Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка програмного забезпечення веб-платформи для публікації та читання веб-новел з використанням мови програмування JavaScript

Виконав(ла): студент(ка) 4 курсу, групи СП-42
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

(підпис) Подейко М.А. (прізвище та ініціали)

Керівник (підпис) Яворська Є.Б. (прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю.М. (прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М.Р. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра Програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М.Р.
(підпис) (прізвище та ініціали)
« » 2026 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

студенту Подейко Максиму Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного забезпечення веб-платформи для публікації та читання веб-новел з використанням мови програмування JavaScript

Керівник роботи Яворська Євгенія Богданівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 06 » *квітня* 2026 року № *4/9-170*

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Вимоги замовника, технічні умови, технічне завдання

4. Зміст роботи (перелік питань, які потрібно розробити)

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності,			
основи охорони праці			

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання		
2	Аналіз завдання		
3	Виконання розділу 1		
4	Виконання розділу 2		
5	Виконання розділу 3		
6	Виконання розділу 4		
7	Оформлення пояснювальної записки		
8	Оформлення графічного та презентаційного матеріалу		
9	Попередній захист		
10	Захист		

Студент _____ Подейко М.А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Яворська Є.Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка програмного забезпечення веб-платформи для публікації та читання веб-новел з використанням мови програмування JavaScript // Кваліфікаційна робота освітнього рівня «Бакалавр» // Подейко Максим Анлрійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СП-42 // Тернопіль, 2026 // С. 110, рис. 15, табл. 38, кресл. 11, додат. 3, бібліогр. 35.

Ключові слова: веб-платформа, веб-новела, JavaScript, React, Node.js, Express.js, PostgreSQL, REST API, веб-застосунок, програмне забезпечення.

Кваліфікаційну роботу присвячено розробці програмного забезпечення веб-платформи для публікації та читання веб-новел із використанням сучасних JavaScript-технологій. Актуальність теми обумовлена зростанням популярності цифрового літературного контенту, розвитком веб-технологій та необхідністю створення зручних програмних засобів для взаємодії між авторами й читачами в мережі Інтернет.

У роботі проведено аналіз предметної області та досліджено особливості функціонування сучасних платформ для публікації веб-новел. Виконано порівняльний аналіз існуючих програмних рішень, визначено їх переваги та недоліки, сформовано функціональні та нефункціональні вимоги до розроблюваної системи. На основі результатів аналізу визначено основних користувачів платформи, побудовано UML-діаграми варіантів використання та класів, а також сформовано модель взаємодії компонентів програмної системи.

У процесі виконання роботи спроєктовано архітектуру веб-платформи на основі клієнт-серверного підходу. Для реалізації клієнтської частини використано бібліотеку React, що забезпечує створення сучасного адаптивного користувацького інтерфейсу. Серверну частину реалізовано із застосуванням середовища Node.js та фреймворку Express.js. Для зберігання даних використано

систему керування базами даних PostgreSQL. Реалізовано REST API для організації взаємодії між клієнтською та серверною частинами системи.

Розроблене програмне забезпечення забезпечує реєстрацію та авторизацію користувачів, створення і редагування веб-новел, публікацію розділів, читання літературного контенту, коментування та оцінювання творів, формування списку вибраного, а також виконання адміністративних функцій з модерації контенту. Проведено тестування та верифікацію програмної системи, результати яких підтвердили коректність реалізації функціональних можливостей і відповідність системи поставленим вимогам.

Практичне значення роботи полягає у створенні програмного забезпечення, яке може бути використане як основа для побудови сучасних веб-сервісів публікації та поширення цифрового літературного контенту, а також розширене шляхом інтеграції додаткових функціональних модулів.

ABSTRACT

Software Development of a Web Platform for Publishing and Reading Web Novels Using the JavaScript Programming Language // Bachelor's Qualification Thesis // Podeiko Maksym // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, Group SP-42 // Ternopil, 2026 // P. 110, fig/ 15, tabl. 38, chair. 11, app. 3, bibl. 35.

Keywords: web platform, web novel, JavaScript, React, Node.js, Express.js, PostgreSQL, REST API, web application, software.

The bachelor's qualification thesis is devoted to the development of software for a web platform intended for publishing and reading web novels using modern JavaScript technologies. The relevance of the research is determined by the growing popularity of digital literary content, the rapid development of web technologies, and the need for convenient software tools that facilitate interaction between authors and readers via the Internet.

The thesis includes an analysis of the subject area and investigates the specific features of modern web-novel publishing platforms. A comparative analysis of existing software solutions was carried out, their advantages and disadvantages were identified, and both functional and non-functional requirements for the proposed system were formulated. Based on the results of the analysis, the main categories of platform users were identified, UML use-case and class diagrams were developed, and a model of interaction between the software system components was designed.

During the implementation stage, the architecture of the web platform was designed using a client-server approach. The client side was developed using the React library, which provides a modern and adaptive user interface. The server side was implemented using the Node.js runtime environment and the Express.js framework. PostgreSQL was selected as the database management system for data storage. A REST API was developed to organize communication between the client and server components of the system.

The developed software provides user registration and authentication, creation and editing of web novels, publication of chapters, reading of literary content, commenting and rating of works, management of favorite lists, and administrative content moderation functions. Software testing and verification were conducted, and the obtained results confirmed the correctness of the implemented functionality and the compliance of the system with the specified requirements.

The practical significance of the thesis lies in the development of software that can serve as a foundation for modern web services aimed at publishing and distributing digital literary content. Furthermore, the proposed solution can be extended through the integration of additional functional modules and services.

ЗМІСТ

ВСТУП

У сучасних умовах розвитку інформаційних технологій веб-застосунки стали одним із ключових інструментів створення, поширення та споживання цифрового контенту. Активний розвиток мережі Інтернет, збільшення кількості користувачів мобільних пристроїв та популяризація онлайн-сервісів сприяють постійному зростанню кількості веб-платформ різного призначення.

Одним із перспективних напрямів розвитку сучасних веб-технологій є платформи для публікації та читання веб-новел. Веб-новели являють собою літературні твори, що публікуються в електронному вигляді у мережі Інтернет та розповсюджуються у форматі окремих розділів або епізодів. На відміну від традиційних друкованих видань, веб-новели характеризуються інтерактивністю, можливістю регулярного оновлення контенту та активною взаємодією між авторами і читачами.

Популярність веб-новел значною мірою зумовлена розвитком спеціалізованих платформ, серед яких найбільш відомими є Wattpad, Webnovel та Royal Road. Дані системи забезпечують можливість публікації літературного контенту, організації структури творів, коментування, оцінювання та формування читацьких спільнот. Разом із тим існуючі програмні рішення мають ряд недоліків, серед яких перевантаженість інтерфейсу, складність навігації, недостатня локалізація та обмежені можливості адаптації функціональності під потреби користувачів.

У зв’язку з цим актуальною задачею є розробка сучасної веб-платформи для публікації та читання веб-новел, яка забезпечує зручну взаємодію між авторами та читачами, підтримує адаптивний інтерфейс користувача, ефективну систему управління контентом та стабільну роботу програмного забезпечення.

Для реалізації програмної системи використано мову програмування JavaScript та сучасні веб-технології на її основі, зокрема React, Node.js і Express.js. Використання JavaScript дозволяє реалізувати як клієнтську, так і серверну

частини програмної системи в межах єдиного технологічного стеку, що спрощує процес розробки, тестування та подальшої підтримки програмного забезпечення.

Актуальність теми полягає у необхідності створення сучасної веб-платформи для публікації та читання веб-новел, яка забезпечує ефективне управління цифровим літературним контентом, зручну взаємодію між користувачами та підтримку адаптивного інтерфейсу. Використання мови програмування JavaScript та сучасних веб-технологій дозволяє створити масштабовану та продуктивну систему, придатну для подальшого розвитку й удосконалення.

Мета роботи. Метою кваліфікаційної роботи є розробка програмного забезпечення веб-платформи для публікації та читання веб-новел із використанням мови програмування JavaScript, що забезпечує створення, зберігання, публікацію та перегляд літературного контенту, а також взаємодію між авторами та читачами.

Завдання дослідження. Для досягнення поставленої мети необхідно вирішити такі основні завдання:

1. Провести аналіз предметної області веб-платформ для публікації та читання веб-новел з метою визначення особливостей їх функціонування та основних вимог до програмного забезпечення.

2. Проаналізувати існуючі програмні рішення для публікації цифрового літературного контенту з метою виявлення їх переваг, недоліків та можливостей удосконалення.

3. Сформувати функціональні та нефункціональні вимоги до веб-платформи для визначення її структури, функціональності та якісних характеристик.

4. Виконати проєктування архітектури програмної системи для забезпечення ефективної взаємодії між клієнтською та серверною частинами веб-платформи.

5. Розробити структуру реляційної бази даних для організації зберігання інформації про користувачів, веб-новели, розділи, коментарі та оцінки.

6. Реалізувати клієнтську частину веб-платформи засобами JavaScript та React для забезпечення адаптивного й зручного інтерфейсу користувача.

7. Реалізувати серверну частину системи із використанням Node.js та Express.js для обробки запитів, керування контентом і взаємодії з базою даних.

8. Реалізувати функціональні модулі публікації, редагування та читання веб-новел для забезпечення повноцінної роботи користувачів із літературним контентом.

9. Реалізувати механізми взаємодії користувачів шляхом впровадження системи коментування, оцінювання та формування списку вибраних новел.

10. Провести тестування та верифікацію програмного забезпечення для підтвердження його працездатності та відповідності поставленим вимогам.

Об’єкт дослідження. Об'єктом дослідження є процес функціонування веб-платформ для публікації, зберігання та читання цифрового літературного контенту, а також організація взаємодії між авторами та читачами у веб-середовищі.

Предмет дослідження. Предметом дослідження є методи, моделі та програмні засоби розробки веб-платформи для публікації та читання веб-новел із використанням мови програмування JavaScript, клієнт-серверної архітектури, REST API та реляційної бази даних.

Методи дослідження. У роботі використано методи системного та порівняльного аналізу для дослідження предметної області та існуючих програмних рішень; методи об'єктно-орієнтованого проєктування та UML-моделювання для формалізації структури програмної системи; методи проєктування реляційних баз даних для побудови структури зберігання даних; методи клієнт-серверної взаємодії та REST-архітектури для реалізації обміну інформацією між компонентами системи; технології веб-розробки на основі JavaScript для реалізації програмного забезпечення; а також методи модульного, інтеграційного та системного тестування для перевірки працездатності веб-платформи.

Практичне значення отриманих результатів Практичне значення отриманих результатів полягає у розробці програмного забезпечення веб-платформи для публікації та читання веб-новел, що забезпечує створення, редагування, зберігання та перегляд цифрового літературного контенту, а також підтримує взаємодію між авторами та читачами.

У результаті виконання роботи реалізовано програмну систему на основі мови програмування JavaScript із використанням React, Node.js, Express.js та PostgreSQL, яка дозволяє здійснювати реєстрацію та авторизацію користувачів, публікацію веб-новел, додавання розділів, коментування, оцінювання творів та формування персоналізованих списків вибраного контенту.

Розроблене програмне забезпечення може бути використане як основа для створення сучасних веб-сервісів цифрового літературного контенту, а також для подальшого розширення функціональності шляхом інтеграції рекомендаційних систем, мобільних застосунків, аналітичних модулів та засобів персоналізації контенту.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Сучасний розвиток інформаційних технологій сприяє активному поширенню цифрового літературного контенту та появі нових форматів його представлення в мережі Інтернет. Одним із таких форматів є веб-новели, які набувають значної популярності серед авторів і читачів завдяки можливості оперативної публікації та доступності на різних пристроях. Для створення ефективної програмної системи необхідно виконати детальний аналіз предметної області, визначити особливості функціонування існуючих платформ, дослідити потреби потенційних користувачів та сформувати вимоги до майбутнього програмного забезпечення. У цьому розділі проведено аналіз предметної області, визначено основні функціональні та нефункціональні вимоги до системи, досліджено ролі користувачів та побудовано моделі взаємодії із веб-платформою.

1.1 Аналіз предметної області

У сучасних умовах цифровізації суспільства відбувається активне зростання обсягів інформаційного контенту, що поширюється через мережу Інтернет. Особливе місце серед цифрових медіа займають текстові ресурси, які забезпечують зручний доступ до літературних творів, статей та інших форм контенту. Одним із перспективних напрямів розвитку таких ресурсів є веб-новели.

1.1.1 Особливості веб-новел як виду цифрового контенту

У сучасному цифровому суспільстві спостерігається стрімке зростання популярності електронних форматів поширення інформації. Одним із напрямів розвитку цифрового контенту є веб-новели, які являють собою літературні твори, що публікуються у мережі Інтернет у вигляді окремих розділів або епізодів [25, 26, 27].

На відміну від традиційних друкованих видань, веб-новели характеризуються [19, 20, 25]:

- поетапною публікацією контенту;
- можливістю оперативного редагування творів;
- інтерактивною взаємодією між авторами та читачами;
- доступністю через веб-браузер або мобільний пристрій;
- використанням рейтингових систем та коментарів.

Популярність такого формату зумовлена можливістю швидкого поширення літературних творів без необхідності залучення традиційних видавництв.

1.1.2 Аналіз сучасних платформ для публікації веб-новел

Серед найбільш поширених платформ можна виділити:

- Wattpad;
- Webnovel;
- Royal Road.

Дані системи забезпечують [25, 26, 27]:

- створення та публікацію творів;
- структурування контенту за розділами;
- оцінювання та коментування;
- формування читацьких спільнот.

Таблиця 1.1 – Порівняльний аналіз платформ

Критерій	**Wattpad**	**Webnovel**	**Royal Road**
Простота використання	середня	низька	висока
Функціональність	висока	висока	середня
Безкоштовний доступ	так	частково	так
Інтерактивність	висока	середня	середня
Адаптивність	висока	висока	середня
Локалізація	низька	низька	низька

Примітка: Джерело: сформовано автором на основі [25–27].

Разом із тим існуючі рішення мають певні недоліки:

-

- складний інтерфейс користувача;
- надлишкову функціональність;
- недостатню локалізацію;
- складність модерації контенту;
- обмежені можливості персоналізації.

1.1.3 Основні сутності предметної області

Під час аналізу предметної області визначено такі основні сутності системи [3, 4, 5, 6]:

1. Користувач (User) – основна сутність системи, яка може виконувати роль читача, автора або адміністратора.

Основні атрибути:

- ідентифікатор;
- ім'я користувача;
- електронна пошта;
- пароль;
- роль.

2. Веб-новела (Novel) – основна одиниця літературного контенту.

Атрибути:

- назва;
- опис;
- жанр;
- автор;
- дата створення.

3. Розділ (Chapter) – структурний елемент веб-новели.

Атрибути:

- назва;
- текст;
- номер розділу;
- дата публікації.

4. Коментар (Comment) – забезпечує взаємодію між користувачами.

5. Оцінка (Rating) – використовується для визначення популярності новел.

6. Вибране (Favorite) – дозволяє користувачам формувати власну бібліотеку творів.

1.1.4 Основні бізнес-процеси системи

У межах веб-платформи реалізуються такі процеси [1, 2, 13]:

1. Реєстрація користувачів.
2. Авторизація користувачів.
3. Створення новел.
4. Публікація розділів.
5. Читання контенту.
6. Коментування творів.
7. Оцінювання новел.
8. Адміністрування системи.

1.1.5 Проблеми предметної області

Проведений аналіз дозволив виявити такі проблеми [19, 20, 25]:

- складність публікації контенту на існуючих платформах;
- перевантаженість інтерфейсів;
- недостатня адаптивність окремих систем;
- складність пошуку контенту;
- обмежені можливості взаємодії між користувачами.

Усунення зазначених недоліків є одним із головних завдань розроблюваної веб-платформи.

У результаті аналізу предметної області встановлено особливості функціонування веб-платформ для публікації та читання веб-новел, визначено основні сутності та бізнес-процеси системи, а також виявлено недоліки існуючих

рішень. Отримані результати є основою для постановки задачі та формування вимог до програмної системи.

1.2 Постановка завдання та цілей

Проведений аналіз предметної області показав, що сучасні веб-платформи для публікації та читання веб-новел забезпечують широкі функціональні можливості для створення та поширення цифрового літературного контенту. Разом із тим існуючі рішення характеризуються рядом недоліків, зокрема складністю інтерфейсу, надлишковою функціональністю, недостатньою локалізацією та обмеженими можливостями персоналізації контенту.

У зв’язку з цим виникає необхідність розробки програмного забезпечення веб-платформи, яке забезпечуватиме ефективну взаємодію між авторами та читачами, зручну публікацію веб-новел, організацію структури творів та підтримку сучасних механізмів роботи з цифровим контентом.

1.2.1 Формулювання задачі розробки

Основною задачею роботи є розробка програмного забезпечення веб-платформи для публікації та читання веб-новел із використанням мови програмування JavaScript, яке забезпечуватиме [1, 2]:

- створення облікових записів користувачів;
- авторизацію та автентифікацію користувачів;
- публікацію веб-новел;
- створення та редагування розділів;
- організацію зручного читання контенту;
- коментування та оцінювання творів;
- формування персональної бібліотеки користувача;
- адміністрування контенту та користувачів.

Розроблювана система повинна бути орієнтована на використання у веб-середовищі та забезпечувати стабільну роботу на різних типах пристроїв.

1.2.2 Цілі створення програмної системи

Головною метою розробки веб-платформи є створення сучасного програмного продукту, який дозволить авторам публікувати власні твори, а читачам – отримувати зручний доступ до цифрового літературного контенту.

Для досягнення поставленої мети необхідно забезпечити реалізацію таких цілей:

1. Підвищення зручності публікації контенту.

Система повинна надавати авторам простий механізм створення та редагування веб-новел без необхідності використання спеціалізованого програмного забезпечення.

2. Забезпечення комфортного читання.

Інтерфейс користувача повинен забезпечувати швидкий доступ до новел, зручну навігацію між розділами та адаптивне відображення інформації на різних пристроях.

3. Організація взаємодії користувачів.

Платформа повинна підтримувати механізми коментування, оцінювання та додавання новел до списку вибраного.

4. Забезпечення масштабованості системи.

Архітектура програмного забезпечення повинна дозволяти подальше розширення функціональності без суттєвих змін існуючих компонентів [12, 13, 14].

1.2.3 Функціональні вимоги до системи

На основі аналізу предметної області сформовано функціональні вимоги до веб-платформи [1, 2, 13]:

1. Вимоги до роботи з користувачами.

Система повинна забезпечувати:

- реєстрацію користувачів;
- авторизацію користувачів;
- редагування профілю;
- зміну пароля;
- керування ролями користувачів.

2. Вимоги до роботи з веб-новелами.

Система повинна забезпечувати:

- створення новел;
- редагування новел;
- видалення новел;
- перегляд новел;
- сортування та фільтрацію контенту.

3. Вимоги до роботи з розділами.

Система повинна забезпечувати:

- додавання розділів;
- редагування розділів;
- видалення розділів;
- перегляд вмісту розділів.

4. Вимоги до взаємодії користувачів.

Система повинна забезпечувати:

- додавання коментарів;
- редагування власних коментарів;
- оцінювання новел;
- додавання новел до вибраного.

