

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка web-застосунку для ведення домашньої
бібліотеки та трекінгу прочитаного матеріалу»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Юлія ЗДОРОВЕНКО
(підпис)

Виконала: здобувачка вищої освіти групи ПД-44

_____ Юлія ЗДОРОВЕНКО

Керівник: _____ Вікторія КОРЕЦЬКА
канд.пед.наук, доц.

Рецензент: _____

Київ 2026

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут інформаційних технологій**

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного забезпечення

_____ Ірина ЗАМРІЙ

«_____» _____ 2026 р.

**ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

_____ Здоровенко Юлії Сергіївни _____

1. Тема кваліфікаційної роботи: «Розробка web-застосунку для ведення домашньої бібліотеки та трекінгу прочитаного матеріалу».

керівник кваліфікаційної роботи канд.пед.наук, доц. Вікторія КОРЕЦЬКА

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026 року № 51.

2. Строк подання кваліфікаційної роботи «29» травня 2026 року.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про процес організації домашньої бібліотеки, ведення читачького прогресу, принципи роботи вебплатформ для обліку книг, підходи до формування персональної статистики та рекомендацій, особливості контролю правового статусу цифрових матеріалів, принципи побудови клієнт-серверних web-застосунків, технічна документація щодо використання технологій Java, Spring Boot, Spring Security, JWT, Spring Data JPA, PostgreSQL, Liquibase, React, Vite, TypeScript, Tailwind CSS, Docker Compose та REST API.

4. Зміст розрахунково-пояснювальної записки:

1. Огляд та аналіз предметної галузі ведення домашньої бібліотеки.

2. Аналіз існуючих програмних засобів для обліку книг, ведення читацьких списків та контролю читацького прогресу.
3. Огляд засобів реалізації web-платформи ShelfMate.
4. Проектування структури, бази даних та основних UML-діаграм web-платформи.
5. Програмна реалізація та опис функціонування web-платформи.
6. Тестування web-платформи ShelfMate.
7. Апробація результатів дослідження.
8. Перелік графічного матеріалу: презентація.

5. Вимоги до web-застосунку ShelfMate.

1. Логічна структура системи.
2. Діаграма варіантів використання web-платформи ShelfMate.
3. Блок-схема процесу авторизації користувача.
4. Блок-схема роботи користувача з бібліотекою.
5. Блок-схема перевірки правового статусу цифрового матеріалу.
6. Структура бази даних web-платформи.
7. Клієнт-серверна архітектура web-платформи.
8. Схема обміну даними між клієнтською частиною, серверною частиною та базою даних через REST API.
9. Діаграма класів серверної частини застосунку.
10. Діаграма діяльності формування читацького паспорта.
11. Блок-схема формування рекомендацій користувачу.
12. Діаграма розгортання web-платформи.
13. Схема та результати тестування програмного забезпечення.

6. Дата видачі завдання «23» лютого 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури з теми ведення домашньої бібліотеки, організації читацького прогресу та web-застосунків для обліку книг	23.02–01.03.2026	
2	Аналіз існуючих аналогів програмного забезпечення для ведення книжкових колекцій, читацьких списків та контролю прогресу читання	02.03–08.03.2026	
3	Аналіз предметної галузі ведення домашньої бібліотеки та визначення вимог до web-платформи ShelfMate	09.03–16.03.2026	

4	Огляд засобів реалізації web-платформи ShelfMate	17.03–23.03.2026	
5	Проектування web-платформи ShelfMate, структури системи, бази даних та UML-діаграм	24.03–06.04.2026	
6	Програмна реалізація серверної та клієнтської частини web-платформи	07.04–20.04.2026	
7	Реалізація модулів читацького паспорта, рекомендацій, контролю правового статусу цифрових матеріалів та адміністративної панелі	21.04–27.04.2026	
8	Тестування web-платформи ShelfMate та оформлення результатів тестування	28.04–04.05.2026	
9	Оформлення роботи: вступ, висновки, реферат, список джерел та додатки	05.05–11.05.2026	
10	Розробка демонстраційних матеріалів	12.05–17.05.2026	
11	Попередній захист роботи	18.05–22.05.2026	

Здобувач вищої освіти

(підпис)

Юлія ЗДОРОВЕНКО

Керівник

кваліфікаційної роботи

(підпис)

Вікторія КОРЕЦЬКА

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 60 стор., 3 табл., 23 рис., 29 джерел.

Мета роботи – спрощення ведення домашньої бібліотеки за рахунок розробки вебплатформи ShelfMate для обліку книг, контролю читацького прогресу, формування персональної статистики, рекомендацій та керування доступом до цифрових матеріалів.

Об'єкт дослідження – процес організації, систематизації та контролю читацького прогресу в межах домашньої бібліотеки.

Предмет дослідження – засоби, методи та технології реалізації вебплатформи ShelfMate для ведення домашньої бібліотеки, обліку книг, формування читацького паспорта та контролю правового статусу цифрових матеріалів.

Короткий зміст роботи: у роботі проаналізовано предметну галузь ведення домашньої бібліотеки та визначено основні проблеми, пов'язані з неструктурованим зберіганням інформації про книги, складністю пошуку потрібних видань, відсутністю єдиного інструменту для контролю читацького прогресу, розрізненістю нотаток та оцінок, а також правовою невизначеністю щодо доступу до цифрових матеріалів. Встановлено, що користувачам потрібна персональна вебплатформа, яка поєднує каталог книг, читацький щоденник, статистику, рекомендації та контроль доступу до електронних матеріалів.

Проаналізовано існуючі програмні засоби для ведення книжкових колекцій та читацьких списків: Excel, Google Sheets, Goodreads, The StoryGraph, LibraryThing та Libib. Визначено їх переваги, недоліки та функціональні обмеження. Особливу увагу приділено можливостям додавання книг, пошуку, фільтрації, ведення нотаток, оцінювання, відстеження прогресу, формування статистики, персоналізації, рекомендацій та контролю правового статусу цифрових матеріалів.

