виключні потоки:	• Відсутні.