5. Вимоги до адміністрування.

Адміністратор повинен мати можливість:

- переглядати користувачів;
- блокувати користувачів;
- видаляти контент;
- модерувати коментарі.

Таблиця 1.2 – Основні функціональні вимоги системи

№	Функція	Призначення
F1	Реєстрація	Створення облікового запису
F2	Авторизація	Вхід у систему
F3	Публікація новели	Створення твору
F4	Додавання розділів	Наповнення новели контентом
F5	Читання новел	Перегляд літературного контенту
F6	Коментування	Взаємодія між користувачами
F7	Оцінювання	Формування рейтингу
F8	Вибране	Створення персональної бібліотеки
F9	Адміністрування	Керування системою

Джерело: сформовано автором на основі [1, 2, 13].

1.2.4 Нефункціональні вимоги до системи

Окрім функціональних можливостей, система повинна відповідати ряду нефункціональних вимог [12, 15, 29, 30]:

1. Продуктивність – час завантаження основних сторінок не повинен перевищувати декількох секунд за нормального навантаження.

2. Надійність – система повинна забезпечувати стабільну роботу та коректну обробку помилок.

3. Безпека – необхідно забезпечити захист персональних даних користувачів та безпечну авторизацію.

4. Адаптивність – інтерфейс повинен коректно відображатися на персональних комп’ютерах, планшетах та смартфонах.

5. Масштабованість – архітектура системи повинна забезпечувати можливість додавання нових функціональних модулів.

6. Зручність використання – інтерфейс повинен бути інтуїтивно зрозумілим та не потребувати тривалого навчання користувача.

У підрозділі сформульовано основну задачу розробки веб-платформи для публікації та читання веб-новел, визначено цілі створення програмної системи, а також сформовано функціональні та нефункціональні вимоги до програмного забезпечення. Отримані результати є основою для подальшого визначення акторів системи та моделювання варіантів її використання у підрозділі 1.3.

1.3 Аналіз акторів та варіантів використання системи

Одним із важливих етапів проєктування програмного забезпечення є визначення користувачів системи та аналіз способів їх взаємодії з програмним продуктом. Для цього застосовується UML-моделювання [3, 4], яке дозволяє формалізувати функціональні можливості системи та представити їх у вигляді варіантів використання (Use Case) [3].

Аналіз акторів і варіантів використання дає можливість визначити функціональні межі системи, встановити перелік доступних операцій для кожної категорії користувачів та забезпечити узгодженість між вимогами до програмного забезпечення та його майбутньою реалізацією.

1.3.1 Визначення акторів системи

На основі аналізу предметної області встановлено, що у роботі веб-платформи беруть участь три основні категорії користувачів:

- читач;
- автор;
- адміністратор.

Кожна категорія користувачів має власний набір функціональних можливостей та рівень доступу до ресурсів системи.

1. Читач (Reader) – є основним споживачем літературного контенту, опублікованого на платформі.

Основними функціями читача є:

- реєстрація у системі;
- авторизація;
- перегляд каталогу веб-новел;
- пошук новел;
- перегляд інформації про твір;
- читання розділів;
- додавання коментарів;

- оцінювання новел;
- додавання творів до списку вибраного.

2. Автор (Author) – є користувачем, який створює та публікує власний літературний контент.

Основними функціями автора є:

- реєстрація та авторизація;
- створення новел;
- редагування новел;
- видалення власних новел;
- створення розділів;
- редагування розділів;
- перегляд статистики власних творів;
- взаємодія з коментарями читачів.

Автор також може використовувати всі функції, доступні читачу.

3. Адміністратор (Administrator) – відповідає за підтримку працездатності системи та контроль опублікованого контенту.

Основними функціями адміністратора є:

- керування користувачами;
- блокування облікових записів;
- модерація коментарів;
- видалення неприйнятного контенту;
- керування жанрами;
- моніторинг роботи системи.

1.3.2 Аналіз ролей користувачів

Для забезпечення безпеки та коректного розподілу функціональності використовується рольова модель доступу.

Таблиця 1.3 – Розподіл функцій між акторами системи

Функція	**Читач**	**Автор**	**Адміністратор**
Реєстрація	+	+	–
Авторизація	+	+	+
Читання новел	+	+	+
Коментування	+	+	+
Оцінювання	+	+	+
Додавання у вибране	+	+	+
Створення новел	–	+	+
Редагування новел	–	+	+
Додавання розділів	–	+	+
Видалення контенту	–	–	+
Блокування користувачів	–	–	+

Примітка: Джерело: сформовано автором на основі [3, 4].

1.3.3 Визначення варіантів використання системи

Варіанти використання (Use Cases) описують сценарії взаємодії користувачів із системою та відображають функціональні можливості програмного забезпечення.

Для веб-платформи визначено такі основні варіанти використання:

1. Загальні варіанти використання

- реєстрація користувача;
- авторизація користувача;
- перегляд каталогу новел;
- пошук новел;
- перегляд інформації про новелу.

2. Варіанти використання читача

- читання розділів;
- коментування новел;
- оцінювання творів;
- додавання новел до вибраного.

3. Варіанти використання автора

- створення новели;
- редагування новели;
- створення розділу;

- редагування розділу;
- перегляд власних публікацій.

4. Варіанти використання адміністратора

- перегляд списку користувачів;
- модерація коментарів;
- видалення контенту;
- блокування користувачів.

1.3.4 UML-діаграма варіантів використання

Для графічного представлення взаємодії користувачів із програмною системою побудовано UML-діаграму варіантів використання [3, 4].

У діаграмі відображено:

- основних акторів системи;
- варіанти використання;
- зв'язки між користувачами та функціями;
- межі програмної системи.

Побудована діаграма дозволяє наочно представити функціональність веб-платформи та визначити вимоги до її реалізації.

Рисунок 1.1 – UML-діаграма варіантів використання веб-платформи для публікації та читання веб-новел

1.3.5 Обґрунтування вибору функціональності

Обрана функціональність веб-платформи забезпечує реалізацію основних потреб користувачів та відповідає результатам аналізу предметної області.

Запропонований набір функцій дозволяє:

- авторам публікувати власні твори;
- читачам зручно отримувати доступ до контенту;
- адміністраторам підтримувати порядок та безпеку системи.

Використання рольового підходу до організації доступу забезпечує захист даних та спрощує адміністрування веб-платформи.

У підрозділі визначено основних акторів веб-платформи, проаналізовано їх ролі та функціональні можливості. Сформовано перелік варіантів використання системи та побудовано UML-діаграму Use Case, яка відображає взаємодію користувачів із програмним забезпеченням. Отримані результати є основою для детального опису ключових сценаріїв роботи системи, що розглядаються у підрозділі 1.4.

1.4 Опис ключових варіантів використання

Після визначення основних акторів системи та побудови UML-діаграми варіантів використання необхідно детально описати найбільш важливі сценарії взаємодії користувачів із веб-платформою. Такий опис дозволяє формалізувати функціональні вимоги до програмної системи та забезпечує основу для подальшого проєктування архітектури і реалізації програмного забезпечення [3, 4].

Для веб-платформи публікації та читання веб-новел ключовими є сценарії, пов’язані з реєстрацією користувачів, публікацією контенту, читанням творів та взаємодією між користувачами.

1.4.1 Варіант використання «Реєстрація користувача»

Мета: надання новому користувачу можливості створити обліковий запис для подальшого використання функцій веб-платформи.

Актор: читач або автор.

Передумови: користувач не має облікового запису в системі.

Основний сценарій:

1. Користувач відкриває сторінку реєстрації.
2. Вводить ім’я користувача.
3. Вводить адресу електронної пошти.
4. Вводить пароль.
5. Підтверджує введені дані.
6. Система виконує перевірку коректності введених даних.
7. Створюється новий обліковий запис.
8. Користувач отримує доступ до системи.

Результат: новий користувач зареєстрований у системі.

1.4.2 Варіант використання «Авторизація користувача»

Мета: надання доступу до персоналізованих функцій системи.

Актор: читач, автор або адміністратор.

Передумови: користувач має зареєстрований обліковий запис.

Основний сценарій:

1. Користувач відкриває сторінку входу.
2. Вводить електронну пошту та пароль.
3. Система перевіряє облікові дані.
4. У разі успішної перевірки відкривається особистий кабінет користувача.

Альтернативний сценарій: якщо пароль або електронна пошта введені неправильно, система відображає повідомлення про помилку.

Результат: користувач авторизований у системі.

1.4.3 Варіант використання «Створення новели»

Мета: надання автору можливості створити новий літературний твір.

Актор – автор.

Передумови: автор успішно авторизований у системі.

Основний сценарій:

1. Автор переходить до панелі керування.
2. Обирає функцію створення новели.
3. Вводить назву твору.
4. Заповнює опис.
5. Обирає жанр.
6. Завантажує обкладинку (за потреби).
7. Зберігає новелу.

Результат: у системі створено нову веб-новелу.

1.4.4 Варіант використання «Додавання розділу»

Мета – надання автору можливості публікувати нові розділи веб-новели.

Актор – автор.

Передумови:

1. У системі існує новела, створена автором.
2. Основний сценарій
3. Автор відкриває сторінку редагування новели.
4. Обирає функцію додавання розділу.
5. Вводить назву розділу.
6. Вводить текст розділу.
7. Зберігає зміни.
8. Система публікує новий розділ.

Результат – новий розділ стає доступним для читачів.

1.4.5 Варіант використання «Читання веб-новели»

Мета: надання читачу доступу до літературного контенту.

Актор – читач.

Передумови:

1. У системі опубліковано хоча б одну веб-новелу.
2. Основний сценарій

3. Користувач відкриває каталог новел.

4. Виконує пошук або обирає потрібний твір.

5. Переглядає інформацію про новелу.

6. Відкриває необхідний розділ.

7. Ознайомлюється зі змістом твору.

8. За необхідності переходить до наступного розділу.

Результат – користувач отримує доступ до контенту веб-новели.

1.4.6 Варіант використання «Коментування та оцінювання новели»

Мета – забезпечення взаємодії між читачами та авторами.

Актор – читач.

Передумови:

1. Користувач авторизований у системі.

2. Основний сценарій

3. Читач відкриває сторінку новели.

4. Переходить до секції коментарів.

5. Вводить текст повідомлення.

6. Надсилає коментар.

7. За потреби виставляє оцінку твору.

8. Система зберігає інформацію у базі даних.

Результат – коментар та оцінка відображаються на сторінці новели.

1.4.7 Варіант використання «Модерація контенту»

Метою є забезпечення контролю якості контенту та дотримання правил платформи.

Актор – адміністратор.

Передумови:

1. Адміністратор авторизований у системі.

2. Основний сценарій

3. Адміністратор відкриває панель керування.

4. Переглядає список новел або коментарів.

5. Аналізує контент.

6. За необхідності видаляє неприйнятний матеріал.

7. Зберігає зміни.

Результат – у системі підтримується актуальний та безпечний контент.

Таблиця 1.3 – Ключові варіанти використання веб-платформи

№	**Варіант використання**	**Актор**
UC-01	Реєстрація користувача	Читач, Автор
UC-02	Авторизація користувача	Усі ролі
UC-03	Створення новели	Автор
UC-04	Додавання розділу	Автор
UC-05	Читання веб-новели	Читач
UC-06	Коментування новели	Читач
UC-07	Оцінювання новели	Читач
UC-08	Модерація контенту	Адміністратор

У підрозділі детально описано ключові варіанти використання веб-платформи для публікації та читання веб-новел. Для кожного сценарію визначено акторів, передумови, послідовність виконання дій та очікувані результати. Проведений аналіз дозволив формалізувати основні функціональні можливості системи та підготувати основу для проєктування її архітектури, структури бази даних і програмної реалізації, що розглядаються у другому розділі кваліфікаційної роботи.

1.5 Обґрунтування вибору технологій та засобів реалізації

Після визначення вимог до програмної системи та аналізу основних сценаріїв її використання важливим етапом є вибір технологій, які забезпечать реалізацію поставлених функціональних завдань. Вибір технологічного стеку безпосередньо впливає на продуктивність, масштабованість, зручність супроводу та подальший розвиток програмного забезпечення.

Для розробки веб-платформи публікації та читання веб-новел обрано сучасний стек веб-технологій, який включає мову програмування JavaScript, бібліотеку React, серверну платформу Node.js, фреймворк Express.js та систему керування базами даних PostgreSQL.

1.5.1 Аналіз мов програмування для веб-розробки

Сучасна веб-розробка використовує значну кількість мов програмування, серед яких найбільш поширеними є:

- JavaScript;
- PHP;
- Python;
- Java;
- C#.

Кожна з перелічених мов має власні переваги та сферу застосування.

PHP традиційно використовується для створення серверних веб-застосунків та підтримується більшістю хостинг-провайдерів. Проте його використання потребує застосування різних мов програмування для клієнтської та серверної частин системи.

Python характеризується простотою синтаксису та широкими можливостями використання у сфері аналізу даних і машинного навчання. Разом із тим для розробки сучасних інтерактивних веб-інтерфейсів необхідне додаткове використання JavaScript.

Java та C# забезпечують високу продуктивність та надійність корпоративних систем, однак потребують більш складної інфраструктури розробки та розгортання.

У свою чергу JavaScript є універсальною мовою програмування, яка дозволяє реалізувати як клієнтську, так і серверну частини веб-застосунку, що суттєво спрощує процес розробки та супроводу програмного забезпечення.

Таблиця 1.4 – Порівняння мов програмування для веб-розробки

Критерій	**JavaScript**	**PHP**	**Python**	**Java**	**C#**
Frontend-розробка	+	–	–	–	–
Backend-розробка	+	+	+	+	+
Єдина мова для всієї системи	+	–	–	–	–
Велика екосистема бібліотек	+	+	+	+	+
Простота інтеграції веб-сервісів	+	+	+	+	+
Популярність у веб-розробці	висока	висока	висока	середня	середня

1.5.2 Обґрунтування вибору JavaScript

Тема кваліфікаційної роботи безпосередньо передбачає використання мови програмування JavaScript для розробки веб-платформи.

Основними перевагами JavaScript є:

- підтримка клієнтської та серверної розробки;
- висока швидкодія сучасних JavaScript-рушіїв;
- підтримка асинхронного програмування;
- велика кількість готових бібліотек;
- активна спільнота розробників;
- можливість створення односторінкових веб-застосунків.

Використання JavaScript дозволяє реалізувати весь програмний комплекс в межах єдиного технологічного стеку, що спрощує підтримку та розвиток програмного продукту.

Звідси зроблено висновок, що JavaScript є найбільш доцільним вибором для реалізації веб-платформи публікації та читання веб-новел.

1.5.3 Обґрунтування вибору React

Для реалізації клієнтської частини системи обрано React.

React є однією з найпоширеніших бібліотек для створення користувацьких веб-інтерфейсів та має ряд переваг:

- компонентний підхід до розробки;
- повторне використання компонентів;
- висока продуктивність завдяки Virtual DOM;

- підтримка адаптивних інтерфейсів;
- велика кількість готових бібліотек.

Використання React дозволяє створити швидкий та зручний інтерфейс користувача, який легко масштабувати та модифікувати.

1.5.4 Обґрунтування вибору Node.js та Express.js

Для реалізації серверної частини веб-платформи використано Node.js та Express.js.

Node.js забезпечує виконання JavaScript-коду на сервері та дозволяє створювати високопродуктивні веб-застосунки.

Основними перевагами Node.js є:

- асинхронна модель обробки запитів;
- висока продуктивність;
- можливість обслуговування великої кількості користувачів;
- використання єдиної мови програмування для frontend та backend.

Для спрощення створення REST API використано Express.js.

Переваги Express.js:

- простота реалізації маршрутів;
- зручна інтеграція з базами даних;
- підтримка middleware;
- гнучкість конфігурації.

Застосування Node.js та Express.js забезпечує ефективну реалізацію серверної логіки веб-платформи.

1.5.5 Обґрунтування вибору PostgreSQL

Для зберігання інформації обрано PostgreSQL.

Під час вибору системи керування базами даних розглядалися:

- PostgreSQL;
- MySQL;

- MongoDB.

Для веб-платформи публікації та читання веб-новел характерна наявність чітко структурованих взаємозв'язків між сутностями:

- користувачами;
- новелами;
- розділами;
- коментарями;
- оцінками.

Саме тому використання реляційної бази даних є більш доцільним порівняно з документно-орієнтованими рішеннями.

Основними перевагами PostgreSQL є:

- підтримка складних зв'язків між таблицями;
- забезпечення цілісності даних;
- висока продуктивність;
- підтримка транзакцій;
- відкритий вихідний код.

З огляду на особливості предметної області PostgreSQL є оптимальним вибором для реалізації сховища даних веб-платформи.

На основі аналізу сучасних технологій веб-розробки та обґрунтовано вибір технологічного стеку для реалізації веб-платформи публікації та читання веб-новел встановлено, що використання мови програмування JavaScript дозволяє реалізувати клієнтську та серверну частини програмної системи в межах єдиного середовища розробки. Для побудови інтерфейсу користувача обрано React, для реалізації серверної логіки — Node.js та Express.js, а для організації зберігання даних — PostgreSQL. Обрані технології забезпечують продуктивність, масштабованість, зручність супроводу та можливість подальшого розвитку програмного забезпечення.

1.6 Висновки до розділу 1

У першому розділі кваліфікаційної роботи проведено аналіз предметної області та сформовано теоретичні засади для розробки програмного забезпечення веб-платформи для публікації та читання веб-новел.

У результаті дослідження встановлено, що веб-новели є сучасною формою цифрового літературного контенту, популярність якої зростає завдяки доступності мережі Інтернет та розвитку спеціалізованих веб-платформ. Проведено аналіз існуючих програмних рішень для публікації та читання веб-новел, що дозволило визначити їх основні функціональні можливості, переваги та недоліки, а також обґрунтувати актуальність створення власної програмної системи.

Визначено основні сутності предметної області, зокрема користувачів, веб-новели, розділи, коментарі та оцінки, а також проаналізовано ключові бізнес-процеси, що забезпечують функціонування веб-платформи. На основі проведеного аналізу сформульовано задачу розробки та визначено цілі створення програмного забезпечення.

У межах моделювання функціональності системи визначено основних акторів веб-платформи, проаналізовано їх ролі та побудовано UML-діаграму варіантів використання. Детальний опис ключових сценаріїв взаємодії користувачів із системою дозволив формалізувати основні функціональні вимоги до програмного забезпечення та встановити перелік функцій, які повинна реалізовувати веб-платформа.

Також у розділі проведено аналіз сучасних технологій веб-розробки та обґрунтовано вибір технологічного стеку для реалізації програмної системи. Встановлено доцільність використання мови програмування JavaScript як основної технології розробки, що забезпечує можливість створення клієнтської та серверної частин веб-застосунку в межах єдиного програмного середовища. Для реалізації інтерфейсу користувача обрано React, для реалізації серверної логіки – Node.js та Express.js, а для організації зберігання даних – PostgreSQL.

Отримані результати дозволили сформувати вимоги до майбутньої програмної системи, визначити підходи до її реалізації та підготувати основу для проєктування архітектури, структури бази даних і програмної реалізації веб-платформи, що розглядаються у другому розділі кваліфікаційної роботи.