Розроблено структуру вебплатформи ShelfMate та визначено її ключові функціональні можливості: реєстрацію та авторизацію користувачів, перегляд власної бібліотеки, додавання, редагування та видалення книг, пошук і фільтрацію, зміну статусу читання, додавання нотаток та оцінок, відстеження прогресу за сторінками, перегляд детальної сторінки книги, редагування профілю користувача, формування читацького паспорта, створення рекомендацій, роботу адміністративної панелі та зміну правового статусу цифрових матеріалів.

У роботі використано Java та Spring Boot для реалізації серверної частини, Spring Security і JWT для авторизації та розмежування доступу, Spring Data JPA і Hibernate для роботи з базою даних, PostgreSQL для зберігання даних, Liquibase для керування міграціями бази даних, React, Vite, TypeScript і Tailwind CSS для створення клієнтського інтерфейсу, Docker Compose для запуску окремих компонентів системи, а також REST API для організації взаємодії між клієнтською та серверною частинами.

Сферою використання вебплатформи є індивідуальне ведення домашньої бібліотеки, організація особистих книжкових колекцій, контроль читацького прогресу, збереження нотаток і оцінок, формування персональної статистики читання та відповідальна робота з цифровими матеріалами. Розроблена платформа може бути використана студентами, викладачами, активними читачами, власниками великих домашніх бібліотек та користувачами, які хочуть систематизувати інформацію про прочитані, активні й заплановані книги.

КЛЮЧОВІ СЛОВА: SHELFMATE, WEB-ПЛАТФОРМА, ДОМАШНЯ БІБЛІОТЕКА, КАТАЛОГ КНИГ, ЧИТАЦЬКИЙ ПРОГРЕС, ЧИТАЦЬКИЙ ПАСПОРТ, РЕКОМЕНДАЦІЇ, ЦИФРОВІ МАТЕРІАЛИ, ПРАВОВИЙ СТАТУС, JAVA, SPRING BOOT, SPRING SECURITY, JWT, POSTGRESQL, LIQUIBASE, REACT, TYPESCRIPT, TAILWIND CSS, REST API.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	11
ВСТУП.....	13
1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ІСНУЮЧИХ РІШЕНЬ ДЛЯ ВЕДЕННЯ ДОМАШНЬОЇ БІБЛІОТЕКИ	17
1.1 Загальна характеристика предметної галузі.....	17
1.2 Проблеми організації домашньої бібліотеки.....	19
1.3 Цільова аудиторія вебплатформи ShelfMate	20
1.4 Аналіз існуючих способів та програмних засобів для ведення домашньої бібліотеки	21
1.4.1 Паперові списки та звичайні нотатки	22
1.4.2 Електронні таблиці Excel та Google Sheets.....	22
1.4.3 Goodreads	23
1.4.4 The StoryGraph.....	24
1.4.5 LibraryThing.....	26
1.4.6 Libib	27
1.5 Порівняльний аналіз існуючих рішень	28
1.6 Загальна характеристика вебплатформи ShelfMate	30
1.7 Визначення функціональних вимог до вебплатформи	32
1.8 Визначення нефункціональних вимог до вебплатформи	33
2 ЗАСОБИ РЕАЛІЗАЦІЇ ВЕБПЛАТФОРМИ SHELFMATE	35
2.1 Загальна характеристика засобів реалізації.....	35
2.2 Середовище розробки	36
2.3 Мова програмування Java.....	37
2.4 Фреймворк Spring Boot.....	38
2.5 Spring Security та JWT	39
2.6 Spring Data JPA.....	40
2.7 Система управління базами даних PostgreSQL	40
2.8 Liquibase	42
2.9 React.....	42
2.10 TypeScript	43
2.11 REST API.....	44
2.12 Docker та Docker Compose.....	45
3 ПРОЄКТУВАННЯ ВЕБПЛАТФОРМИ SHELFMATE	46
3.1 Логічна структура вебплатформи ShelfMate	46
3.2 Клієнт-серверна архітектура системи.....	48
3.3 Діаграма варіантів використання	48

3.4	Проектування бази даних	50
3.5	Проектування основних сутностей системи.....	52
3.6	Проектування процесу авторизації	54
3.7	Проектування адміністративної панелі.....	55
3.8	Діаграма послідовності додавання книги	57
3.9	Діаграма розгортання системи	58
4	РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБПЛАТФОРМИ SHELFMATE	60
4.1	Загальна характеристика реалізації вебплатформи.....	60
4.2	Реалізація серверної частини.....	60
4.3	Реалізація авторизації та розмежування доступу	61
4.4	Реалізація клієнтської частини.....	62
4.5	Реалізація сторінки входу та реєстрації	63
4.6	Реалізація головної сторінки бібліотеки	65
4.7	Реалізація додавання та редагування книги	66
4.8	Реалізація детальної сторінки книги	66
4.11	Реалізація профілю користувача	67
4.12	Реалізація читацького паспорта	68
4.14	Реалізація контролю правового статусу цифрових матеріалів	69
4.15	Реалізація адміністративної панелі	70
4.17	Тестування вебплатформи	71
4.18	Результати функціонального тестування.....	72
4.20	Розгортання та запуск системи.....	74
	ВИСНОВКИ	75
	ПЕРЕЛІК ПОСИЛАНЬ	78
	ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ.....	80
	ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ	89

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – інтерфейс прикладного програмування, що використовується для взаємодії між програмними компонентами системи.

REST – архітектурний стиль організації обміну даними між клієнтською та серверною частинами web-застосунку.

HTTP – протокол передавання даних у мережі Інтернет.

HTTPS – захищена версія протоколу HTTP, що забезпечує безпечний обмін даними між клієнтом і сервером.

JSON – текстовий формат обміну даними, який використовується для передавання інформації між frontend- і backend-частинами системи.

UI – користувацький інтерфейс програмного забезпечення.

UX – користувацький досвід під час взаємодії з програмним продуктом.

HTML – мова розмітки гіпертексту, що використовується для створення структури web-сторінок.

CSS – мова стилів, що використовується для оформлення зовнішнього вигляду web-сторінок.

JavaScript – мова програмування, що використовується для створення інтерактивної поведінки web-сторінок.

TypeScript – мова програмування, що розширює JavaScript за рахунок статичної типізації.

React – бібліотека JavaScript для створення компонентного користувацького інтерфейсу.