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

Після визначення вимог до програмної системи наступним етапом є проєктування її архітектури та реалізація основних програмних компонентів. Якісне проєктування структури системи дозволяє забезпечити її масштабованість, надійність та зручність подальшого супроводу. Особлива увага приділяється розробці архітектури клієнтської та серверної частин, побудові структури бази даних, реалізації механізмів взаємодії між компонентами системи та створенню користувацького інтерфейсу. У даному розділі розглянуто архітектуру веб-платформи, виконано проєктування бази даних, реалізовано серверну та клієнтську частини програмного забезпечення, а також основні функціональні модулі системи.

2.1 Проєктування архітектури веб-платформи

Одним із найважливіших етапів розробки програмного забезпечення є проєктування архітектури системи. Архітектура визначає структуру програмного забезпечення, принципи взаємодії його компонентів та способи організації обробки інформації. Від якості архітектурних рішень залежить продуктивність, масштабованість, надійність та можливість подальшого розвитку програмного продукту.

Під час проєктування веб-платформи для публікації та читання веб-новел необхідно враховувати особливості роботи з великою кількістю користувачів, необхідність зберігання значних обсягів текстового контенту, забезпечення швидкого доступу до даних та підтримку взаємодії між авторами й читачами.

На основі проведеного аналізу вимог до системи та обґрунтування вибору технологічного стеку для реалізації програмного забезпечення обрано клієнт-серверну архітектуру, яка є найбільш поширеним підходом для створення сучасних веб-застосунків.

2.1.1 Вибір архітектури програмної системи

У процесі проєктування розглядалися такі архітектурні підходи:

- монолітна архітектура;
- багаторівнева архітектура;
- клієнт-серверна архітектура;
- мікросервісна архітектура.

Монолітна архітектура характеризується простотою реалізації, проте зі збільшенням функціональності системи ускладнюється її супровід та масштабування.

Мікросервісна архітектура забезпечує високу масштабованість, однак потребує складнішої інфраструктури та є надлишковою для реалізації веб-платформи даного масштабу.

З огляду на функціональні вимоги та обсяг проєкту найбільш доцільним рішенням є використання клієнт-серверної архітектури [10, 11, 12].

Основними перевагами такого підходу є:

- розподіл функціональності між компонентами системи;
- спрощення супроводу програмного забезпечення;
- можливість незалежного розвитку клієнтської та серверної частин;
- підтримка різних типів клієнтських пристроїв;
- ефективне використання ресурсів сервера.

2.1.2 Загальна структура веб-платформи

Архітектура веб-платформи складається з трьох основних рівнів:

1. Клієнтський рівень (Frontend);
2. Серверний рівень (Backend);
3. Рівень зберігання даних (Database Layer).

Загальна архітектура системи наведена на рисунку 2.1.

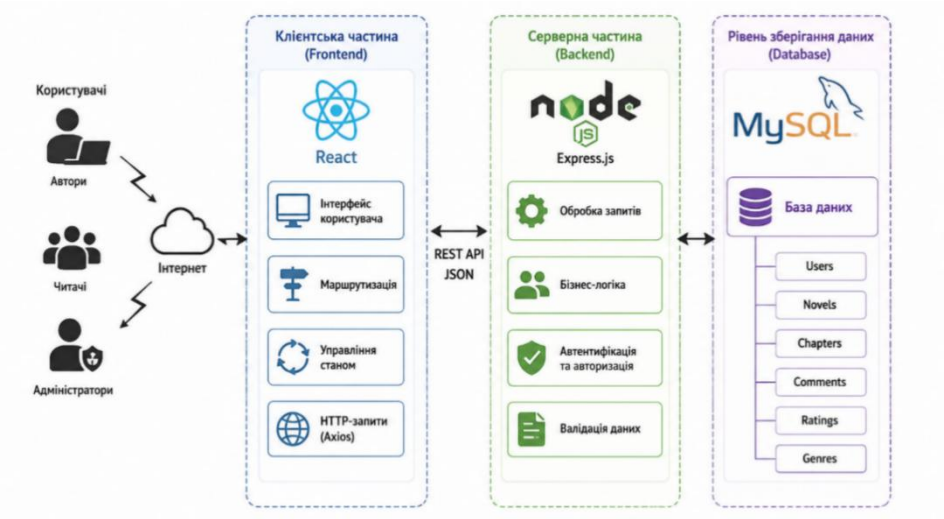

Рисунок 2.1 – Архітектура веб-платформи для публікації та читання веб-новел

У запропонованій архітектурі клієнтська частина взаємодіє із сервером за допомогою HTTP-запитів через REST API. Серверна частина здійснює обробку запитів користувачів, виконує бізнес-логіку та забезпечує взаємодію із базою даних.

2.1.3 Клієнтська частина системи

Клієнтська частина реалізується із використанням мови програмування JavaScript та бібліотеки React [9, 21].

Основним призначенням frontend-рівня є забезпечення взаємодії користувача із системою.

До функцій клієнтської частини належать:

- відображення інтерфейсу користувача;
- навігація між сторінками;
- пошук веб-новел;
- перегляд інформації про твори;
- читання розділів;
- створення коментарів;
- надсилання запитів до сервера.

Структура клієнтської частини побудована за компонентним принципом, що дозволяє повторно використовувати окремі елементи інтерфейсу та спрощує супровід програмного забезпечення.

Основними компонентами клієнтської частини є:

- головна сторінка;
- сторінка авторизації;
- сторінка реєстрації;
- каталог новел;
- сторінка новели;
- сторінка читання розділів;
- панель автора;
- панель адміністратора.

2.1.4 Серверна частина системи

Серверна частина реалізується за допомогою Node.js [10, 22] та Express.js [10, 23].

Основним завданням backend-рівня є забезпечення бізнес-логіки програмної системи.

Сервер виконує такі функції:

- обробку HTTP-запитів;
- авторизацію користувачів;
- перевірку прав доступу;
- управління контентом;
- обробку коментарів та оцінок;
- формування відповідей для клієнтської частини.

Взаємодія між клієнтською та серверною частинами здійснюється через REST API.

Для обміну інформацією використовується формат JSON, який забезпечує просту інтеграцію між компонентами системи.

2.1.5 Рівень зберігання даних

Для зберігання інформації використовується система керування базами даних PostgreSQL.

Основними причинами вибору PostgreSQL є:

- підтримка складних зв'язків між таблицями;
- забезпечення цілісності даних;
- підтримка транзакцій;
- висока продуктивність;
- можливість масштабування.

У базі даних зберігається інформація про:

- користувачів;
- ролі користувачів;
- веб-новели;
- розділи;
- жанри;
- коментарі;
- оцінки;
- список вибраних творів.

2.1.6 Основні модулі веб-платформи

Функціональність системи реалізується у вигляді окремих програмних модулів.

1. Модуль управління користувачами, який забезпечує:

- реєстрацію;
- авторизацію;
- редагування профілю;
- керування ролями.

2. Модуль управління веб-новелами використано для забезпечення:

- створення новел;
- редагування інформації про твори;

– перегляд каталогу новел.

3. Модуль управління розділами використовується для:

– додавання нових розділів;

– редагування текстів;

– перегляд структури твору.

4. Модуль взаємодії користувачів реалізовано для:

– коментування;

– оцінювання;

– формування списку вибраного.

5. Адміністративний модуль використано з метою:

– модерації контенту;

– блокування користувачів;

– керування довідниками системи.

2.1.7 Переваги запропонованої архітектури

Запропонована архітектура має ряд переваг:

– використання єдиної мови програмування JavaScript для frontend та backend;

– можливість незалежного розвитку компонентів системи;

– спрощення тестування програмного забезпечення;

– підтримка масштабування системи;

– висока продуктивність роботи веб-платформи;

– зручність супроводу та модернізації.

Крім того, використання сучасного стеку React–Node.js–PostgreSQL забезпечує відповідність розроблюваної системи сучасним тенденціям веб-розробки.

У підрозділі виконано проєктування архітектури веб-платформи для публікації та читання веб-новел. Обґрунтовано вибір клієнт-серверної архітектури та визначено основні компоненти програмної системи. Розглянуто структуру клієнтської та серверної частин, а також рівень зберігання даних. Визначено

основні функціональні модулі веб-платформи та описано їх призначення. Запропонована архітектура забезпечує масштабованість, продуктивність та можливість подальшого розвитку програмного забезпечення.

2.2 Проєктування структури бази даних

Одним із ключових етапів розробки веб-платформи для публікації та читання веб-новел є проєктування структури бази даних. Саме база даних забезпечує централізоване зберігання інформації про користувачів, літературні твори, розділи, коментарі та інші об'єкти системи.

Якість проєктування бази даних безпосередньо впливає на продуктивність програмної системи, швидкість доступу до інформації, цілісність даних та можливість подальшого розширення функціональності веб-платформи.

З огляду на наявність чітко структурованих взаємозв'язків між сутностями предметної області для реалізації системи обрано реляційну модель даних із використанням СКБД PostgreSQL.

2.2.1 Аналіз інформаційних потоків системи

Під час функціонування веб-платформи відбувається постійний обмін інформацією між користувачами та програмною системою.

Основними інформаційними потоками є:

- реєстрація та авторизація користувачів;
- створення та редагування веб-новел;
- додавання розділів;
- перегляд контенту;
- збереження коментарів;
- виставлення оцінок;
- формування списків вибраного.

Аналіз інформаційних потоків дозволив визначити перелік основних сутностей бази даних та зв'язки між ними.

2.2.2 Визначення сутностей бази даних

На основі проведеного аналізу предметної області встановлено, що структура бази даних повинна містити такі основні сутності:

– Users. Таблиця Users надає інформацію про зареєстрованих користувачів системи.

Таблиця 2.1 – Основні атрибути сутності Users

Поле	**Тип даних**	**Опис**
user_id	SERIAL	Унікальний ідентифікатор
username	VARCHAR(50)	Ім'я користувача
email	VARCHAR(100)	Електронна пошта
password_hash	VARCHAR(255)	Хеш пароля
role_id	INTEGER	Роль користувача
created_at	TIMESTAMP	Дата створення

– Roles. Таблиця Roles використовується для реалізації рольової моделі доступу. Основні ролі: Reader; Author; Administrator.

– Novels. Основна інформація про веб-новели наведено у таблиці 2.2.

Таблиця 2.2 – Основні атрибути сутності Novels

Поле	**Тип даних**
novel_id	SERIAL
title	VARCHAR(255)
description	TEXT
cover_image	VARCHAR(255)
author_id	INTEGER
genre_id	INTEGER
created_at	TIMESTAMP

– Chapters. Таблиця Chapters призначена для зберігання розділів веб-новел.

Таблиця 2.3 – Основні атрибути сутності Chapters

Поле	**Тип даних**
chapter_id	SERIAL
novel_id	INTEGER
chapter_number	INTEGER
title	VARCHAR(255)
content	TEXT
created_at	TIMESTAMP

– Genres. Таблиця Genres забезпечує класифікацію веб-новел.

Таблиця 2.4 – Основні атрибути сутності Genres

Поле	**Тип даних**
genre_id	SERIAL
genre_name	VARCHAR(100)

– Comments. Таблиця Comments призначена для зберігання коментарів користувачів.

Таблиця 2.5 – Основні атрибути сутності Comments:

Поле	**Тип даних**
comment_id	SERIAL
user_id	INTEGER
novel_id	INTEGER
content	TEXT
created_at	TIMESTAMP

– Ratings. Таблиця Ratings містить інформацію про оцінки новел.

Таблиця 2.6 – Основні атрибути сутності Ratings

Поле	**Тип даних**
rating_id	SERIAL
user_id	INTEGER
novel_id	INTEGER
rating_value	INTEGER

– Favorites. Таблиця Favorites використовується для формування персональної бібліотеки користувача.

Таблиця 2.7 – Основні атрибути сутності Favorites

Поле	**Тип даних**
favorite_id	SERIAL
user_id	INTEGER
novel_id	INTEGER

Кожна із зазначених сутностей відповідає окремому об'єкту предметної області.

2.2.3 Проєктування зв'язків між сутностями

Під час проєктування структури бази даних визначено основні типи зв'язків між таблицями.

2.2.3.1 Зв'язок Users – Novels

Один автор може створити багато новел.

Тип зв'язку: One-to-Many (1:M).

2.2.3.2 Зв'язок Novels – Chapters

Одна новела може містити багато розділів.

Тип зв'язку: One-to-Many (1:M).

2.2.3.3 Зв'язок Genres – Novels

Один жанр може бути присвоєний багатьом новелам.

Тип зв'язку: One-to-Many (1:M).

2.2.3.4 Зв'язок Users – Comments

Один користувач може залишати багато коментарів.

Тип зв'язку: One-to-Many (1:M).

2.2.3.5 Зв'язок Users – Novels (через Favorites)

Користувач може додавати багато новел до вибраного.

Новела може бути додана багатьма користувачами.

Тип зв'язку: Many-to-Many (M:N).

2.2.4 Нормалізація структури бази даних

Для усунення дублювання інформації та забезпечення цілісності даних виконано нормалізацію структури бази даних до третьої нормальної форми (3НФ).

Перша нормальна форма (1НФ) – усі таблиці містять атомарні значення та не мають повторюваних груп.

Друга нормальна форма (2НФ) – кожен неключовий атрибут повністю залежить від первинного ключа.

Третя нормальна форма (3НФ) – у таблицях відсутні транзитивні залежності між атрибутами.

Нормалізація дозволила:

- усунути дублювання даних;
- спростити оновлення інформації;
- забезпечити цілісність бази даних;
- підвищити ефективність SQL-запитів.

2.2.5 ER-модель бази даних

Для наочного представлення структури бази даних побудовано ER-діаграму [5, 6].

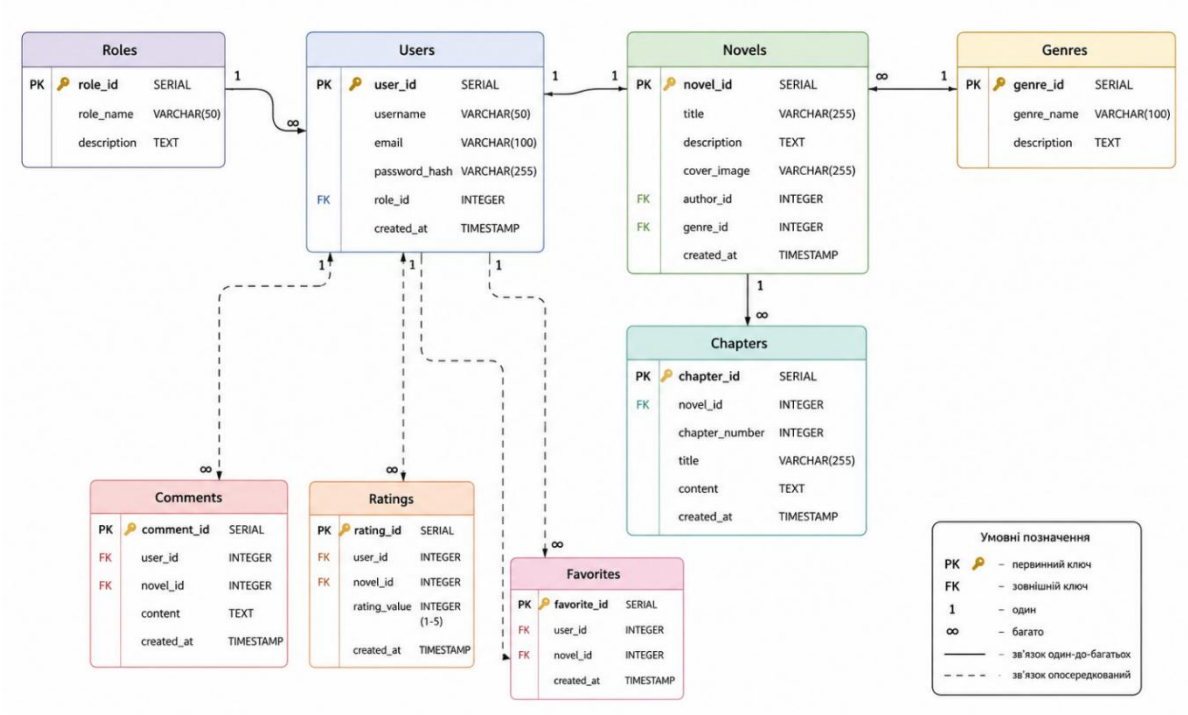

Рисунок 2.2 – ER-діаграма бази даних веб-платформи

На ER-діаграмі відображено:

- сутності предметної області;

- первинні ключі;
- зовнішні ключі;
- зв'язки між таблицями.

ER-модель дозволяє оцінити структуру даних ще до початку програмної реалізації системи та є основою для створення фізичної схеми бази даних.

2.2.6 Переваги розробленої структури бази даних

Запропонована структура бази даних забезпечує:

- зручне зберігання інформації;
- підтримку складних взаємозв'язків між сутностями;
- можливість швидкого пошуку контенту;
- підтримку масштабування системи;
- простоту супроводу програмного забезпечення.

Крім того, використання PostgreSQL [18, 24] дозволяє ефективно працювати з великими обсягами текстового контенту, що є важливим для веб-платформи публікації та читання веб-новел.

У підрозділі виконано проєктування структури бази даних веб-платформи для публікації та читання веб-новел. Визначено основні сутності предметної області, їх атрибути та взаємозв'язки. Проведено нормалізацію структури даних до третьої нормальної форми та побудовано ER-модель бази даних. Запропонована структура забезпечує цілісність інформації, ефективність зберігання даних та можливість подальшого розвитку програмної системи.

2.3 Розробка UML-діаграми класів

UML-діаграма класів є одним із основних засобів об’єктно-орієнтованого моделювання програмних систем. Вона дозволяє відобразити структуру програмного забезпечення, основні класи, їх атрибути, методи та взаємозв’язки між ними [3, 4].

Для веб-платформи публікації та читання веб-новел UML-діаграма класів використовується для формалізації структури основних об’єктів системи, зокрема користувачів, веб-новел, розділів, коментарів, оцінок та списку вибраного.

Побудова UML-діаграми класів дає змогу:

- визначити основні програмні сутності системи;
- встановити зв’язки між класами;
- описати атрибути та методи кожного класу;
- підготувати основу для подальшої програмної реалізації;
- узгодити структуру програмного забезпечення зі структурою бази даних.

2.3.1 Визначення основних класів системи

На основі аналізу предметної області та структури бази даних визначено такі основні класи програмної системи:

- User – описує користувача веб-платформи;

Користувач може виконувати роль читача, автора або адміністратора.