Vite – інструмент для швидкого створення, запуску та збирання frontend-застосунків.

Axios – бібліотека для виконання HTTP-запитів із клієнтської частини web-застосунку.

React Router – бібліотека для організації маршрутизації між сторінками React-застосунку.

Java – об'єктно-орієнтована мова програмування, що використовується для реалізації серверної частини застосунку.

Spring – фреймворк для створення серверних Java-застосунків.

Spring Boot – інструмент екосистеми Spring, що спрощує створення та запуск серверних web-застосунків.

Spring Security – модуль Spring для реалізації автентифікації, авторизації та захисту доступу до ресурсів системи.

JWT – формат токена, що використовується для безпечної передачі даних авторизації між клієнтом і сервером.

JPA – специфікація Java для роботи з реляційними базами даних через об'єктну модель.

Hibernate – фреймворк об'єктно-реляційного відображення, який забезпечує зв'язок між Java-класами та таблицями бази даних.

PostgreSQL – об'єктно-реляційна система керування базами даних.

ORM – технологія об'єктно-реляційного відображення, що забезпечує зв'язок між об'єктами програми та таблицями бази даних.

CRUD – базові операції роботи з даними: створення, читання, оновлення та видалення.

Liquibase – інструмент для керування змінами структури бази даних за допомогою міграцій.

SPA – односторінковий web-застосунок, у якому основна взаємодія з користувачем відбувається без повного перезавантаження сторінки.

IDE – інтегроване середовище розробки програмного забезпечення.

Git – система контролю версій, що використовується для відстеження змін у програмному коді.

Memory Game – ігровий формат, у якому користувач відкриває картки та шукає правильні пари між словами й перекладами.

ВСТУП

У сучасних умовах розвитку інформаційних технологій та цифровізації повсякденних процесів усе більшого значення набувають вебплатформи, які дозволяють користувачам систематизувати особисті дані, зберігати інформацію в електронному вигляді та швидко отримувати доступ до потрібних матеріалів. Однією з таких сфер є ведення домашньої бібліотеки, оскільки значна частина користувачів має не лише друковані книги, а й електронні матеріали, читацькі списки, нотатки, оцінки та власні плани щодо читання. За відсутності єдиного цифрового інструменту така інформація часто зберігається у різних місцях: паперових записниках, електронних таблицях, нотатках у смартфоні або сторонніх онлайн-сервісах [7; 8].

Однією з поширених проблем організації домашньої бібліотеки є відсутність централізованого каталогу книг. Користувач може мати значну кількість видань, але не завжди швидко пам'ятає, які книги вже є в бібліотеці, які з них прочитані, які перебувають у процесі читання, а які тільки заплановані. Особливо це актуально для студентів, викладачів, науковців, активних читачів і користувачів, які працюють із професійною або навчальною літературою. У таких умовах виникає потреба у зручному вебзастосунку, який дозволить не лише зберігати інформацію про книги, а й керувати читацьким процесом [7; 8].

Іншою важливою проблемою є складність відстеження читацького прогресу. Традиційні способи обліку книг здебільшого фіксують лише сам факт наявності книги, але не забезпечують зручного контролю статусу читання, поточної сторінки, кількості прочитаних книг, середньої оцінки або виконання читацької цілі. У результаті користувач може втрачати інформацію про те, на якому етапі читання він перебуває, які книги давно не оновлювалися та які матеріали варто прочитати наступними. Саме тому актуальним є створення системи, яка поєднує каталог книг, читацький щоденник і персональну статистику [26; 27].

Окремого значення набуває питання роботи з цифровими матеріалами. Наявність електронної копії книги не означає, що її можна вільно поширювати, завантажувати або надавати іншим користувачам. У контексті вебплатформи для домашньої бібліотеки важливо враховувати авторське право, статус доступу до цифрового матеріалу та можливість обмеження дій із PDF-файлами або зовнішніми джерелами. Тому система повинна розрізняти книги, які доступні лише як опис, особисті копії, матеріали суспільного надбання, ліцензовані джерела або зовнішні легальні посилання [1; 6].

Актуальність теми кваліфікаційної роботи зумовлена необхідністю створення зручної вебплатформи для ведення домашньої бібліотеки, яка дозволяє користувачу зберігати інформацію про книги, контролювати статус читання, вести нотатки, оцінювати прочитане, переглядати персональну статистику та працювати з цифровими матеріалами з урахуванням їх правового статусу. На відміну від звичайних електронних таблиць або загальних книжкових сервісів, розроблювана платформа ShelfMate орієнтована саме на персональне ведення домашньої бібліотеки, формування читацького паспорта та відповідальне керування доступом до цифрових матеріалів [26; 27; 28; 29].

Вебплатформа ShelfMate передбачає роботу двох основних ролей: USER та ADMIN. Звичайний користувач може зареєструватися, увійти до системи, переглядати власну бібліотеку, додавати книги, редагувати та видаляти записи, змінювати статус читання, додавати нотатки, ставити оцінки, переглядати детальну сторінку книги, редагувати профіль, переглядати читацький паспорт і отримувати рекомендації. Адміністратор є розширеною роллю користувача та додатково має доступ до адміністративної панелі, де може переглядати користувачів, усі книги в системі, загальну статистику, змінювати ролі користувачів і керувати правовим статусом цифрових матеріалів [2; 24].

Особливою функціональною можливістю платформи є читацький паспорт. Він формується на основі бібліотеки користувача та відображає загальну кількість книг, кількість прочитаних, активних і запланованих книг, улюблений жанр, середню оцінку, прогрес за сторінками, виконання читацької цілі та рекомендації.

Завдяки цьому користувач бачить не просто список книг, а персональний читацький портрет, який допомагає краще організувати власну читацьку активність [26; 27].

Ще однією важливою особливістю ShelfMate є механізм контролю правового статусу цифрових матеріалів. Для кожної книги може бути визначено тип доступу: тільки опис, особиста копія, суспільне надбання, ліцензований матеріал, зовнішнє легальне джерело або заблокований доступ. Такий підхід дозволяє не перетворювати платформу на неконтрольоване сховище PDF-файлів, а використовувати її як відповідальний інструмент для організації бібліотеки та роботи з інформацією про книги [1; 6].

Мета роботи – спрощення ведення домашньої бібліотеки за рахунок розробки вебплатформи ShelfMate для обліку книг, контролю читацького прогресу, формування персональної статистики, рекомендацій та керування доступом до цифрових матеріалів.

Об'єкт дослідження – процес організації, систематизації та контролю читацького прогресу в межах домашньої бібліотеки [7; 8].

Предмет дослідження – засоби, методи та технології реалізації вебплатформи ShelfMate для ведення домашньої бібліотеки, обліку книг, формування читацького паспорта та контролю правового статусу цифрових матеріалів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Проаналізувати предметну галузь ведення домашньої бібліотеки та визначити основні проблеми організації, пошуку, систематизації й контролю читацького прогресу [7; 8].
2. Дослідити існуючі програмні засоби для ведення книжкових колекцій, читацьких списків та обліку книг, визначити їх переваги, недоліки й функціональні обмеження [26; 27; 28; 29].
3. Сформувати функціональні та нефункціональні вимоги до вебплатформи ShelfMate.

4. Обґрунтувати вибір технологій для реалізації клієнтської частини, серверної частини, бази даних, авторизації та контейнерного запуску системи [9; 10; 12; 14; 16; 18].
5. Спроекувати логічну структуру вебплатформи ShelfMate, базу даних, UML-діаграми та основні алгоритми роботи системи [24].
6. Реалізувати основні модулі вебплатформи: авторизацію, профіль користувача, керування книгами, пошук, фільтрацію, читацький паспорт, рекомендації та адміністративну панель [10; 12; 13; 18].
7. Розробити механізм контролю правового статусу цифрових матеріалів для обмеження доступу до PDF-файлів відповідно до типу доступу книги [1; 6].
8. Провести тестування вебплатформи ShelfMate та проаналізувати результати перевірки основних функцій системи.

Методи дослідження. У бакалаврській роботі використовуються такі методи: аналіз предметної галузі, порівняльний аналіз існуючих програмних засобів для ведення домашньої бібліотеки, об'єктно-орієнтоване проєктування, UML-моделювання, проєктування структури бази даних, розробка клієнт-серверного вебзастосунку та тестування програмного забезпечення [24].

Наукова новизна одержаних результатів полягає у розробці структури вебплатформи ShelfMate, яка поєднує ведення домашньої бібліотеки, контроль читацького прогресу, формування читацького паспорта, персональні рекомендації та механізм перевірки правового статусу цифрових матеріалів.

Практична значущість одержаних результатів полягає в тому, що розроблена вебплатформа може бути використана для індивідуального ведення домашньої бібліотеки, організації особистих книжкових колекцій, контролю читацького прогресу, збереження нотаток і оцінок, а також відповідальної роботи з цифровими матеріалами. У майбутньому систему можна розширити додаванням вбудованого PDF-reader, окремих цитат, календаря читання, нагадувань, імпорту книг з відкритих бібліотек, спільних книжкових полиць, сімейного доступу та розширених рекомендацій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ІСНУЮЧИХ РІШЕНЬ ДЛЯ ВЕДЕННЯ ДОМАШНЬОЇ БІБЛІОТЕКИ

1.1 Загальна характеристика предметної галузі

У сучасних умовах значна частина користувачів зберігає інформацію про власні книги у різних неструктурованих джерелах: паперових списках, електронних таблицях, нотатках у смартфоні, месенджерах або окремих онлайн-сервісах. Такий підхід може бути прийнятним при невеликій кількості книг, однак із часом він стає незручним, оскільки користувачеві складно швидко знайти потрібне видання, визначити його статус читання, переглянути власні нотатки або оцінити загальний читацький прогрес [7; 8].

Предметною галуззю даної кваліфікаційної роботи є процес організації, зберігання, систематизації та аналізу інформації про книги в межах домашньої бібліотеки. Домашня бібліотека може містити як друковані, так і цифрові книги, тому для користувача важливо не лише зберігати назву та автора, а й мати можливість фіксувати додаткові характеристики: жанр, опис, рік публікації, мову, видавництво, ISBN, статус читання, оцінку, нотатки, кількість сторінок, поточну сторінку та джерело отримання книги.

Особливої актуальності набуває питання контролю читацького прогресу. Користувач може одночасно читати декілька книг, частину книг планувати до читання, а частину вже завершити. Без спеціалізованого інструменту такий процес часто стає хаотичним: користувач забуває, на якій сторінці зупинився, які книги давно не оновлювалися, які твори були оцінені високо, а які залишились лише у списку запланованих. Тому вебплатформа для ведення домашньої бібліотеки повинна не просто зберігати записи, а допомагати користувачеві керувати власним читацьким процесом.

У межах розробленої платформи ShelfMate домашня бібліотека розглядається як персональний цифровий простір користувача, у якому поєднуються каталог книг, читацький щоденник, система нотаток, персональна

статистика, рекомендації та контроль доступу до цифрових матеріалів. Такий підхід дозволяє відійти від простого CRUD-застосунку та сформувати більш повноцінний інструмент для організації читання.

Окремою особливістю предметної галузі є правовий статус цифрових матеріалів. Наявність книги у персональній бібліотеці не означає, що її цифрову копію можна вільно розповсюджувати або завантажувати. Тому у ShelfMate передбачено механізм визначення статусу доступу до цифрового матеріалу: тільки опис, особиста копія, суспільне надбання, ліцензований матеріал або зовнішнє легальне джерело. Це дозволяє враховувати питання авторського права та не перетворювати платформу на неконтрольоване сховище PDF-файлів [1; 6].

Таким чином, предметна галузь охоплює не лише облік книг, а й ширший комплекс задач: персоналізацію бібліотеки, відстеження читацької активності, формування статистики, підтримку ролей користувачів, захист персональних даних та етичне керування цифровими матеріалами [2; 5].



Рис. 1.1 Загальна структура предметної галузі вебплатформи ShelfMate

1.2 Проблеми організації домашньої бібліотеки

У традиційному вигляді ведення домашньої бібліотеки часто не має єдиної структури. Користувач може зберігати частину інформації у паперовому записнику, частину — в електронній таблиці, частину — у нотатках телефону або взагалі не фіксувати її. Через це з часом виникає проблема дублювання, втрати інформації або складності швидкого пошуку потрібної книги.