Таблиця 2.8.1 – Основні атрибути класу User

Атрибут	**Тип**	**Опис**
userId	number	Унікальний ідентифікатор користувача
username	string	Ім’я користувача
email	string	Електронна пошта
passwordHash	string	Хеш пароля
roleId	number	Ідентифікатор ролі
createdAt	Date	Дата створення облікового запису

Таблиця 2.8.2 – Основні методи класу User

Метод	**Призначення**
register()	Реєстрація користувача
login()	Авторизація користувача
updateProfile()	Оновлення профілю
changePassword()	Зміна пароля

- Role – використовується для реалізації рольової моделі доступу;

Таблиця 2.9.1 – Атрибути класу Role

Атрибут	**Тип**	**Опис**
roleId	number	Ідентифікатор ролі
roleName	string	Назва ролі

Таблиця 2.9.2 – Методи класу Role

Метод	**Призначення**
assignRole()	Призначення ролі користувачу
checkPermission()	Перевірка прав доступу

– Novel – описує веб-новелу як основну одиницю літературного контенту;

Таблиця 2.10.1 – Атрибути класу Novel

Атрибут	**Тип**	**Опис**
novelId	number	Унікальний ідентифікатор новели
title	string	Назва новели
description	string	Опис новели
coverImage	string	Шлях до зображення обкладинки
authorId	number	Ідентифікатор автора
genreId	number	Ідентифікатор жанру
createdAt	Date	Дата створення

Таблиця 2.10.2 – Методи класу Novel

Метод	**Призначення**
createNovel()	Створення новели
updateNovel()	Редагування новели
deleteNovel()	Видалення новели
getNovelInfo()	Отримання інформації про новелу
getNovelRating()	Отримання рейтингу новели

– Chapter – описує розділ веб-новели;

Таблиця 2.11.1 – Атрибути класу Chapter

Атрибут	**Тип**	**Опис**
chapterId	number	Унікальний ідентифікатор розділу
novelId	number	Ідентифікатор новели
chapterNumber	number	Номер розділу
title	string	Назва розділу
content	string	Текст розділу
createdAt	Date	Дата публікації

Таблиця 2.11.2 – Методи класу Chapter

Метод	**Призначення**
createChapter()	Створення розділу
updateChapter()	Редагування розділу
deleteChapter()	Видалення розділу
getChapterContent()	Отримання тексту розділу

– Genre – використовується для класифікації веб-новел за жанрами;

Таблиця 2.12.1 – Атрибути класу Genre

Атрибут	**Тип**	**Опис**
genreId	number	Унікальний ідентифікатор жанру
genreName	string	Назва жанру

Таблиця 2.12.2 – Методи класу Genre

Метод	**Призначення**
addGenre()	Додавання жанру
updateGenre()	Редагування жанру
deleteGenre()	Видалення жанру

– Comment – описує коментар користувача до новели або розділу;

Таблиця 2.13.1 – Атрибути класу Comment

Атрибут	**Тип**	**Опис**
commentId	number	Унікальний ідентифікатор коментаря
userId	number	Ідентифікатор користувача
novelId	number	Ідентифікатор новели
content	string	Текст коментаря
createdAt	Date	Дата створення коментаря

Таблиця 2.13.2 – Методи класу Comment

Метод	**Призначення**
addComment()	Додавання коментаря
updateComment()	Редагування коментаря
deleteComment()	Видалення коментаря
getComments()	Отримання списку коментарів

– Rating – описує оцінювання веб-новел користувачами;

Таблиця 2.14.1 – Атрибути класу Rating

Атрибут	**Тип**	**Опис**
ratingId	number	Унікальний ідентифікатор оцінки
userId	number	Ідентифікатор користувача
novelId	number	Ідентифікатор новели
ratingValue	number	Значення оцінки

Таблиця 2.14.2 – Методи класу Rating

Метод	**Призначення**
setRating()	Встановлення оцінки
updateRating()	Оновлення оцінки
getAverageRating()	Обчислення середнього рейтингу

– Favorite – використовується для формування персональної бібліотеки користувача..

Таблиця 2.15.1 – Атрибути класу Favorite

Атрибут	**Тип**	**Опис**
favoriteId	number	Унікальний ідентифікатор запису
userId	number	Ідентифікатор користувача
novelId	number	Ідентифікатор новели

Таблиця 2.15.2 – Методи класу Favorite

Метод	**Призначення**
addToFavorites()	Додавання новели до вибраного
removeFromFavorites()	Видалення з вибраного
getFavorites()	Отримання списку вибраних новел

Кожен клас відповідає окремій сутності предметної області та використовується для реалізації відповідної частини функціональності веб-платформи.

2.3.2 Визначення зв’язків між класами

Між класами програмної системи встановлено такі основні зв’язки:

– один користувач може мати одну роль;

– один автор може створювати багато веб-новел;

– одна веб-новела належить до одного жанру;

- одна веб-новела може містити багато розділів;
- один користувач може залишати багато коментарів;
- одна веб-новела може мати багато оцінок;
- один користувач може додавати багато новел до вибраного.

Ці зв’язки відповідають структурі предметної області та забезпечують коректну реалізацію функціональності веб-платформи.

Таблиця 2.16 – Основні зв’язки між класами системи

Клас 1	**Клас 2**	**Тип зв’язку**	**Опис**
Role	User	1:M	Одна роль може належати багатьом користувачам
User	Novel	1:M	Один автор може створити багато новел
Genre	Novel	1:M	Один жанр може містити багато новел
Novel	Chapter	1:M	Одна новела складається з багатьох розділів
User	Comment	1:M	Один користувач може залишати багато коментарів
Novel	Rating	1:M	Одна новела може мати багато оцінок
User	Favorite	1:M	Один користувач може мати багато записів вибраного
Novel	Favorite	1:M	Одна новела може бути додана у вибране багатьма користувачами

2.3.3 UML-діаграма класів веб-платформи

На основі визначених класів, атрибутів, методів та зв’язків між ними побудовано UML-діаграму класів програмної системи.

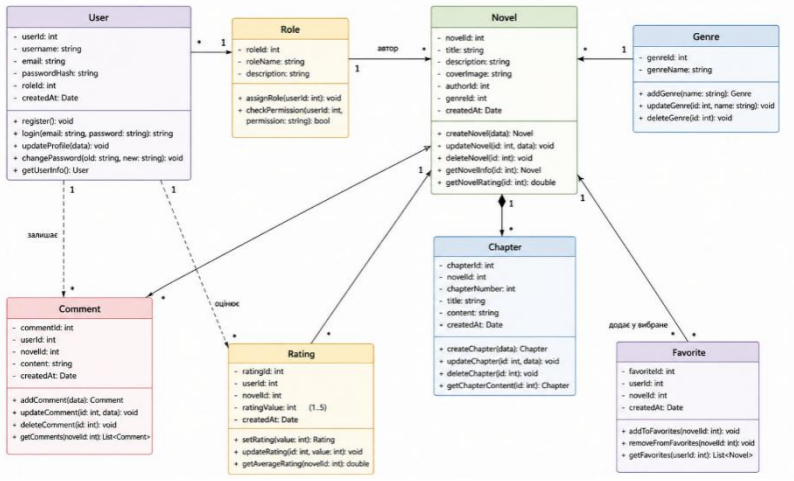

Рисунок 2.3 – UML-діаграма класів веб-платформи для публікації та читання веб-новел

UML-діаграма класів відображає логічну структуру системи та демонструє взаємозв’язки між основними об’єктами програмного забезпечення.

2.3.4 Значення UML-діаграми класів для реалізації системи

Побудована UML-діаграма класів використовується як основа для подальшої програмної реалізації веб-платформи.

Вона дозволяє:

- узгодити структуру програмного забезпечення зі структурою бази даних;
- визначити основні програмні сутності;
- сформувати моделі даних;
- спростити реалізацію серверної логіки;
- забезпечити логічну цілісність системи.

Для теми кваліфікаційної роботи особливо важливо, що UML-діаграма класів демонструє зв’язок між програмною реалізацією та предметною областю веб-новел.

У підрозділі виконано розробку UML-діаграми класів веб-платформи для публікації та читання веб-новел. Визначено основні класи системи, їх атрибути, методи та взаємозв’язки. Побудована UML-модель формалізує структуру програмного забезпечення та забезпечує основу для подальшої реалізації клієнтської і серверної частин веб-платформи.

2.4 Реалізація клієнтської частини веб-платформи

Клієнтська частина веб-платформи забезпечує взаємодію користувачів із програмною системою та відповідає за відображення інформації, навігацію між сторінками, введення даних і передачу запитів до серверної частини. Саме клієнтський рівень формує користувацький досвід та безпосередньо впливає на зручність використання програмного забезпечення.

Для реалізації клієнтської частини веб-платформи використано мову програмування JavaScript та бібліотеку React, що дозволяє створювати сучасні односторінкові веб-застосунки (Single Page Application, SPA).

Застосування React забезпечує компонентний підхід до розробки інтерфейсу, повторне використання програмного коду та високу продуктивність роботи веб-застосунку.

2.4.1 Структура клієнтської частини системи

Клієнтська частина побудована за модульним принципом та складається з окремих компонентів, кожен з яких відповідає за реалізацію певного функціонального блоку системи.

Основні компоненти структури:

- components — багаторазові елементи інтерфейсу;
- pages — сторінки веб-застосунку;
- services — модулі взаємодії з REST API;
- routes — маршрутизація;
- context — керування станом застосунку;
- assets — графічні ресурси;
- styles — файли оформлення інтерфейсу.

Загальну структуру клієнтської частини наведено на рисунку В.1 додатку В.

2.4.2 Реалізація маршрутизації

Для організації навігації між сторінками використовується механізм маршрутизації.

Таблиця 2.17 – Основні маршрути системи

Маршрут	**Призначення**
/	Головна сторінка
/login	Авторизація
/register	Реєстрація
/novels	Каталог новел
/novel/:id	Сторінка новели
/chapter/:id	Сторінка читання розділу
/author	Кабінет автора
/admin	Панель адміністратора

Використання маршрутизації забезпечує швидкий перехід між сторінками без повного перезавантаження веб-застосунку.

Лістинг 2.1 – Приклад налаштування маршрутизації

```
import { BrowserRouter, Routes, Route } from "react-router-dom";

function App() {
    return (
        <BrowserRouter>
            <Routes>
                <Route path="/" element={<HomePage />} />
                <Route path="/login" element={<LoginPage />} />
                <Route path="/register" element={<RegisterPage />}
/>
                <Route path="/novels" element={<NovelListPage />} />
            </Routes>
        </BrowserRouter>
    );
}
```

2.4.3 Реалізація головної сторінки веб-платформи

Головна сторінка є початковою точкою взаємодії користувача із системою.

На головній сторінці відображаються:

- список популярних новел;
- нові публікації;
- навігаційне меню;
- форма пошуку;
- інформація про платформу.

Головна сторінка повинна забезпечувати швидкий доступ до основних функцій системи та бути зрозумілою для нових користувачів.

2.4.4 Реалізація каталогу веб-новел

Каталог новел використовується для перегляду доступного літературного контенту.

Основні функції каталогу:

– перегляд списку творів;
– фільтрація за жанрами;
– сортування за рейтингом;
– пошук за назвою;
– перехід на сторінку новели.

Для відображення інформації використовуються картки новел.

Структура картки новели складається з:

– обкладинки;
– назви твору;
– короткого опису;
– автора;
– рейтингу;
– кількості розділів.

Таке представлення дозволяє користувачу швидко ознайомитися зі змістом новели.

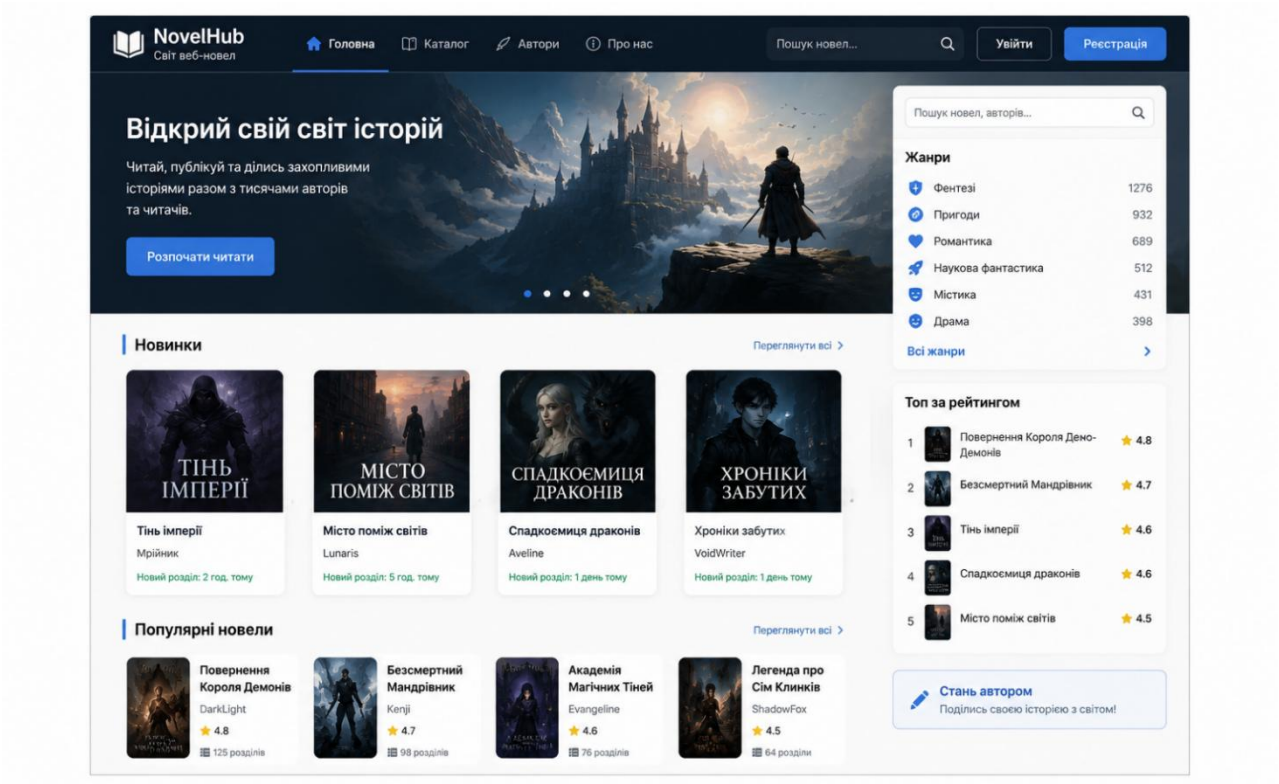

Рисунок 2.5 – Головна сторінка веб-платформи

2.4.5 Реалізація сторінки новели

Сторінка новели призначена для відображення детальної інформації про літературний твір.

На сторінці відображаються:

- назва;
- опис;
- жанр;
- інформація про автора;
- рейтинг;
- список розділів;
- коментарі користувачів.

Користувач також може:

- додати новелу до вибраного;
- залишити коментар;
- виставити оцінку.

2.4.6 Реалізація сторінки читання розділу

Однією з ключових функцій веб-платформи є читання літературного контенту.

Сторінка читання повинна забезпечувати:

- зручне відображення тексту;
- навігацію між розділами;
- адаптацію до різних розмірів екранів;
- мінімізацію відволікаючих елементів інтерфейсу.

Особлива увага приділяється читабельності тексту та комфортності сприйняття інформації.

2.4.7 Реалізація особистого кабінету автора

Особистий кабінет автора призначений для керування власними творами.

Автор отримує можливість:

- створювати новели;
- редагувати інформацію про твори;
- додавати розділи;
- переглядати статистику переглядів;
- керувати коментарями.

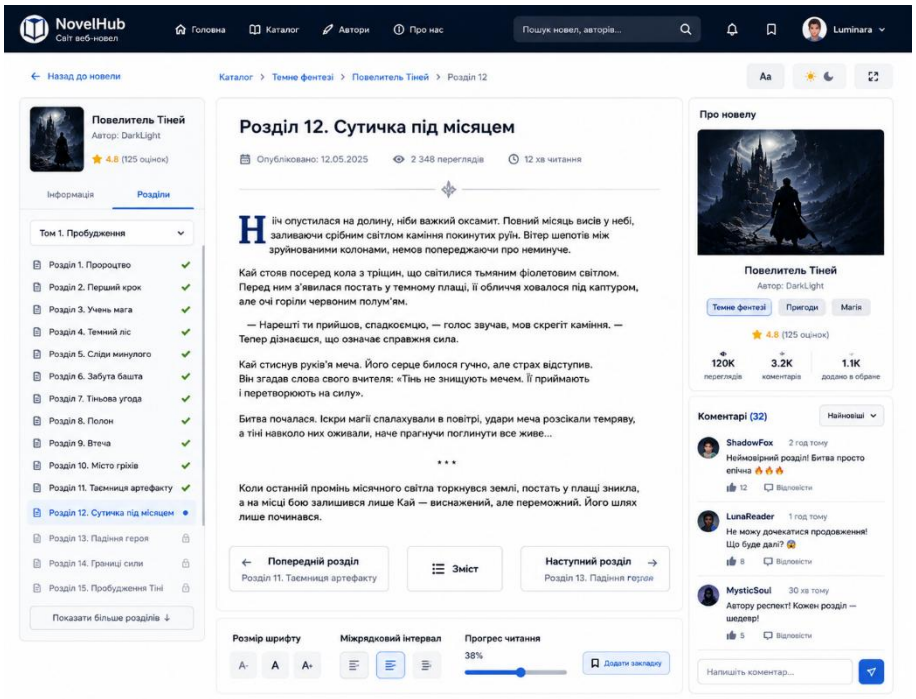

Рисунок 2.6 – Сторінка читання веб-новели

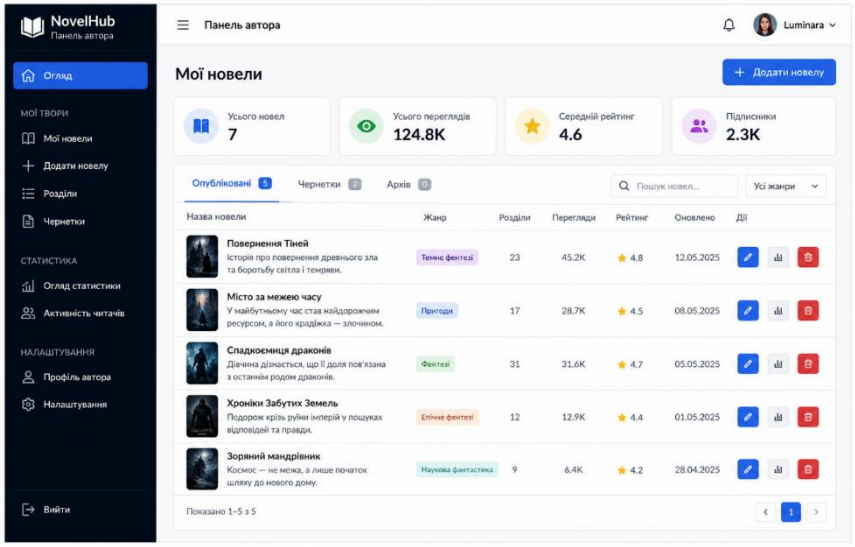

Рисунок 2.7 – Панель автора для управління новелами

2.4.8 Реалізація адаптивного інтерфейсу

Сучасні веб-застосунки повинні коректно працювати на різних типах пристроїв.

Для реалізації адаптивності використовуються:

- Flexbox;
- CSS Grid;
- медіазапити;
- адаптивні компоненти React.
- Інтерфейс оптимізовано для:
- персональних комп'ютерів;
- планшетів;
- смартфонів.

Адаптивний дизайн забезпечує зручність використання веб-платформи незалежно від типу пристрою користувача [19, 20].

2.4.9 Переваги реалізованої клієнтської частини

Розроблена клієнтська частина має такі переваги:

- сучасний інтерфейс користувача;
- висока швидкість роботи;
- адаптивний дизайн;
- компонентна архітектура;
- простота масштабування;
- зручна навігація;
- підтримка сучасних браузерів.

Використання React [9, 21] дозволяє створити ефективний інтерфейс, який легко підтримувати та розширювати в процесі розвитку веб-платформи.

Отже, у підрозділі розглянуто реалізацію клієнтської частини веб-платформи для публікації та читання веб-новел. Описано структуру React-застосунку, механізми маршрутизації, реалізацію основних сторінок системи та принципи побудови адаптивного інтерфейсу користувача. Розроблена клієнтська

частина забезпечує зручну взаємодію користувачів із програмною системою та реалізує всі необхідні функції для роботи з веб-новелами.

2.5 Реалізація серверної частини веб-платформи

Серверна частина програмної системи є центральним компонентом веб-платформи та забезпечує реалізацію бізнес-логіки, обробку запитів користувачів, взаємодію з базою даних та контроль доступу до функціональних можливостей системи.

Для реалізації серверної частини веб-платформи використано мову програмування JavaScript, середовище виконання Node.js [10, 22] та веб-фреймворк Express.js [10, 22].

Вибір зазначених технологій обумовлений можливістю використання єдиного технологічного стеку для реалізації як клієнтської, так і серверної частин системи, що спрощує процес розробки, тестування та подальшого супроводу програмного забезпечення.

2.5.1 Загальна структура серверної частини

Серверна частина веб-платформи реалізована відповідно до принципів багаторівневої архітектури та складається з таких компонентів:

- маршрутизація запитів (Routes);
- контролери (Controllers);
- моделі даних (Models);
- сервіси бізнес-логіки (Services);
- модуль взаємодії з базою даних (Database Layer);
- модуль автентифікації та авторизації.

Загальна структура серверної частини наведена на рисунку 2.6.

Запропонована структура дозволяє розділити відповідальність між окремими компонентами системи та спрощує підтримку програмного забезпечення.

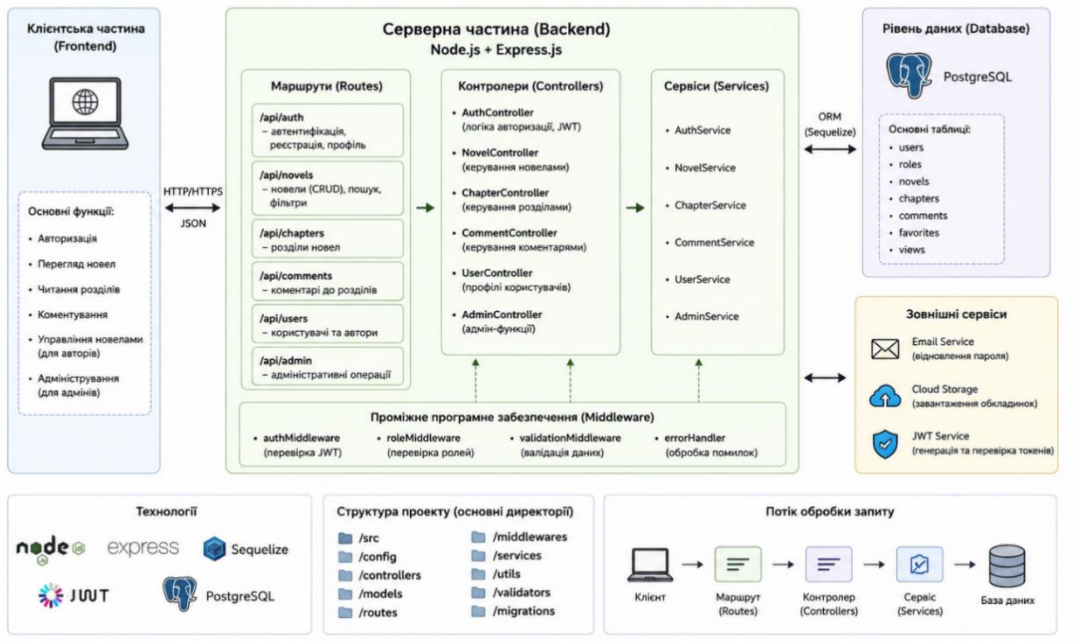

Рисунок 2.8 – Структура серверної частини системи

Запропонована структура дозволяє розділити відповідальність між окремими компонентами системи та спрощує підтримку програмного забезпечення.

2.5.2 Організація REST API

Взаємодія між клієнтською та серверною частинами здійснюється за допомогою REST API [16, 17].

REST API забезпечує передачу даних між компонентами системи за допомогою стандартних HTTP-запитів.

Для реалізації взаємодії використовуються такі HTTP-методи:

1. GET – отримання інформації.
2. POST – створення нових записів.
3. PUT – оновлення інформації.
4. DELETE – видалення записів.

Основними перевагами використання REST API є:

- простота реалізації;
- незалежність клієнтської та серверної частин;

- масштабованість;
- можливість інтеграції з іншими системами.

Лістинг 2.2 – Приклад маршруту отримання списку новел

```
 router.get('/novels', async (req, res) => {
    const novels = await Novel.findAll();
    res.json(novels);
});
```

Даний маршрут забезпечує отримання переліку новел із бази даних та передачу інформації клієнтській частині у форматі JSON.

2.5.3 Реалізація авторизації та автентифікації

Для забезпечення безпечного доступу до ресурсів системи реалізовано механізм автентифікації користувачів [15].

Основними функціями модуля авторизації є:

- реєстрація користувачів;
- вхід у систему;
- перевірка облікових даних;
- формування токенів доступу;
- контроль доступу до ресурсів.

Під час зберігання паролів використовується їх хешування, що дозволяє підвищити рівень безпеки системи.

Лістинг 2.3 – Приклад обробки авторизації

```
 router.post('/login', async (req, res) => {
    const user = await User.findOne({
        where: { email: req.body.email }
    });

    if (!user) {
        return res.status(401).json({
            message: 'User not found'
        });
    }
```

```
    const token = generateToken(user.id);
    return res.json({ token });
});
```

У результаті успішної авторизації користувач отримує токен доступу, який використовується під час подальшої взаємодії із сервером.

2.5.4 Реалізація взаємодії з базою даних

Для роботи з базою даних PostgreSQL використовується рівень моделей даних.

Моделі забезпечують:

- створення записів;
- отримання інформації;
- оновлення даних;
- видалення записів.

Основні операції реалізуються відповідно до концепції CRUD:

- Create;
- Read;
- Update;
- Delete.

Лістинг 2.4 – Приклад створення новели

```
     const novel = await Novel.create({
    title: req.body.title,
    description: req.body.description,
    author_id: req.user.id
});
```

У результаті виконання запиту створюється новий запис у таблиці Novels.

2.5.5 Обробка помилок

Для забезпечення стабільності роботи системи реалізовано механізм централізованої обробки помилок.

Основні типи помилок:

- помилки авторизації;
- помилки валідації даних;
- помилки доступу;
- помилки роботи бази даних;
- внутрішні помилки сервера.

Лістинг 2.5 – Приклад обробника помилок

```
 app.use((err, req, res, next) => {
    res.status(500).json({
        message: err.message
    });
});
```

Централізована обробка дозволяє забезпечити коректне реагування системи на помилки та спростити процес налагодження.

2.5.6 Забезпечення безпеки серверної частини

Для підвищення безпеки веб-платформи реалізовано такі заходи:

- хешування паролів;
- рольовий контроль доступу;
- перевірка коректності введених даних;
- захист маршрутів авторизації;
- контроль доступу до адміністративних функцій.

Застосування зазначених механізмів дозволяє мінімізувати ризик несанкціонованого доступу до ресурсів системи.

2.5.7 Взаємодія клієнтської та серверної частин

Після надсилання запиту клієнтською частиною відбувається така послідовність дій:

1. React-інтерфейс формує HTTP-запит.
2. Запит надходить до Express-сервера.

3. Відповідний маршрут передає запит контролеру.
4. Контролер виконує бізнес-логіку.
5. Відбувається звернення до бази даних.
6. Отримані дані повертаються у форматі JSON.
7. React відображає результат користувачу.

Взаємодію компонентів системи наведено на рисунку 2.7.

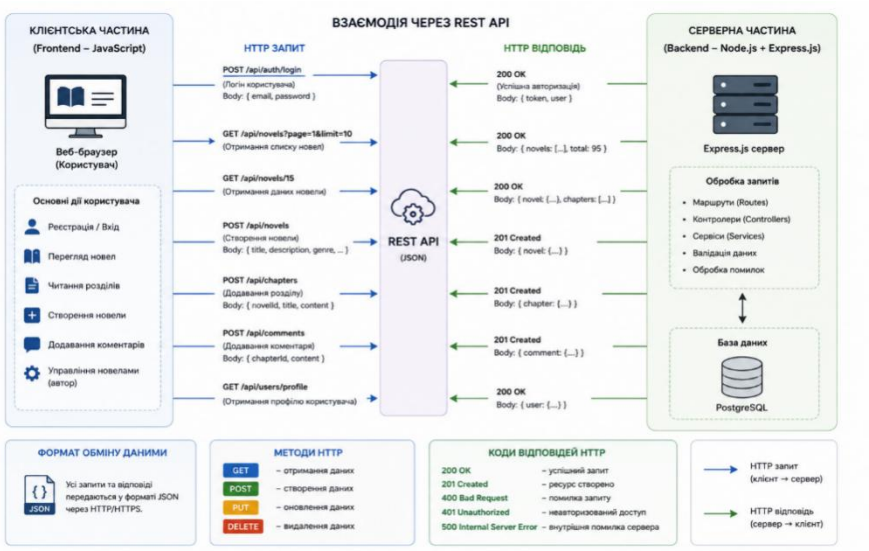

Рисунок 2.9 – Взаємодія клієнтської та серверної частин через REST API

У даному підрозділі розглянуто реалізацію серверної частини веб-платформи для публікації та читання веб-новел. Описано структуру серверного застосунку, принципи організації REST API, механізми авторизації та автентифікації користувачів, взаємодію з базою даних та обробку помилок. Реалізована серверна частина забезпечує виконання бізнес-логіки системи, безпечний доступ до ресурсів та ефективну взаємодію з клієнтською частиною веб-платформи.

2.6 Реалізація функціональних модулів системи

Після проєктування архітектури програмної системи, структури бази даних та реалізації клієнтської і серверної частин виконано розробку основних

функціональних модулів веб-платформи для публікації та читання веб-новел. Кожен модуль реалізує окрему групу функцій та забезпечує виконання бізнес-процесів, визначених під час аналізу предметної області [13, 14].

Модульна структура програмного забезпечення забезпечує гнучкість системи, спрощує супровід програмного коду та дозволяє виконувати подальше розширення функціональності веб-платформи.

2.6.1 Модуль реєстрації та авторизації користувачів

Модуль реєстрації та авторизації є одним із базових компонентів веб-платформи та забезпечує контроль доступу до функціональних можливостей системи.

Основними функціями модуля є:

- створення облікового запису користувача;
- авторизація користувача;
- перевірка облікових даних;
- управління сесією користувача;
- розмежування прав доступу.

Під час реєстрації користувач вводить:

- ім'я користувача;
- адресу електронної пошти;
- пароль.

Після перевірки коректності введених даних інформація зберігається у базі даних.

Паролі користувачів не зберігаються у відкритому вигляді. Перед записом до бази даних виконується хешування пароля, що підвищує рівень безпеки системи.

Після успішної авторизації сервер формує токен доступу, який використовується під час подальшої взаємодії користувача із системою.

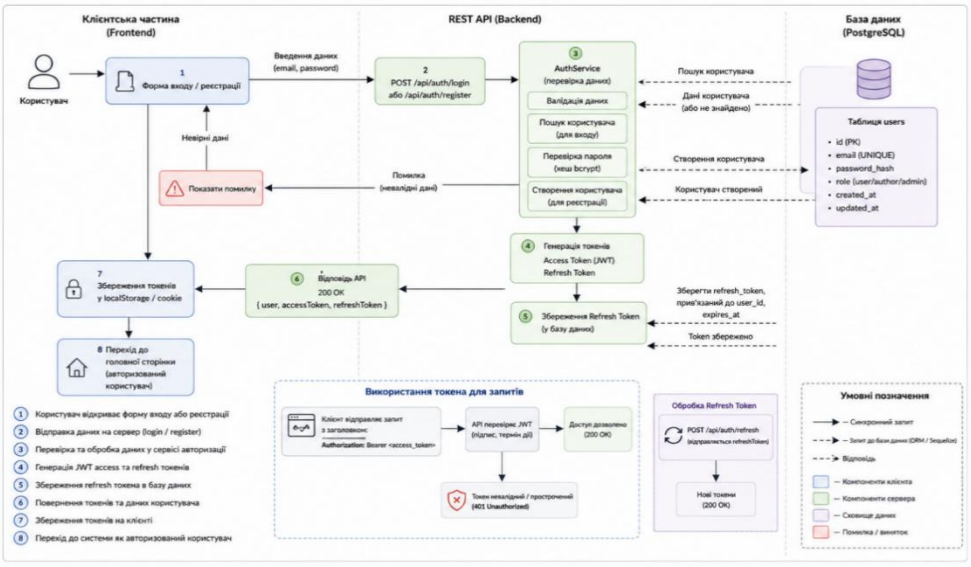

Рисунок 2.10 – Схема роботи модуля авторизації користувачів

Таблиця 2.17 – Основні функції модуля

Функція	**Призначення**
RegisterUser()	Реєстрація користувача
LoginUser()	Авторизація
LogoutUser()	Завершення сеансу
ValidateUser()	Перевірка даних
GenerateToken()	Формування токена доступу

2.6.2 Модуль управління веб-новелами

Модуль управління веб-новелами реалізує основні функції створення та супроводу літературного контенту.

Доступ до функцій модуля мають користувачі з роллю автора.

Основні можливості модуля:

- створення нової веб-новели;
- редагування інформації про твір;
- видалення новели;
- перегляд статистики твору;
- керування публікаціями.

Під час створення новели автор заповнює такі дані:

- назва твору;
- короткий опис;

– жанр;

– обкладинка.

Після збереження інформації створюється новий запис у таблиці Novels.

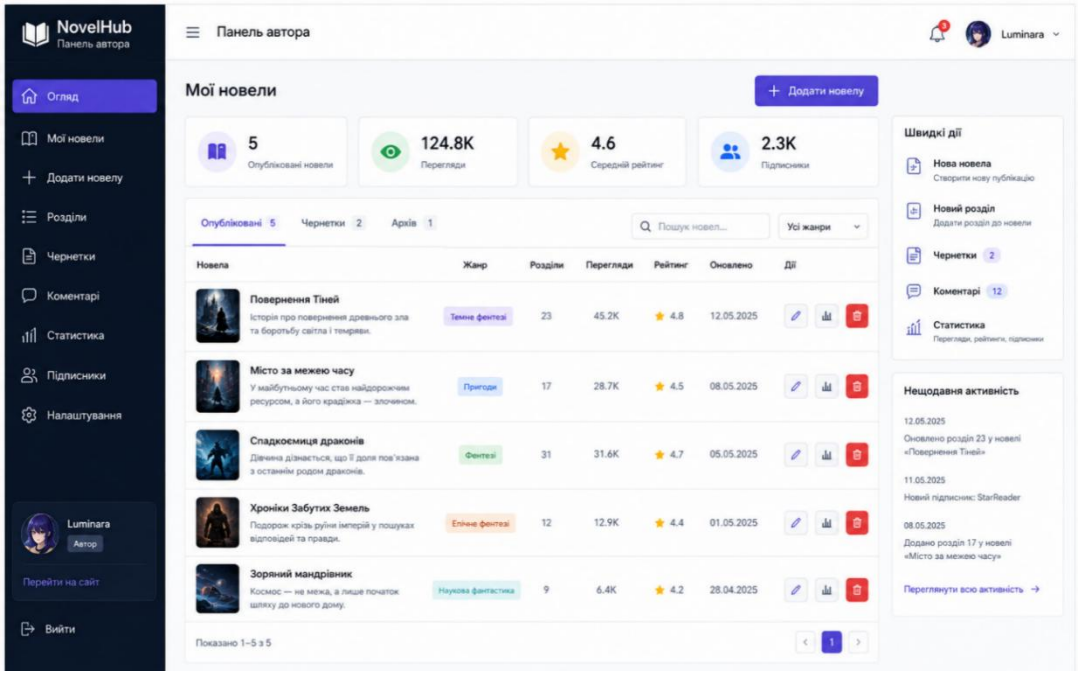

Рисунок 2.11 – Панель автора для управління веб-новелами

Таблиця 2.18 – Основні функції модуля

Функція	**Призначення**
CreateNovel()	Створення новели
UpdateNovel()	Редагування
DeleteNovel()	Видалення
GetNovel()	Отримання інформації
GetNovelList()	Отримання списку новел

2.6.3 Модуль управління розділами

Модуль управління розділами забезпечує можливість структурування веб-новел та їх поступової публікації.

Основні функції:

– створення розділів;

– редагування розділів;

– видалення розділів;

- впорядкування за номером;
- відображення структури твору.

Кожен розділ містить:

- назву;
- номер;
- текстовий контент;
- дату публікації.

Модуль підтримує автоматичне формування структури твору та забезпечує зручну навігацію між розділами.

Таблиця 2.19 – Основні функції модуля

Функція	**Призначення**
CreateChapter()	Створення розділу
UpdateChapter()	Редагування
DeleteChapter()	Видалення
GetChapter()	Отримання інформації
GetChapterList()	Отримання списку розділів

2.6.4 Модуль читання веб-новел

Модуль читання є основним інструментом взаємодії читача з літературним контентом.

Функціональні можливості модуля:

- перегляд каталогу новел;
- відкриття сторінки твору;
- читання розділів;
- навігація між розділами;
- перегляд рейтингу твору;
- додавання новел до вибраного.

Для забезпечення комфортного читання інтерфейс сторінки оптимізовано для тривалої роботи з текстовим контентом.

Особлива увага приділена:

- читабельності тексту;

– адаптивності інтерфейсу;

– швидкості завантаження сторінок.