Однією з основних проблем є відсутність централізованого каталогу. Якщо бібліотека містить десятки або сотні книг, користувачеві стає складно пам'ятати, які книги вже є в наявності, які були прочитані, які очікують читання, а які тільки планується придбати. Особливо це актуально для користувачів, які читають не лише художню літературу, а й навчальні, професійні або науково-популярні матеріали.

Другою проблемою є недостатнє відстеження читацького прогресу. Більшість простих способів обліку дозволяють лише записати факт наявності книги, але не надають зручного механізму для контролю поточної сторінки, статусу читання, дати оновлення прогресу або виконання читацької цілі. У результаті користувач може втрачати мотивацію, забувати про книги, які давно почав читати, або не бачити реальної динаміки власного читання.

Третьою проблемою є розрізненість нотаток і оцінок. Під час читання користувач може робити короткі записи, зберігати власні враження, оцінювати книгу або позначати важливі думки. Якщо такі записи не пов'язані безпосередньо з конкретною книгою, їх важко знайти повторно. Це знижує практичну цінність домашньої бібліотеки як персонального інформаційного простору [26; 27; 28; 29].

Четвертою проблемою є обмежена персоналізація існуючих інструментів. Багато сервісів орієнтовані або на соціальну взаємодію, або на загальні книжкові рейтинги, або на публічні списки. Проте користувачу домашньої бібліотеки часто потрібен саме приватний, зручний і зрозумілий інструмент, який буде адаптований під його власні читацькі звички, а не під публічну активність у спільноті.

П'ятою проблемою є правова невизначеність щодо цифрових копій книг. Якщо система дозволяє додавати посилання на PDF або цифровий файл, вона повинна враховувати, чи має користувач право на доступ, читання або завантаження такого матеріалу. Без цього вебзастосунок може створювати ризики некоректного використання захищених авторським правом матеріалів [1; 6].

Отже, для ефективного ведення домашньої бібліотеки потрібна система, яка забезпечує структуроване збереження книг, підтримку статусів читання, нотаток, оцінок, персональної статистики, рекомендацій та контрольованого доступу до цифрових матеріалів.

1.3 Цільова аудиторія вебплатформи ShelfMate

Цільовою аудиторією вебплатформи ShelfMate є користувачі, які мають власну домашню бібліотеку та хочуть систематизувати інформацію про книги в електронному вигляді. До такої аудиторії можна віднести студентів, школярів старших класів, викладачів, науковців, читачів художньої літератури, фахівців, які працюють із професійними матеріалами, а також користувачів, які регулярно купують або читають книги.

Для студентів і викладачів важливо мати можливість швидко знаходити навчальні та професійні джерела, залишати нотатки, розподіляти книги за темами або жанрами та контролювати, які матеріали вже опрацьовані. Для активних читачів художньої літератури більш важливими є статуси читання, оцінки, персональна статистика, рекомендації та можливість бачити власний читацький прогрес.

Окрему групу користувачів становлять люди, які мають велику кількість друкованих книг удома. Для них важливим є створення зручного каталогу, у якому можна швидко перевірити, чи є певна книга в бібліотеці, хто її автор, до якого жанру вона належить і чи була вона вже прочитана. Такий користувач не обов'язково потребує складної соціальної платформи, але потребує простого й надійного інструменту для персонального обліку.

Також ShelfMate може бути корисною для користувачів, які працюють з електронними книгами. У цьому випадку важливо не лише зберігати інформацію про книгу, а й розуміти, чи доступний цифровий матеріал для читання або завантаження. Саме тому у системі передбачено правовий статус доступу до матеріалу, що робить платформу більш відповідальною з точки зору використання цифрового контенту.

Основними потребами цільової аудиторії є:

- швидке додавання нової книги до бібліотеки;
- зручний пошук за назвою, автором або жанром;
- фільтрація книг за статусом читання;
- збереження нотаток та оцінок;
- перегляд читацького прогресу;
- формування персональної статистики;
- отримання простих рекомендацій;
- захист особистих даних;
- контроль доступу до цифрових матеріалів.

Таким чином, ShelfMate орієнтована на користувачів, яким потрібен не публічний книжковий рейтинг або соціальна мережа, а персональна вебплатформа для організації власної бібліотеки та читацької активності.

1.4 Аналіз існуючих способів та програмних засобів для ведення домашньої бібліотеки

Для вирішення задач ведення домашньої бібліотеки можуть використовуватися різні підходи: паперові списки, електронні таблиці, універсальні нотатки, спеціалізовані онлайн-сервіси та мобільні додатки. Кожен із цих способів має певні переваги, однак більшість із них не забезпечує повного поєднання персонального каталогу, читацького прогресу, нотаток, статистики, рекомендацій та контролю правового статусу цифрових матеріалів [26; 27; 28; 29].

1.4.1 Паперові списки та звичайні нотатки

Найпростішим способом ведення домашньої бібліотеки є використання паперових списків або звичайних нотаток. Користувач може записувати назви книг, авторів, короткі коментарі або список літератури, яку планує прочитати. Такий підхід не потребує спеціальних технічних знань, доступу до Інтернету або встановлення окремого програмного забезпечення.

Перевагою паперових списків є простота використання. Користувач самостійно визначає структуру записів і може швидко зафіксувати потрібну інформацію. Однак при збільшенні кількості книг цей спосіб стає незручним. У паперовому списку складно виконувати пошук, сортування, фільтрацію, редагування та оновлення інформації. Також неможливо автоматично підрахувати кількість прочитаних книг, визначити улюблений жанр або сформувати статистику.

Звичайні нотатки у смартфоні частково вирішують проблему доступності, однак вони також не є повноцінним інструментом для ведення бібліотеки. Нотатки не мають спеціалізованої структури, не підтримують зв'язки між книгами, статусами, оцінками та прогресом читання. Тому такий підхід доцільний лише для невеликої кількості записів.

1.4.2 Електронні таблиці Excel та Google Sheets

Електронні таблиці є більш структурованим способом ведення домашньої бібліотеки. У Microsoft Excel або Google Sheets користувач може створити таблицю з окремими колонками для назви книги, автора, жанру, року видання, статусу читання, оцінки та нотаток. Також таблиці дозволяють використовувати сортування, фільтрацію та базові формули.

Перевагами такого підходу є гнучкість налаштування, можливість самостійно додавати потрібні поля, зручний експорт даних і доступність інструментів для більшості користувачів. Якщо користувач має базові навички

роботи з таблицями, він може створити власний каталог без розробки окремого програмного продукту.

Водночас електронні таблиці мають низку недоліків. Вони не мають спеціалізованого інтерфейсу для роботи з книгами, не відображають записи у вигляді зручних карток, не забезпечують повноцінної роботи з профілем користувача, читацьким паспортом або рекомендаціями. Усі зв'язки між даними потрібно створювати вручну, що ускладнює використання таблиць для постійного ведення бібліотеки. Крім того, таблиці не забезпечують зручного розмежування ролей користувача та адміністратора, а також не вирішують питання правового доступу до цифрових матеріалів.

1.4.3 Goodreads

Goodreads є одним із найвідоміших онлайн-сервісів для обліку прочитаних книг, пошуку літератури, перегляду оцінок і формування читацьких списків. Сервіс дозволяє користувачам додавати книги до власних полиць, встановлювати статус читання, писати рецензії, оцінювати книги та переглядати активність інших читачів [26].

До переваг Goodreads можна віднести велику базу книг, наявність статусів читання, систему оцінювання, можливість залишати рецензії та отримувати рекомендації. Сервіс також має елементи соціальної взаємодії, що дозволяє користувачам бачити думки інших читачів і формувати власні списки на основі популярності книг.

Однак Goodreads більше орієнтований на соціальну складову та публічну читацьку активність, ніж на ведення персональної домашньої бібліотеки. Для користувача, який хоче створити приватний каталог власних книг, можливості сервісу можуть бути надмірними або недостатньо гнучкими. Також сервіс не надає спеціального механізму контролю правового статусу цифрових матеріалів і не орієнтований на адміністрування власної локальної платформи [26].

У порівнянні з Goodreads, ShelfMate робить акцент не на соціальній взаємодії, а на персональному обліку, читацькому паспорті, власній статистиці, приватності даних і контрольованому доступі до цифрових матеріалів [26].

Deciding what to read next?

You're in the right place. Tell us what titles or genres you've enjoyed in the past, and we'll give you surprisingly insightful recommendations.

What are your friends reading?

Chances are your friends are discussing their favorite (and least favorite) books on Goodreads.

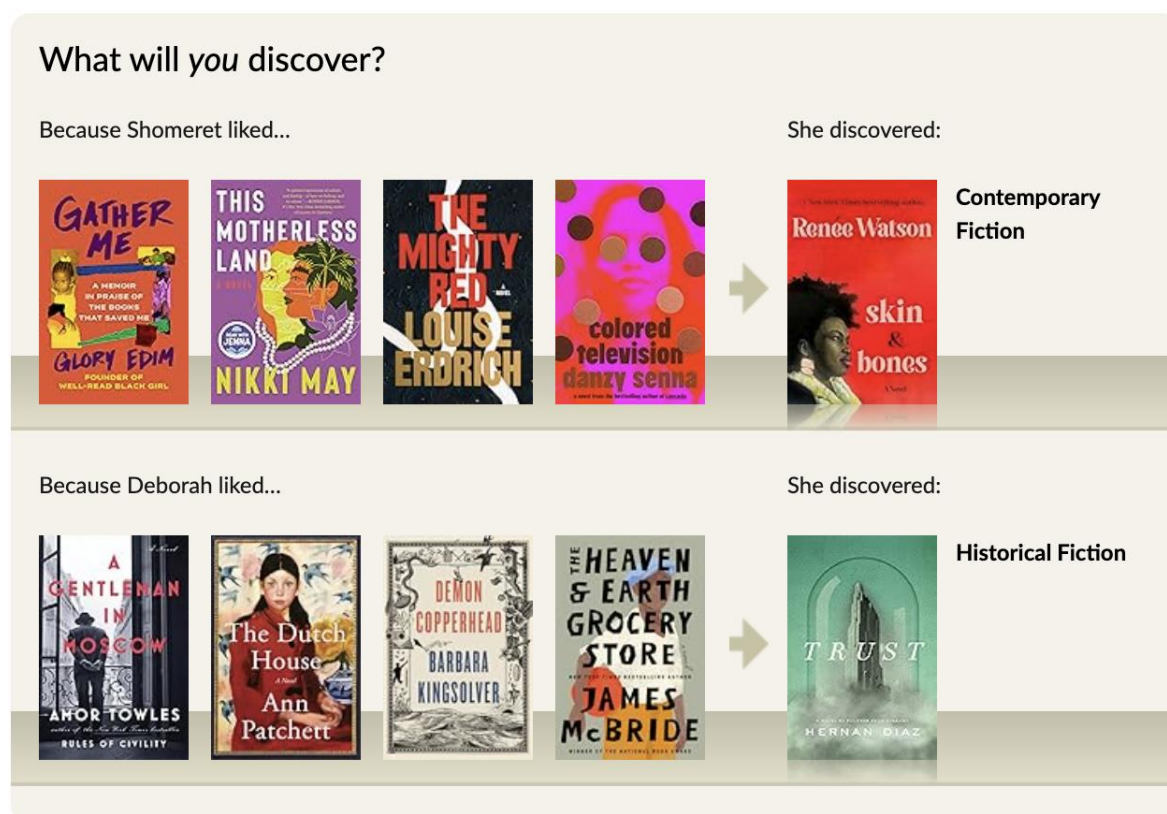


Рис. 1.2 Приклад екранних форм сервісу Goodreads

1.4.4 The StoryGraph

The StoryGraph є сучасним сервісом для відстеження читацького прогресу та аналізу читацьких звичок. На відміну від Goodreads, цей сервіс більше зосереджений на персональній статистиці, налаштуваннях книг, темпі читання та рекомендаціях. Користувач може додавати книги, встановлювати статус читання,

оцінювати прочитане, переглядати статистику та отримувати рекомендації на основі власних вподобань [27].