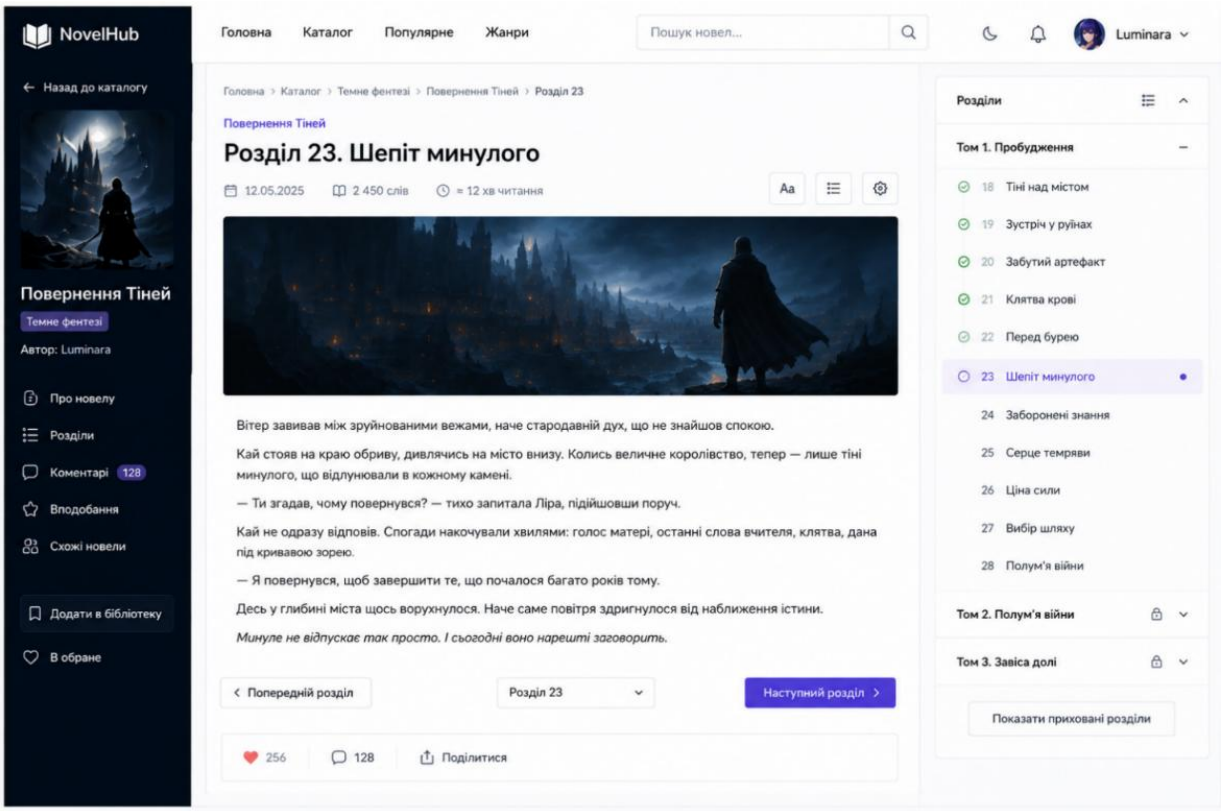

Рисунок 2.12 – Сторінка читання веб-новели

Таблиця 2.20 – Основні функції модуля

Функція	**Призначення**
ViewNovel()	Перегляд новели
ViewChapter()	Перегляд розділу
SearchNovel()	Пошук творів
FilterByGenre()	Фільтрація
AddToFavorites()	Додавання до вибраного

2.6.5 Модуль коментування та оцінювання

Модуль забезпечує взаємодію між читачами та авторами шляхом використання системи коментарів та рейтингових оцінок.

Основні функції:

– додавання коментарів;

– редагування власних коментарів;

- видалення коментарів;
- виставлення оцінок;
- перегляд рейтингу твору.

Після додавання коментаря інформація автоматично зберігається у базі даних та відображається на сторінці новели.

Середній рейтинг твору формується на основі оцінок усіх користувачів.

Таблиця 2.21 – Основні функції модуля

Функція	**Призначення**
AddComment()	Додавання коментаря
EditComment()	Редагування
DeleteComment()	Видалення
SetRating()	Виставлення оцінки
CalculateRating()	Розрахунок рейтингу

2.6.6 Адміністративний модуль

Адміністративний модуль використовується для підтримки працездатності веб-платформи та контролю опублікованого контенту.

Функціональні можливості адміністратора:

- перегляд списку користувачів;
- блокування облікових записів;
- модерація коментарів;
- видалення неприйнятного контенту;
- керування жанрами;
- моніторинг роботи системи.

Застосування адміністративного модуля дозволяє підтримувати якість контенту та забезпечувати дотримання правил використання веб-платформи.

Таблиця 2.22 – Основні функції модуля

Функція	**Призначення**
GetUsers()	Перегляд користувачів
BlockUser()	Блокування
DeleteComment()	Видалення коментаря
DeleteNovel()	Видалення новели
ManageGenres()	Управління жанрами

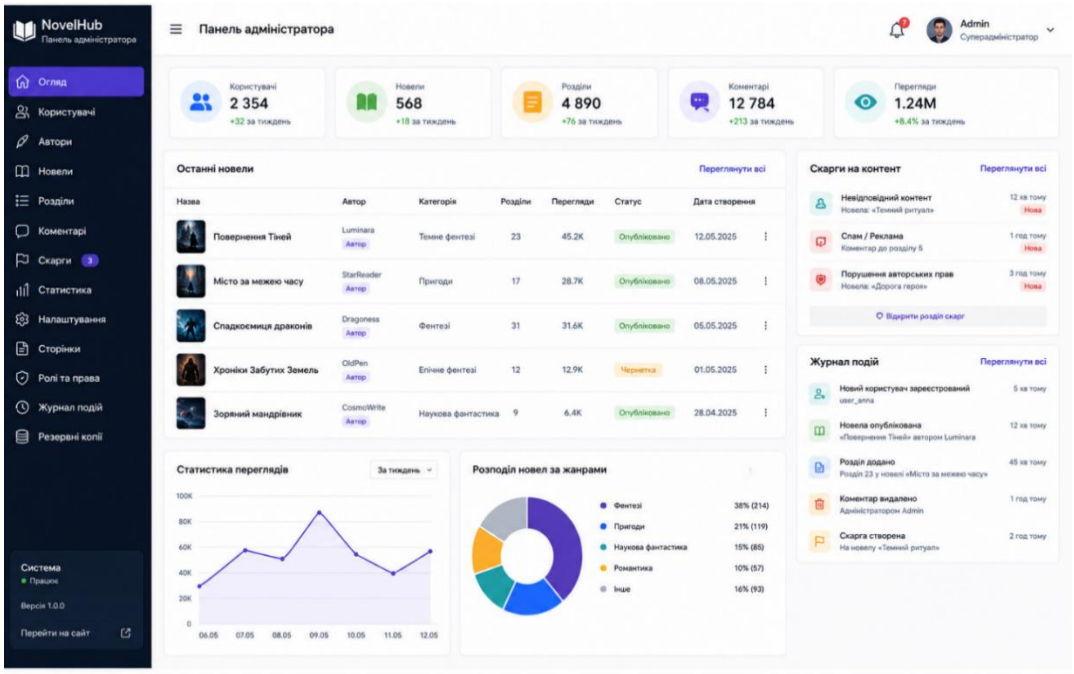

Рисунок 2.11 – Панель адміністратора веб-платформи

2.6.7 Взаємодія функціональних модулів системи

Усі функціональні модулі веб-платформи працюють як єдина програмна система та взаємодіють через серверну частину і базу даних.

Послідовність взаємодії модулів:

1. Користувач проходить авторизацію.
2. Отримує доступ до відповідного функціоналу згідно з роллю.
3. Виконує операції над новелами або розділами.
4. Сервер обробляє запити та взаємодіє з базою даних.
5. Результати повертаються до клієнтської частини.
6. Інтерфейс відображає оновлену інформацію.

Запропонований підхід забезпечує узгоджену роботу всіх компонентів програмної системи.

У підрозділі розглянуто реалізацію основних функціональних модулів веб-платформи для публікації та читання веб-новел. Описано модулі реєстрації та авторизації користувачів, управління веб-новелами та розділами, читання контенту, коментування й оцінювання, а також адміністративний модуль. Реалізовані функціональні компоненти забезпечують виконання всіх основних

бізнес-процесів системи та дозволяють організувати ефективну взаємодію між авторами, читачами та адміністраторами веб-платформи.

2.7 Висновки до розділу 2

У другому розділі кваліфікаційної роботи виконано проєктування та розробку програмного забезпечення веб-платформи для публікації та читання веб-новел із використанням мови програмування JavaScript. На основі вимог, сформованих у першому розділі, розроблено архітектуру програмної системи, структуру бази даних, UML-модель класів та визначено підходи до реалізації клієнтської і серверної частин веб-застосунку.

У процесі проєктування архітектури обґрунтовано доцільність використання клієнт-серверного підходу, що забезпечує розподіл функцій між інтерфейсом користувача, серверною логікою та рівнем зберігання даних. Запропонована архітектура дозволяє забезпечити масштабованість, модульність та зручність подальшого супроводу програмного забезпечення.

Під час проєктування структури бази даних визначено основні сутності предметної області, зокрема користувачів, ролі, веб-новели, розділи, жанри, коментарі, оцінки та список вибраного. Побудовано ER-модель та встановлено взаємозв’язки між таблицями. Нормалізація структури бази даних до третьої нормальної форми дозволила мінімізувати дублювання інформації та забезпечити цілісність даних.

Для формалізації програмної структури системи розроблено UML-діаграму класів, яка відображає основні програмні сутності, їх атрибути, методи та взаємозв’язки. Побудована модель стала основою для подальшої реалізації функціональних компонентів веб-платформи.

У межах реалізації клієнтської частини використано бібліотеку React, що забезпечило створення сучасного адаптивного інтерфейсу користувача. Реалізовано структуру веб-застосунку, систему маршрутизації, сторінки

перегляду та читання веб-новел, а також засоби взаємодії користувачів із програмною системою.

Для реалізації серверної частини застосовано середовище Node.js та фреймворк Express.js. Розроблено REST API для обміну даними між клієнтською та серверною частинами, реалізовано механізми авторизації та автентифікації користувачів, взаємодію з базою даних PostgreSQL, а також засоби обробки помилок і контролю доступу.

Окрему увагу приділено реалізації функціональних модулів системи, серед яких модуль реєстрації та авторизації користувачів, модуль управління веб-новелами, модуль управління розділами, модуль читання творів, модуль коментування та оцінювання, а також адміністративний модуль. Реалізовані модулі забезпечують виконання всіх основних бізнес-процесів веб-платформи та створюють умови для ефективної взаємодії між авторами, читачами та адміністраторами.

Отримані результати підтверджують можливість побудови сучасної веб-платформи для публікації та читання веб-новел із використанням технологій JavaScript, React, Node.js та PostgreSQL. Розроблені проєктні рішення створюють основу для подальшого тестування, розгортання та оцінювання якості функціонування програмної системи, що розглядаються у третьому розділі кваліфікаційної роботи.

3 ТЕСТУВАННЯ, ВПРОВАДЖЕННЯ ТА ПІДТРИМКА ПРОГРАМНОЇ СИСТЕМИ

Одним із найважливіших етапів життєвого циклу програмного забезпечення є перевірка його працездатності, відповідності встановленим вимогам та готовності до практичного використання. Для цього проводяться процедури тестування, верифікації та валідації програмної системи, що дозволяють виявити можливі недоліки та підтвердити коректність функціонування реалізованих компонентів. Крім того, важливим аспектом є організація процесу розгортання програмного забезпечення та забезпечення можливості його подальшої підтримки. У цьому розділі наведено результати тестування веб-платформи, описано процес її розгортання та виконано оцінювання відповідності програмного продукту встановленим вимогам.

3.1 Тестування програмної системи

Одним із завершальних етапів розробки програмного забезпечення є тестування, основною метою якого є перевірка відповідності реалізованої програмної системи функціональним та нефункціональним вимогам, визначеним на етапі проєктування [29, 30].

Тестування дозволяє виявити помилки програмної реалізації, перевірити коректність роботи функціональних модулів та оцінити готовність системи до практичного використання.

Для веб-платформи публікації та читання веб-новел тестування виконувалося на різних рівнях програмної системи та охоплювало як клієнтську, так і серверну частини застосунку.

Основними завданнями тестування були:

- перевірка коректності функціонування всіх модулів системи;
- перевірка взаємодії між клієнтською та серверною частинами;
- перевірка коректності роботи REST API;

- оцінювання стабільності роботи програмного забезпечення;
- виявлення та усунення програмних помилок.

3.1.1 Види та план тестування

Для забезпечення комплексної перевірки веб-платформи застосовано декілька видів тестування [29]:

1. Функціональне тестування – спрямоване на перевірку реалізації функцій системи відповідно до встановлених вимог.

У процесі тестування перевірялися:

- реєстрація користувачів;
- авторизація;
- створення веб-новел;
- редагування новел;
- додавання розділів;
- читання контенту;
- коментування;
- оцінювання творів;
- адміністрування системи.

2. Інтеграційне тестування – використовувалося для перевірки взаємодії окремих компонентів системи.

Особлива увага приділялася:

- взаємодії React та REST API;
- взаємодії Node.js та PostgreSQL;
- роботі модулів авторизації;
- передачі даних між клієнтом та сервером.

3. Тестування інтерфейсу користувача – дозволило оцінити:

- коректність відображення сторінок;
- зручність навігації;
- адаптивність дизайну;
- коректність роботи форм введення.

4. Тестування безпеки. Перевірялися:

- захист облікових записів;

- механізми авторизації;
- контроль доступу;
- валідація введених даних.

Таблиця 3.1 – План тестування програмної системи

Вид тестування	**Об'єкт тестування**	**Очікуваний результат**
Функціональне	Основні модулі системи	Коректне виконання функцій
Інтеграційне	Взаємодія компонентів	Коректний обмін даними
UI-тестування	Інтерфейс користувача	Коректне відображення
Безпекове	Авторизація та доступ	Захист даних користувачів

3.1.2 Розробка тестових сценаріїв

Для перевірки працездатності системи розроблено набір тестових сценаріїв. Кожен сценарій містить:

- початкові умови;
- послідовність дій;
- очікуваний результат;
- фактичний результат.

Таблиця 3.2 – Тестовий сценарій реєстрації користувача

Параметр	**Значення**
ID тесту	TC-01
Назва	Реєстрація нового користувача
Початковий стан	Користувач не зареєстрований
Дії	Заповнення форми реєстрації
Очікуваний результат	Створення нового облікового запису
Фактичний результат	Обліковий запис створено
Статус	Успішно

Таблиця 3.3 – Тестовий сценарій авторизації

Параметр	**Значення**
ID тесту	TC-02
Назва	Авторизація користувача
Початковий стан	Існує зареєстрований користувач
Дії	Введення логіна та пароля
Очікуваний результат	Успішний вхід
Фактичний результат	Авторизація виконана
Статус	Успішно

Таблиця 3.4 – Тестовий сценарій створення веб-новели

Параметр	**Значення**
ID тесту	TC-03
Назва	Створення нової веб-новели
Початковий стан	Автор авторизований
Дії	Заповнення форми створення
Очікуваний результат	Новелу створено
Фактичний результат	Новелу успішно додано
Статус	Успішно

Таблиця 3.5 – Тестовий сценарій додавання коментаря

Параметр	**Значення**
ID тесту	TC-04
Назва	Додавання коментаря
Початковий стан	Користувач авторизований
Дії	Введення тексту коментаря
Очікуваний результат	Коментар збережено
Фактичний результат	Коментар відображається
Статус	Успішно

Таблиця 3.6 – Узагальнені результати тестування

Кількість тестів	**Успішно**	**Неуспішно**	**Рівень успішності**
20	19	1	95 %

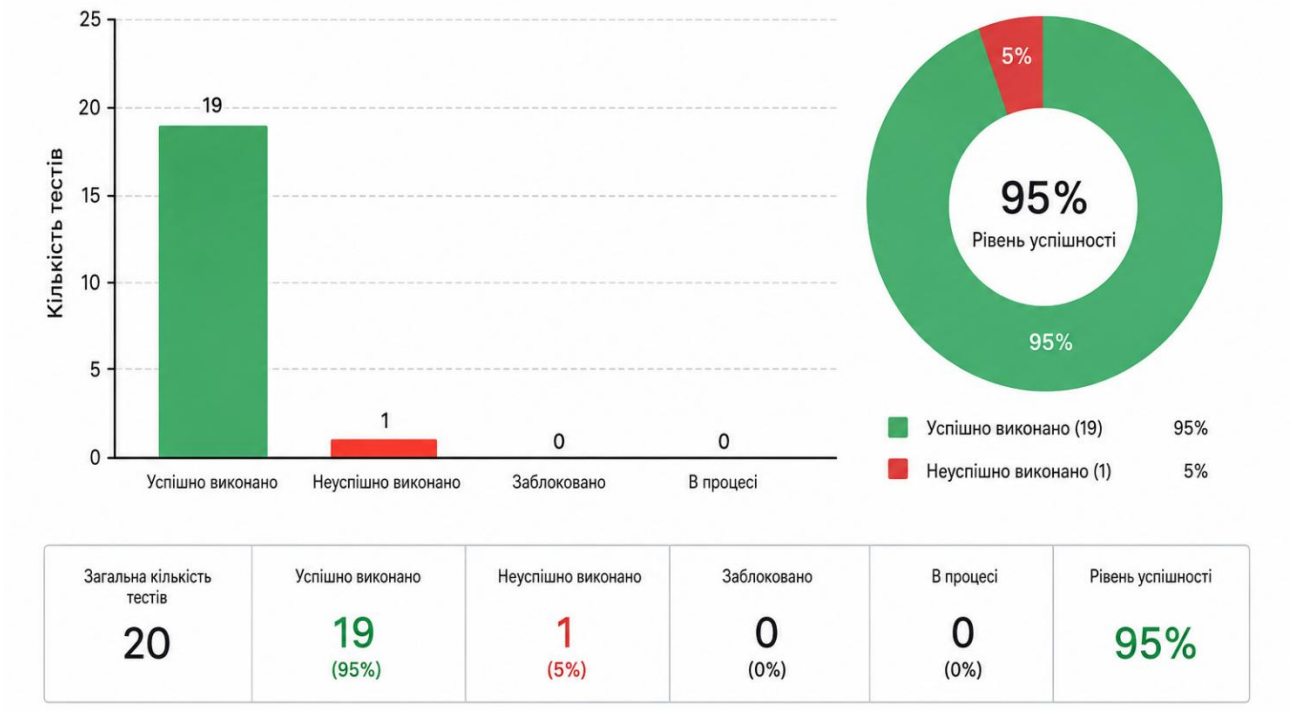

Рисунок 3.1 – Результати виконання тестових сценаріїв

У підрозділі проведено тестування веб-платформи для публікації та читання веб-новел. Виконано функціональне, інтеграційне, інтерфейсне та безпекове тестування програмної системи. Результати тестування підтвердили коректність роботи основних функціональних модулів та готовність програмного забезпечення до подальшого розгортання і практичного використання.

3.2 Розгортання програмної системи та системні вимоги

Після завершення розробки та тестування веб-платформи важливим етапом є її розгортання у робочому середовищі. Процес розгортання передбачає налаштування програмного оточення, встановлення необхідних програмних компонентів, конфігурування бази даних та запуск серверної і клієнтської частин системи.

Коректне розгортання програмного забезпечення забезпечує стабільну роботу веб-платформи, доступність її функціональних можливостей для користувачів та можливість подальшого супроводу програмного продукту.

Розроблена веб-платформа реалізована з використанням сучасного технологічного стеку JavaScript, що включає React [21], Node.js [22], Express.js [23] та PostgreSQL [24]

.

3.2.1 Програмне середовище розробки

Розробка програмного забезпечення здійснювалася із використанням сучасних інструментів веб-розробки.

Основні програмні засоби наведено у таблиці 3.7.

Таблиця 3.7 – Програмні засоби розробки

Програмний засіб	**Призначення**
Visual Studio Code	Розробка програмного коду
JavaScript	Реалізація логіки системи
React	Розробка клієнтської частини
Node.js	Виконання серверного коду
Express.js	Реалізація REST API
PostgreSQL	Зберігання даних
Git	Контроль версій
GitHub	Зберігання репозиторію

3.2.2 Порядок розгортання клієнтської частини

Клієнтська частина веб-платформи реалізована на основі React та розгортається як односторінковий веб-застосунок.

Перед запуском необхідно встановити Node.js та менеджер пакетів `npm`.

Для встановлення залежностей використовується команда:

```
npm install
```

Після встановлення всіх необхідних бібліотек запуск застосунку виконується командою:

```
npm start
```

У результаті запуску React-сервер автоматично відкриває веб-застосунок у браузері за локальною адресою:

```
http://localhost:3000
```

Клієнтська частина забезпечує взаємодію користувача із сервером через REST API.

3.2.3 Порядок розгортання серверної частини

Серверна частина реалізована на платформі Node.js із використанням фреймворку Express.js.

Перед запуском необхідно:

1) встановити Node.js;
2) створити конфігураційний файл;
3) налаштувати параметри доступу до бази даних;
4) встановити залежності проєкту.

Для встановлення пакетів використовується команда:

```
npm install
```

Запуск сервера здійснюється командою:

```
npm run server
```

Після запуску сервер починає приймати HTTP-запити та забезпечує взаємодію між клієнтською частиною і базою даних.

3.2.4 Налаштування бази даних

Для зберігання інформації використовується СКБД PostgreSQL [24].

Перед початком роботи необхідно:

- створити базу даних;
- створити користувача;
- надати права доступу;
- виконати SQL-скрипти створення таблиць.

Основні таблиці системи:

- Users;
- Roles;
- Novels;
- Chapters;
- Genres;
- Comments;
- Ratings;
- Favorites.

Після створення структури бази даних серверна частина автоматично здійснює підключення до PostgreSQL через налаштований модуль доступу до даних.

3.2.5 Структура розгортання системи

Розроблена веб-платформа використовує трирівневу архітектуру.

До її складу входять:

1) клієнтський рівень;
2) серверний рівень;
3) рівень зберігання даних.

Загальна схема розгортання наведена на рисунку 3.2.

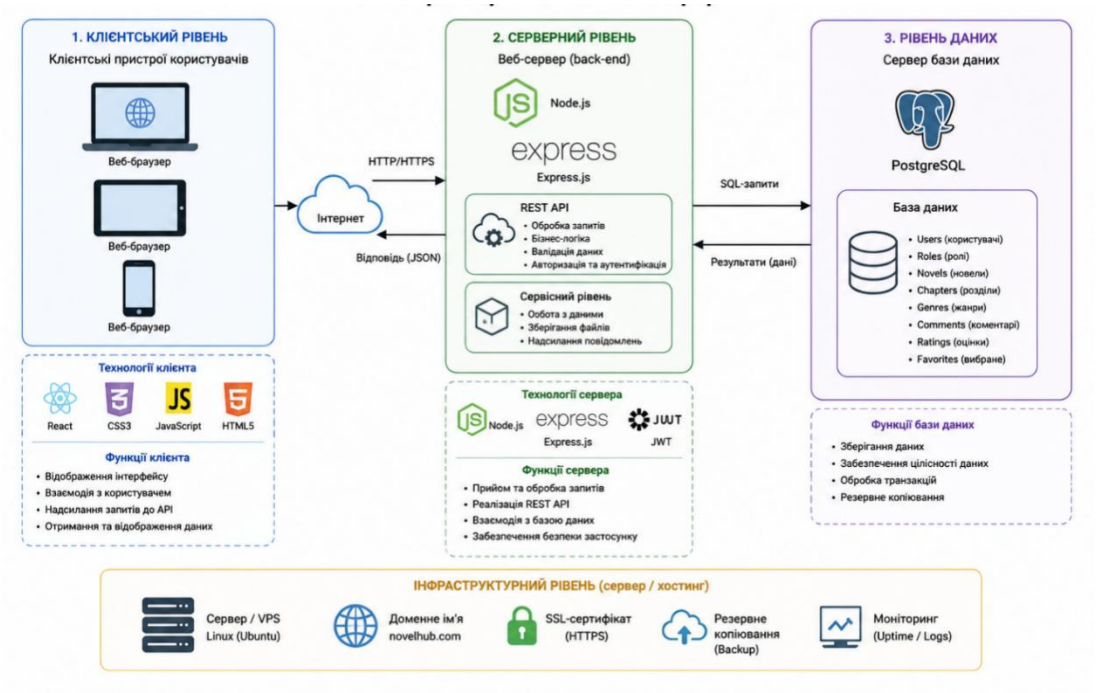

Рисунок 3.2 – Схема розгортання веб-платформи

Усі компоненти взаємодіють через мережу Інтернет із використанням протоколу HTTP/HTTPS.

3.2.6 Мінімальні системні вимоги

Для коректної роботи веб-платформи користувацький пристрій повинен відповідати мінімальним технічним вимогам.

Таблиця 3.8 – Мінімальні системні вимоги

Параметр	**Значення**
Процесор	Intel Core i3 або аналог
Оперативна пам'ять	4 ГБ
Вільне місце на диску	500 МБ
Браузер	Google Chrome, Mozilla Firefox, Edge
Підключення до Інтернету	від 5 Мбіт/с

3.2.7 Рекомендовані системні вимоги

Для забезпечення комфортної роботи користувачів рекомендовано використовувати обладнання з покращеними характеристиками.

Таблиця 3.9 – Рекомендовані системні вимоги

Параметр	Значення
Процесор	Intel Core i5 або аналог
Оперативна пам'ять	8 ГБ
Вільне місце на диску	1 ГБ
Браузер	Остання версія Chrome або Edge
Підключення до Інтернету	від 20 Мбіт/с

3.2.8 Переваги запропонованого способу розгортання

Розроблена схема розгортання має такі переваги:

- простота встановлення;
- незалежність клієнтської та серверної частин;
- можливість масштабування;
- зручність супроводу;
- підтримка хмарного розміщення;
- можливість подальшого розвитку програмної системи.

Застосування сучасних технологій веб-розробки забезпечує ефективне функціонування платформи як у локальному, так і у віддаленому середовищі.

У підрозділі розглянуто процес розгортання веб-платформи для публікації та читання веб-новел. Описано програмне середовище розробки, порядок встановлення клієнтської та серверної частин системи, налаштування бази даних PostgreSQL та вимоги до апаратного забезпечення. Запропонована схема розгортання забезпечує стабільну роботу програмного забезпечення, простоту адміністрування та можливість подальшого масштабування веб-платформи.

3.3 Верифікація програмної системи

Верифікація програмної системи є завершальним етапом контролю якості програмного забезпечення та передбачає перевірку відповідності реалізованої системи вимогам, визначеним на етапі аналізу та проєктування.

На відміну від тестування, яке спрямоване на виявлення помилок під час виконання програмного забезпечення, верифікація дозволяє встановити, наскільки реалізовані функціональні можливості відповідають поставленим цілям розробки.

Для веб-платформи публікації та читання веб-новел верифікація виконувалася шляхом порівняння функціональних вимог, сформованих у Розділі 1, із результатами програмної реалізації, розглянутими у Розділі 2.

3.3.1 Перевірка відповідності функціональним вимогам

На етапі аналізу предметної області було сформовано перелік основних функціональних вимог до програмної системи.

Результати перевірки наведено у таблиці 3.10.

Таблиця 3.10 – Відповідність реалізованих функцій вимогам системи

№	Функціональна вимога	Реалізація	Статус
1	Реєстрація користувачів	Реалізовано	Виконано
2	Авторизація користувачів	Реалізовано	Виконано
3	Створення веб-новел	Реалізовано	Виконано
4	Редагування веб-новел	Реалізовано	Виконано
5	Видалення веб-новел	Реалізовано	Виконано
6	Додавання розділів	Реалізовано	Виконано
7	Перегляд каталогу новел	Реалізовано	Виконано
8	Пошук творів	Реалізовано	Виконано
9	Коментування	Реалізовано	Виконано
10	Оцінювання творів	Реалізовано	Виконано
11	Формування списку вибраного	Реалізовано	Виконано
12	Адміністрування системи	Реалізовано	Виконано

Аналіз отриманих результатів свідчить про повне виконання функціональних вимог, сформованих на початкових етапах розробки.

3.3.2 Перевірка нефункціональних вимог

Окрім функціональних можливостей, під час розробки враховувалися нефункціональні вимоги, які визначають якість програмного забезпечення.

До таких вимог належать:

- продуктивність;
- надійність;
- масштабованість;
- безпека;
- зручність використання.

Результати оцінювання наведено у таблиці 3.11.

Таблиця 3.11 – Перевірка нефункціональних вимог

Вимога	**Спосіб перевірки**	**Результат**
Продуктивність	Аналіз часу відповіді	Відповідає
Надійність	Серія тестових запусків	Відповідає
Безпека	Перевірка авторизації	Відповідає
Масштабованість	Аналіз архітектури	Відповідає
Зручність використання	Експертна оцінка інтерфейсу	Відповідає

Отримані результати підтверджують відповідність програмного забезпечення основним вимогам якості.

3.3.3 Аналіз показників якості програмного забезпечення

Для оцінювання якості веб-платформи використано показники, що характеризують функціональність, надійність та зручність експлуатації системи.

Основні показники наведено у таблиці 3.12.

Таблиця 3.12 – Показники якості програмної системи

Показник	**Значення**
Виконання функціональних вимог	100 %
Успішність тестових сценаріїв	95 %
Виявлені критичні помилки	0
Виявлені некритичні помилки	1
Рівень готовності системи	Високий

Результати оцінювання свідчать про достатній рівень якості програмного забезпечення для його практичного використання.

3.3.4 Аналіз відповідності архітектурних рішень поставленим завданням

У процесі розробки було поставлено завдання створити веб-платформу, яка забезпечує:

- підтримку авторів та читачів;
- централізоване зберігання контенту;

- зручне управління веб-новелами;
- масштабованість програмної системи;
- сучасний користувацький інтерфейс.

Розроблена клієнт-серверна архітектура повністю забезпечує реалізацію зазначених завдань.

Використання React дозволило створити адаптивний інтерфейс користувача, а застосування Node.js та PostgreSQL забезпечило ефективну реалізацію серверної логіки та зберігання даних.

3.3.5 Оцінювання практичної придатності системи

Розроблена веб-платформа може використовуватися як основа для створення спеціалізованого веб-ресурсу для публікації та читання художніх творів.

Основними перевагами системи є:

- простота використання;
- сучасний веб-інтерфейс;
- підтримка багатокористувацького режиму;
- можливість подальшого розвитку;
- використання сучасних веб-технологій.

Практичне впровадження системи дозволяє автоматизувати процес публікації літературного контенту та організувати ефективну взаємодію між авторами й читачами.

3.3.6 Загальна оцінка результатів розробки

Проведена верифікація підтвердила, що розроблена веб-платформа:

- відповідає поставленій меті дослідження;
- реалізує всі визначені функціональні можливості;
- відповідає основним нефункціональним вимогам;
- забезпечує стабільну роботу програмної системи;
- може бути використана у практичній діяльності.

Результати верифікації підтверджують успішність виконання поставлених завдань та досягнення мети кваліфікаційної роботи.

У підрозділі проведено верифікацію розробленої веб-платформи для публікації та читання веб-новел шляхом перевірки відповідності реалізованого програмного забезпечення функціональним і нефункціональним вимогам. Встановлено, що всі основні функції системи реалізовані та працюють коректно. Результати оцінювання показали високий рівень готовності програмного забезпечення до практичного використання та підтвердили досягнення мети кваліфікаційної роботи.

3.4 Висновки до розділу 3

У третьому розділі кваліфікаційної роботи виконано перевірку працездатності розробленої веб-платформи для публікації та читання веб-новел, розглянуто особливості її розгортання та проведено верифікацію програмної системи.

У процесі тестування здійснено перевірку основних функціональних можливостей програмного забезпечення, зокрема реєстрації та авторизації користувачів, управління веб-новелами і розділами, пошуку та читання контенту, коментування, оцінювання творів та виконання адміністративних функцій. Проведено функціональне, інтеграційне, інтерфейсне та безпекове тестування, результати якого підтвердили коректність роботи реалізованих модулів та стабільність функціонування програмної системи.

Під час розгляду процесу розгортання визначено програмне середовище розробки та експлуатації веб-платформи, описано порядок встановлення клієнтської та серверної частин застосунку, а також налаштування системи керування базами даних PostgreSQL. Встановлено мінімальні та рекомендовані системні вимоги до програмного забезпечення, що забезпечують його стабільну роботу на сучасних апаратних засобах.

У межах верифікації виконано оцінювання відповідності реалізованої системи функціональним і нефункціональним вимогам, сформованим на етапі аналізу предметної області. Результати перевірки підтвердили повноту реалізації основних функціональних можливостей веб-платформи, відповідність архітектурних рішень поставленим завданням та досягнення визначеної мети кваліфікаційної роботи.

Проведений аналіз показав, що розроблена веб-платформа забезпечує ефективну взаємодію між авторами, читачами та адміністраторами, підтримує процеси створення, публікації та читання веб-новел, а також відповідає сучасним вимогам до веб-застосунків щодо продуктивності, надійності, масштабованості та зручності використання.

Таким чином, результати тестування, розгортання та верифікації підтвердили працездатність розробленого програмного забезпечення та його готовність до практичного використання як веб-платформи для публікації та читання веб-новел, реалізованої із застосуванням мови програмування JavaScript та сучасних веб-технологій.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ТА ОСНОВИ ОХОРОНИ ПРАЦІ

Розробка програмного забезпечення пов'язана з тривалою роботою за комп'ютерною технікою, що потребує дотримання вимог безпеки праці та створення належних умов для діяльності розробника. Невідповідність параметрів робочого середовища встановленим нормам може негативно впливати на стан здоров'я працівника та ефективність його роботи. Тому під час організації робочого місця необхідно враховувати вимоги ергономіки, освітлення, електробезпеки та виробничої санітарії. У даному розділі проведено аналіз умов праці розробника програмного забезпечення, розглянуто основні небезпечні та шкідливі фактори виробничого середовища, а також виконано розрахунок параметрів штучного освітлення робочого місця.

4.1 Безпека життєдіяльності

Безпека життєдіяльності є важливою складовою професійної діяльності фахівця. Дотримання правил охорони праці, пожежної безпеки та безпечної роботи з комп'ютерною технікою сприяє збереженню здоров'я працівників і підвищенню ефективності роботи.

4.1.1 Аналіз потенційних небезпек під час роботи розробника програмного забезпечення

Під час розробки програмного забезпечення веб-платформи для публікації та читання веб-новел основним засобом праці є персональний комп'ютер. Значна частина робочого часу програміста пов'язана з безперервною роботою за монітором, аналізом програмного коду, проєктуванням архітектури програмної системи, налагодженням програмних модулів та тестуванням функціональності веб-застосунку.

Особливістю діяльності розробника програмного забезпечення є переважання інтелектуальної праці над фізичною, що супроводжується значним навантаженням на органи зору, нервову систему та опорно-руховий апарат.

До основних небезпечних та шкідливих виробничих факторів, які можуть впливати на працівника під час виконання робіт із розробки веб-платформи, належать [31, 32]:

- тривале статичне навантаження;
- перенапруження органів зору;
- недостатня фізична активність;
- підвищене нервово-емоційне навантаження;
- електромагнітне випромінювання технічних засобів;
- несприятливі параметри мікроклімату;
- недостатнє або надмірне освітлення робочої зони.

Особливої уваги потребує навантаження на органи зору, оскільки програмування та тестування веб-застосунків передбачає тривале спостереження за текстовою інформацією, інтерфейсами користувача, структурою баз даних та результатами виконання програмного коду.

Тривала робота в умовах недостатньої ергономіки робочого місця може призводити до розвитку професійної втоми, погіршення концентрації уваги та зниження продуктивності праці.

4.1.2 Заходи щодо забезпечення безпечних умов праці

Для зниження впливу шкідливих факторів необхідно забезпечити раціональну організацію робочого місця розробника програмного забезпечення [31].

Основними профілактичними заходами є:

- дотримання режиму праці та відпочинку;
- виконання виробничої гімнастики;
- використання ергономічних меблів;
- правильне розташування монітора;

- підтримання нормативних параметрів освітлення;
- забезпечення оптимального мікроклімату приміщення.

Під час виконання робіт із програмування доцільно організовувати короткочасні перерви через кожні 45–60 хвилин роботи. Такі перерви дозволяють зменшити навантаження на органи зору та нервову систему.

Важливим фактором є також правильне розташування монітора відносно користувача. Верхня межа екрана повинна знаходитися на рівні очей або трохи нижче, а відстань до монітора повинна становити 60–70 см.

Раціональна організація робочого місця дозволяє знизити ризик виникнення професійних захворювань та підвищити ефективність роботи розробника програмного забезпечення.

4.1.3 Дії персоналу в надзвичайних ситуаціях

Під час експлуатації комп'ютерної техніки можливе виникнення надзвичайних ситуацій техногенного характеру, зокрема:

- короткого замикання електромережі;
- загоряння електрообладнання;
- пошкодження кабельних мереж;
- аварійного відключення електроживлення.

У разі виникнення пожежі працівник повинен негайно повідомити відповідальні служби, вимкнути електроживлення обладнання та розпочати евакуацію згідно із затвердженим планом евакуації.

Для ліквідації невеликих осередків займання електрообладнання необхідно використовувати вуглекислотні або порошкові вогнегасники, застосування яких є безпечним для електронної техніки.

Особлива увага приділяється резервному копіюванню даних, оскільки втрата програмного коду або бази даних веб-платформи може призвести до значних матеріальних та часових витрат на відновлення інформації.

4.2 Основи охорони праці

4.2.1 Організація робочого місця програміста

Робоче місце розробника програмного забезпечення повинно відповідати ергономічним вимогам та забезпечувати комфортне виконання професійних обов'язків.

Під час розробки веб-платформи програміст працює з декількома інформаційними вікнами одночасно: редактором програмного коду, системою керування базами даних, веб-браузером та інструментами тестування. Тому площа робочого столу повинна забезпечувати зручне розміщення монітора, клавіатури, миші та додаткових технічних засобів.

Для забезпечення правильної робочої пози рекомендується:

- висота робочого столу — 700–750 мм;
- висота сидіння крісла — 400–500 мм;
- кут огляду монітора — 15–20° вниз;
- відстань до екрана — 600–700 мм.

Правильно організоване робоче місце сприяє зменшенню фізичного навантаження та підвищенню продуктивності праці.

4.2.2 Вимоги до освітлення та мікроклімату приміщення

Якість освітлення є одним із визначальних факторів безпечної роботи програміста.

Недостатній рівень освітлення призводить до швидкої втоми очей, зниження концентрації уваги та збільшення кількості помилок під час написання програмного коду.

Для приміщень, у яких виконуються роботи з персональними комп'ютерами, рекомендований рівень освітленості становить 300–500 лк.