Перевагами The StoryGraph є сучасний інтерфейс, зручне відстеження прогресу, наявність персональної аналітики та більший фокус на індивідуальному користувачеві. Сервіс дозволяє краще бачити власні читацькі звички, ніж звичайні таблиці або прості списки [27].

Недоліком The StoryGraph є те, що він більше орієнтований на аналіз читання, ніж на ведення повноцінного каталогу домашньої бібліотеки. Можливості створення власних полів, контролю локальних записів, адміністрування користувачів або перевірки правового статусу цифрових матеріалів є обмеженими. Крім того, користувач залежить від зовнішнього сервісу та його структури.

ShelfMate враховує переваги персональної статистики, але доповнює їх функціями домашнього каталогу, профілю користувача, читацького паспорта, рекомендацій з власної бібліотеки та контролю доступу до цифрових матеріалів [27].

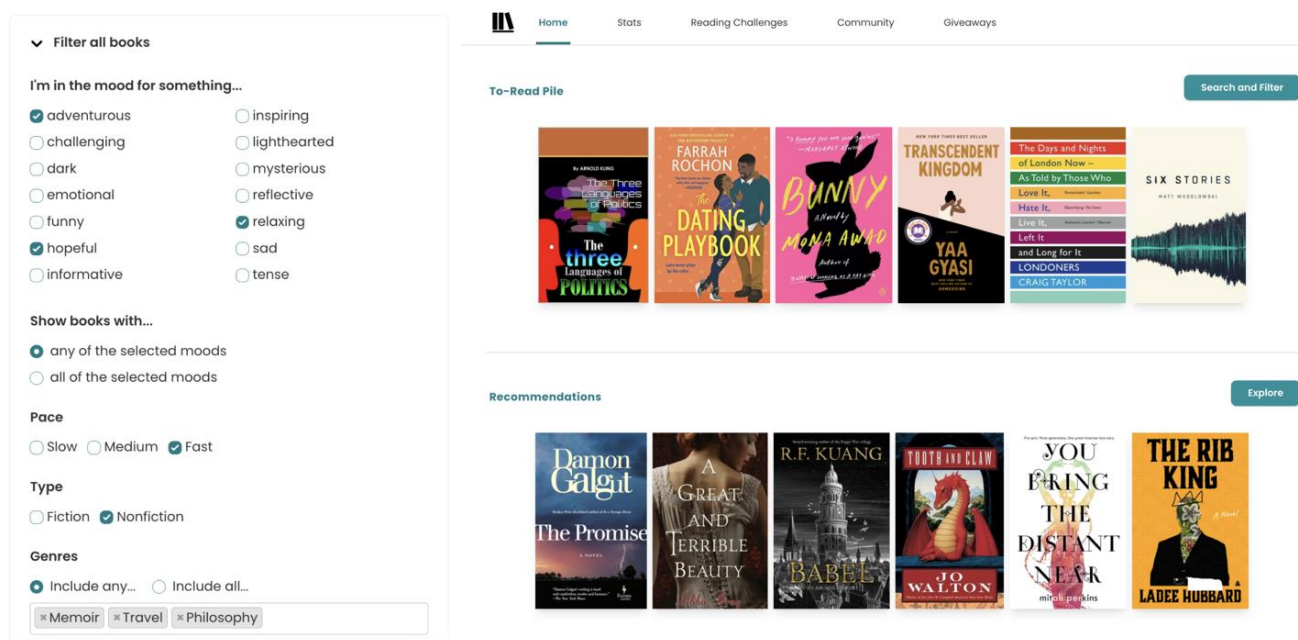


Рис. 1.3 Приклад екранних форм сервісу The StoryGraph

1.4.5 LibraryThing

LibraryThing є спеціалізованим сервісом для каталогізації книг, який дозволяє створювати персональну бібліотеку, додавати бібліографічні дані, використовувати теги, сортувати записи та формувати тематичні колекції. У порівнянні з Goodreads, цей сервіс має більш виражену спрямованість саме на облік і класифікацію літератури [28].

Перевагою LibraryThing є глибша каталогізація книг, підтримка тегів, можливість детального опису книжкових записів і зручність для користувачів із великою кількістю книг. Такий сервіс може бути корисним для тих, хто хоче створити структуровану бібліотеку з великою кількістю параметрів [28].

Водночас LibraryThing може бути менш зручним для користувачів, які очікують простого сучасного вебінтерфейсу, швидкої роботи зі статусами читання, персонального читацького паспорта та зрозумілої статистики. Сервіс більше орієнтований на каталогізацію, ніж на щоденне відстеження читацького процесу. Також у ньому відсутній акцент на правовому статусі цифрових матеріалів.

ShelfMate, на відміну від LibraryThing, поєднує каталогізацію з простим інтерфейсом, контролем прогресу, читацькою статистикою, рекомендаціями та адміністративними можливостями [28].



Рис. 1.4 Приклад екранних форм сервісу LibraryThing

1.4.6 Libib

Libib є сервісом для каталогізації книг, фільмів, музики та інших медіа. Він дозволяє створювати власні колекції, додавати записи, використовувати пошук і керувати бібліотекою через вебінтерфейс або мобільний застосунок. Для користувачів, які хочуть обліковувати не лише книги, а й інші типи матеріалів, Libib може бути досить корисним інструментом [29].

До переваг Libib можна віднести підтримку різних типів колекцій, можливість створення особистого каталогу, зручний пошук і наявність базових інструментів для організації записів. Сервіс підходить для користувачів, які хочуть систематизувати домашні медіаресурси [29].

Однак Libib не повністю відповідає задачам, які вирішує ShelfMate. Він не зосереджений саме на читацькому прогресі, не формує повноцінний читацький паспорт, не надає спеціалізованої логіки рекомендацій на основі власної бібліотеки

та не має окремого механізму контролю правового статусу цифрових матеріалів. Крім того, частина розширених можливостей може бути обмежена умовами використання сервісу.