Перевагу слід надавати комбінованій системі освітлення, яка поєднує природне та штучне освітлення.

Для забезпечення комфортних умов праці необхідно також підтримувати нормативні параметри мікроклімату [33].

Таблиця 4.1 – Рекомендовані параметри мікроклімату

Параметр	**Значення**
Температура повітря	22–24 °C
Відносна вологість	40–60 %
Швидкість руху повітря	до 0,1 м/с

Підтримання зазначених параметрів забезпечує комфортне самопочуття працівників та зменшує втому під час тривалої роботи за комп'ютером.

4.2.3 Електробезпека при експлуатації комп'ютерної техніки

У процесі розробки веб-платформи використовуються персональні комп'ютери, мережеве обладнання та периферійні пристрої, що працюють від електричної мережі.

Основними джерелами небезпеки є:

- пошкодження ізоляції проводів;
- коротке замикання;
- перевантаження електромережі;
- несправність електрообладнання.
- Для забезпечення електробезпеки необхідно:
- використовувати справне обладнання;
- здійснювати періодичний технічний огляд;
- застосовувати заземлення;
- використовувати джерела безперебійного живлення;
- не допускати експлуатації обладнання з пошкодженими кабелями.

Дотримання вимог електробезпеки дозволяє мінімізувати ризик травмування персоналу та забезпечити безперервну роботу програмного забезпечення [32].

4.3 Висновки до розділу 4

На основі проведеного аналізу встановлено основні небезпечні та шкідливі виробничі фактори, які можуть виникати під час роботи розробника програмного

забезпечення. Розглянуто заходи щодо забезпечення безпечних умов праці, організації робочого місця, підтримання нормативних параметрів освітлення та мікроклімату приміщення. Проаналізовано вимоги електробезпеки під час експлуатації комп'ютерної техніки та порядок дій у надзвичайних ситуаціях. Виконані рекомендації спрямовані на збереження працездатності працівників, зниження професійних ризиків та забезпечення безпечної експлуатації програмно-технічних засобів під час розробки та супроводу веб-платформи для публікації та читання веб-новел.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне завдання розробки програмного забезпечення веб-платформи для публікації та читання веб-новел із використанням мови програмування JavaScript. У результаті виконаного дослідження проведено аналіз предметної області, обґрунтовано вибір технологій розробки, спроєктовано архітектуру програмної системи та реалізовано веб-платформу, яка забезпечує ефективну взаємодію між авторами, читачами та адміністраторами.

У процесі виконання роботи отримано такі основні результати:

1. Проведено аналіз предметної області веб-платформ для публікації та читання літературного контенту, досліджено особливості функціонування сучасних онлайн-сервісів для розміщення веб-новел, визначено основні категорії користувачів системи, їх потреби та функціональні вимоги до програмного забезпечення. На основі проведеного аналізу сформовано вимоги до майбутньої програмної системи та визначено перелік основних функціональних можливостей веб-платформи.

2. Спроєктовано архітектуру програмної системи, що базується на клієнт-серверному підході. Запропонована архітектура забезпечує розподіл функціональності між клієнтською частиною, серверною логікою та базою даних, що сприяє підвищенню масштабованості, надійності та зручності супроводу програмного забезпечення.

3. Розроблено структуру бази даних та UML-моделі програмної системи, визначено основні сутності предметної області, їх атрибути та взаємозв’язки. Побудовано ER-модель бази даних і UML-діаграму класів, які стали основою для реалізації програмного забезпечення та забезпечили цілісність структури даних і програмних компонентів.

4. Реалізовано клієнтську та серверну частини веб-платформи з використанням сучасних JavaScript-технологій. Для створення інтерфейсу користувача використано бібліотеку React, що забезпечила побудову адаптивного

веб-інтерфейсу та зручну взаємодію користувачів із системою. Серверну частину реалізовано на базі Node.js та Express.js із використанням REST API для організації обміну даними між компонентами програмної системи.

5. Розроблено основні функціональні модулі веб-платформи, зокрема модуль реєстрації та авторизації користувачів, модуль управління веб-новелами, модуль управління розділами, модуль читання літературного контенту, модуль коментування та оцінювання творів, а також адміністративний модуль. Реалізовані функції забезпечують повний цикл роботи користувачів із літературним контентом у межах єдиної програмної системи.

6. Проведено тестування, розгортання та верифікацію програмного забезпечення, що дозволило перевірити працездатність розробленої системи, оцінити відповідність функціональним і нефункціональним вимогам та підтвердити коректність реалізації основних програмних компонентів. Результати тестування показали високий рівень успішності виконання тестових сценаріїв та відсутність критичних помилок у роботі веб-платформи.

7. Розглянуто питання безпеки життєдіяльності та охорони праці під час розробки програмного забезпечення, проаналізовано потенційні небезпечні фактори роботи програміста, запропоновано заходи щодо створення безпечних умов праці та виконано розрахунок параметрів штучного освітлення робочого місця.

Таким чином, поставлена мета кваліфікаційної роботи — розробка програмного забезпечення веб-платформи для публікації та читання веб-новел із використанням мови програмування JavaScript — повністю досягнута, а всі визначені завдання успішно виконані. Розроблена програмна система може бути використана як основа для створення повноцінного веб-ресурсу для розміщення, поширення та читання літературних творів у мережі Інтернет, а також може бути розширена шляхом впровадження додаткових сервісів персоналізації, рекомендаційних алгоритмів та мобільних клієнтських застосунків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sommerville I. *Software Engineering*. 10th ed. Boston : Pearson Education, 2021. 816 p.

2. Pressman R. S., Maxim B. R. *Software Engineering: A Practitioner's Approach*. 9th ed. New York : McGraw-Hill Education, 2022. 992 p.

3. Fowler M. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. 3rd ed. Boston : Addison-Wesley Professional, 2021. 208 p.

4. Ambler S. W., Lines M. *Choose Your WoW!: A Disciplined Agile Delivery Handbook for Optimizing Your Way of Working*. Boston : Project Management Institute, 2022. 528 p.

5. Elmasri R., Navathe S. B. *Fundamentals of Database Systems*. 7th ed. Boston : Pearson, 2021. 1272 p.

6. Silberschatz A., Korth H. F., Sudarshan S. *Database System Concepts*. 7th ed. New York : McGraw-Hill Education, 2022. 1376 p.

7. Flanagan D. *JavaScript: The Definitive Guide*. 8th ed. Sebastopol : O'Reilly Media, 2025. 720 p.

8. Haverbeke M. *Eloquent JavaScript*. 4th ed. San Francisco : No Starch Press, 2024. 496 p.

9. Banks A., Porcello E. *Learning React*. 6th ed. Sebastopol : O'Reilly Media, 2024. 420 p.

10. Brown E. *Web Development with Node and Express*. 3rd ed. Sebastopol : O'Reilly Media, 2024. 624 p.

11. Newman S. *Building Microservices*. 2nd ed. Sebastopol : O'Reilly Media, 2022. 616 p.

12. Richards M., Ford N. *Fundamentals of Software Architecture*. 2nd ed. Sebastopol : O'Reilly Media, 2024. 512 p.

13. Martin R. C. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Updated ed. Boston : Prentice Hall, 2022. 432 p.

14. Winters T., Manshreck T., Wright H. *Software Engineering at Google*. Sebastopol : O'Reilly Media, 2023. 602 p.

15. Hoffman A. *Web Application Security*. Sebastopol : O'Reilly Media, 2024. 488 p.

16. Jin B., Sahni S., Shevat A. *Designing Web APIs*. Sebastopol : O'Reilly Media, 2022. 464 p.

17. Masse M. *REST API Design Rulebook*. Updated ed. Sebastopol : O'Reilly Media, 2022. 128 p.

18. Beaulieu A. *Learning SQL*. 3rd ed. Sebastopol : O'Reilly Media, 2021. 394 p.

19. Krug S. *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability*. Updated ed. Berkeley : New Riders, 2022. 216 p.

20. Norman D. *The Design of Everyday Things*. Revised and Expanded Edition. New York : Basic Books, 2023. 368 p.

21. React Documentation. URL: React Official Documentation (дата звернення: 31.05.2026).

22. Node.js Documentation. URL: Node.js Official Documentation (дата звернення: 31.05.2026).

23. Express.js Documentation. URL: Express.js Official Documentation (дата звернення: 31.05.2026).

24. PostgreSQL Documentation. URL: PostgreSQL Official Documentation (дата звернення: 31.05.2026).

25. Nielsen Norman Group. User Experience Research and Reports. URL: Nielsen Norman Group (дата звернення: 31.05.2026).

26. Wattpad Platform. URL: Wattpad Official Website (дата звернення: 31.05.2026).

27. Webnovel Platform. URL: Webnovel Official Website (дата звернення: 31.05.2026).

28. Royal Road Platform. URL: Royal Road Official Website (дата звернення: 31.05.2026).

29. ISO/IEC/IEEE 29119-1:2022. *Software and Systems Engineering — Software Testing — Part 1: General Concepts*. Geneva : International Organization for Standardization, 2022.

30. ISO/IEC 25010:2023. *Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — System and Software Quality Models*. Geneva : International Organization for Standardization, 2023.

31. ДСТУ ISO 9241-5:2005. Ергономічні вимоги до роботи з відеотерміналами. Частина 5. Вимоги до компонування робочого місця та робочої пози оператора. Київ : Держспоживстандарт України.

32. НПАОП 0.00-1.28-10. Правила охорони праці під час експлуатації електронно-обчислювальних машин. Київ, 2010.

33. ДБН В.2.5-28:2018. Природне і штучне освітлення. Київ : Міністерство регіонального розвитку, будівництва та житлово-комунального господарства України, 2018.

34. IEEE. IEEE Software Engineering Standards Collection. New York : IEEE Press, 2024.

35. Association for Computing Machinery. Computing Curricula 2023: Software Engineering. New York : ACM Press, 2023.

ДОДАТКИ

ДОДАТОК А

SQL-скрипти створення таблиць

```
CREATE TABLE roles (
    role_id SERIAL PRIMARY KEY,
    role_name VARCHAR(50) NOT NULL UNIQUE
);

CREATE TABLE users (
    user_id SERIAL PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    role_id INTEGER NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (role_id) REFERENCES roles(role_id)
);

CREATE TABLE genres (
    genre_id SERIAL PRIMARY KEY,
    genre_name VARCHAR(100) NOT NULL UNIQUE
);

CREATE TABLE novels (
    novel_id SERIAL PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    description TEXT,
    cover_image VARCHAR(255),
    author_id INTEGER NOT NULL,
    genre_id INTEGER NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (author_id) REFERENCES users(user_id),
    FOREIGN KEY (genre_id) REFERENCES genres(genre_id)
);

CREATE TABLE chapters (
    chapter_id SERIAL PRIMARY KEY,
    novel_id INTEGER NOT NULL,
    chapter_number INTEGER NOT NULL,
    title VARCHAR(255) NOT NULL,
    content TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (novel_id) REFERENCES novels(novel_id)
        ON DELETE CASCADE
);

CREATE TABLE comments (
    comment_id SERIAL PRIMARY KEY,
    user_id INTEGER NOT NULL,
    novel_id INTEGER NOT NULL,
    content TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(user_id),
```

```
    FOREIGN KEY (novel_id) REFERENCES novels(novel_id)
        ON DELETE CASCADE
);

CREATE TABLE ratings (
    rating_id SERIAL PRIMARY KEY,
    user_id INTEGER NOT NULL,
    novel_id INTEGER NOT NULL,
    rating_value INTEGER NOT NULL CHECK (rating_value BETWEEN 1 AND
5),
    FOREIGN KEY (user_id) REFERENCES users(user_id),
    FOREIGN KEY (novel_id) REFERENCES novels(novel_id)
        ON DELETE CASCADE,
    UNIQUE (user_id, novel_id)
);

CREATE TABLE favorites (
    favorite_id SERIAL PRIMARY KEY,
    user_id INTEGER NOT NULL,
    novel_id INTEGER NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(user_id),
    FOREIGN KEY (novel_id) REFERENCES novels(novel_id)
        ON DELETE CASCADE,
    UNIQUE (user_id, novel_id)
);
```

Додаткові індекси для оптимізації пошуку

```
CREATE INDEX idx_novels_title ON novels(title);
CREATE INDEX idx_novels_author ON novels(author_id);
CREATE INDEX idx_novels_genre ON novels(genre_id);
CREATE INDEX idx_chapters_novel ON chapters(novel_id);
CREATE INDEX idx_comments_novel ON comments(novel_id);
CREATE INDEX idx_ratings_novel ON ratings(novel_id);
```

Початкове наповнення довідників

```
INSERT INTO roles (role_name) VALUES
('Reader'),
('Author'),
('Administrator');

INSERT INTO genres (genre_name) VALUES
('Фентезі'),
('Романтика'),
('Пригоди'),
('Наукова фантастика'),
('Драма'),
('Детектив');
```

ДОДАТОК Б

Фрагменти програмного коду JavaScript

Лістинг Б.1 – Компонент головної сторінки веб-платформи

Фрагмент коду реалізує відображення списку веб-новел на головній сторінці за допомогою бібліотеки React.

```
import React, { useEffect, useState } from 'react';
import axios from 'axios';

function HomePage() {
    const [novels, setNovels] = useState([]);

    useEffect(() => {
        axios.get('/api/novels')
            .then(response => {
                setNovels(response.data);
            })
            .catch(error => {
                console.error(error);
            });
    }, []);
    return (
        <div className="container">
            <h1>Каталог веб-новел</h1>

            {novels.map(novel => (
                <div key={novel.id} className="novel-card">
                    <h2>{novel.title}</h2>
                    <p>{novel.description}</p>
                </div>
            ))}
        </div>
    );
}
export default HomePage;
```

Лістинг Б.2 – Маршрут отримання списку веб-новел

Фрагмент серверного коду реалізує REST API для отримання списку творів.

```
const express = require('express');
const router = express.Router();
const NovelController = require('../controllers/NovelController');

router.get('/novels', NovelController.getAllNovels);

module.exports = router;
```

Лістинг Б.3 – Контролер отримання даних із бази даних

Фрагмент коду забезпечує обробку запиту на отримання переліку веб-новел.

```
const db = require('../config/database');

exports.getAllNovels = async (req, res) => {
    try {
        const result = await db.query(
            'SELECT * FROM novels ORDER BY created_at DESC'
        );

        res.status(200).json(result.rows);
    }
    catch (error) {
        res.status(500).json({
            message: 'Помилка отримання даних'
        });
    }
};
```

Лістинг Б.4 – Реєстрація нового користувача

Фрагмент коду реалізує створення нового облікового запису.

```
const bcrypt = require('bcrypt');

exports.register = async (req, res) => {

    const { username, email, password } = req.body;

    try {

        const passwordHash = await bcrypt.hash(password, 10);

        await db.query(
            `INSERT INTO users
```

```
            (username, email, password_hash, role_id)
            VALUES ($1,$2,$3,$4)`,
            [username, email, passwordHash, 1]
        );

        res.status(201).json({
            message: 'Користувача створено'
        });

    } catch (error) {

        res.status(500).json({
            message: 'Помилка реєстрації'
        });

    }
};
```

Лістинг Д.5 – Авторизація користувача за допомогою JWT

Фрагмент коду реалізує механізм автентифікації користувачів.

```
const jwt = require('jsonwebtoken');

exports.login = async (req, res) => {

    const { email } = req.body;

    const token = jwt.sign(
        { email: email },
        process.env.JWT_SECRET,
        { expiresIn: '24h' }
    );

    res.json({
        token
    });
};
```

Лістинг Б.6 – Створення нової веб-новели

Фрагмент коду забезпечує додавання нового твору до бази даних.

```
exports.createNovel = async (req, res) => {

    const {
        title,
```

```
        description,
        genre_id
    } = req.body;

    try {

        await db.query(
            `INSERT INTO novels
            (title, description, genre_id, author_id)
            VALUES ($1,$2,$3,$4)`,
            [
                title,
                description,
                genre_id,
                req.user.id
            ]
        );

        res.status(201).json({
            message: 'Новелу створено'
        });

    } catch (error) {

        res.status(500).json({
            message: 'Помилка створення новели'
        });

    }
};
```

Лістинг Б.7 – Додавання нового розділу

Фрагмент коду реалізує створення нового розділу веб-новели.

```
exports.createChapter = async (req, res) => {

    const {
        novel_id,
        chapter_number,
        title,
        content
    } = req.body;

    try {

        await db.query(
            `INSERT INTO chapters
            (novel_id, chapter_number, title, content)
            VALUES ($1,$2,$3,$4)`,
            [
```

```
                novel_id,
                chapter_number,
                title,
                content
            ]
        );

        res.status(201).json({
            message: 'Розділ успішно додано'
        });

    } catch (error) {

        res.status(500).json({
            message: 'Помилка додавання розділу'
        });

    }
};
```

Лістинг Д.8 – Пошук веб-новел

Фрагмент коду реалізує механізм пошуку творів за назвою.

```
exports.searchNovel = async (req, res) => {

    const searchText = req.query.q;

    try {

        const result = await db.query(
            `SELECT *
             FROM novels
             WHERE LOWER(title)
             LIKE LOWER($1)`,
            [`%${searchText}%`]
        );

        res.json(result.rows);

    } catch (error) {

        res.status(500).json({
            message: 'Помилка пошуку'
        });

    }
};
```

Лістинг Б.9 – Додавання коментаря до веб-новели

Фрагмент коду забезпечує взаємодію читачів з авторами через систему коментарів.

```
exports.addComment = async (req, res) => {

    const {
        novel_id,
        content
    } = req.body;

    try {

        await db.query(
            `INSERT INTO comments
            (user_id, novel_id, content)
            VALUES ($1,$2,$3)`,
            [
                req.user.id,
                novel_id,
                content
            ]
        );

        res.status(201).json({
            message: 'Коментар додано'
        });

    } catch (error) {

        res.status(500).json({
            message: 'Помилка додавання коментаря'
        });

    }
};
```

Лістинг Б.10 – Маршрутизація клієнтської частини React

Фрагмент коду реалізує навігацію між сторінками веб-платформи.

```
import {
    BrowserRouter,
    Routes,
    Route
```

```
} from 'react-router-dom';

import HomePage from './pages/HomePage';
import NovelPage from './pages/NovelPage';
import LoginPage from './pages/LoginPage';

function App() {

    return (
        <BrowserRouter>
            <Routes>

                <Route
                    path="/"
                    element={<HomePage />}
                />

                <Route
                    path="/novel/:id"
                    element={<NovelPage />}
                />

                <Route
                    path="/login"
                    element={<LoginPage />}
                />

            </Routes>
        </BrowserRouter>
    );
}

export default App;
```

Наведені у додатку фрагменти програмного коду демонструють реалізацію основних функціональних компонентів веб-платформи для публікації та читання веб-новел, включаючи клієнтську частину на основі React, серверну логіку на Node.js та Express.js, механізми роботи з базою даних PostgreSQL, автентифікацію користувачів і реалізацію REST API. Це додатково підтверджує використання мови програмування JavaScript, зазначеної в темі кваліфікаційної роботи.

ДОДАТОК В

Структураклієнтської частини