ShelfMate є більш вузько орієнтованою платформою саме для роботи з книгами, тому її функціональність краще відповідає потребам користувача домашньої бібліотеки [29].

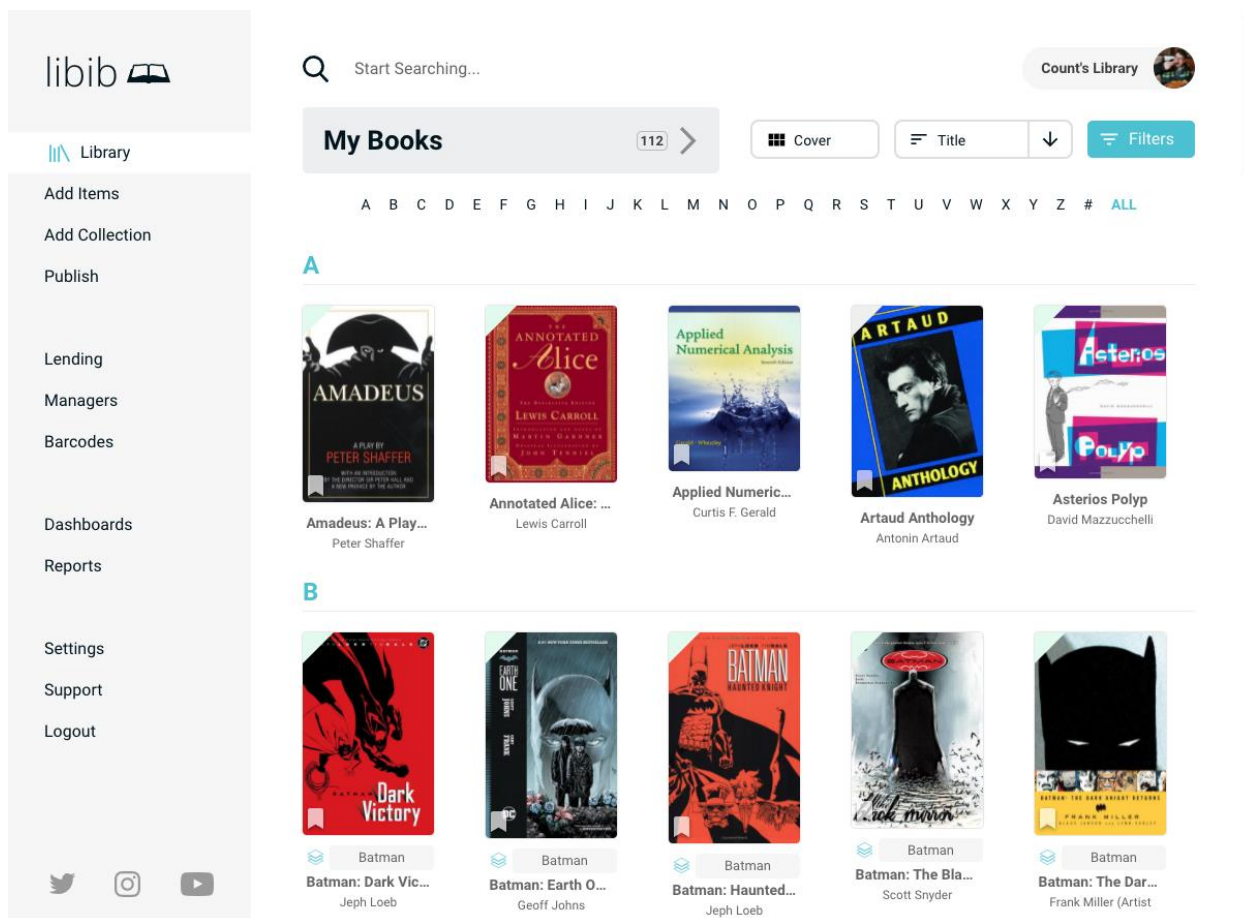


Рис. 1.5 Приклад екранних форм сервісу Libib

1.5 Порівняльний аналіз існуючих рішень

Для визначення переваг і недоліків існуючих засобів було проведено порівняння паперових списків, електронних таблиць, Goodreads, The StoryGraph, LibraryThing, Libib та розроблюваної вебплатформи ShelfMate. Порівняння виконано за основними критеріями, які є важливими для ведення домашньої бібліотеки: додавання книг, підтримка статусів читання, нотатки, оцінки,

статистика, персоналізація, рекомендації, контроль цифрових матеріалів та розмежування ролей.

Таблиця 1.1

Порівняльний аналіз програмних засобів для ведення домашньої бібліотеки

Критерій	Goodreads	The StoryGraph	LibraryThing	Libib	ShelfMate
Додавання книг	+	+	+	+	+
Редагування інформації про книгу	обмежене редагування	обмежене редагування	+	+	+
Пошук за назвою або автором	+	+	+	+	+
Фільтрація за жанром	базова фільтрація	+	+	+	+
Статуси читання	+	+	часткова підтримка	часткова підтримка	+
Збереження нотаток	+	обмежені нотатки	+	обмежені нотатки	+
Оцінювання книг	+	+	+	базове оцінювання	+
Відстеження прогресу за сторінками	часткове відстеження	+	часткове відстеження	-	+
Персональна статистика	базова статистика	+	базова статистика	базова статистика	+
Читацький паспорт	-	окремі елементи статистик	-	-	+
Рекомендації з власної бібліотеки	+	+	часткові рекомендації	-	+
Розмежування ролей користувача й адміністратора	-	-	-	обмежене адміністрування	+

За результатами порівняння можна зробити висновок, що існуючі рішення частково вирішують задачі ведення домашньої бібліотеки, однак не забезпечують повного набору функцій, необхідних для персонального вебзастосунку. Паперові списки та електронні таблиці є простими, але не мають спеціалізованого інтерфейсу й автоматизованої статистики. Goodreads і The StoryGraph краще працюють із читацькою активністю, але більше орієнтовані на зовнішні сервіси та соціальні або аналітичні функції. LibraryThing і Libib корисні для каталогізації, однак не надають повного поєднання читацького паспорта, рекомендацій, адміністративної панелі та контролю правового статусу цифрових матеріалів.

ShelfMate поєднує переваги електронного каталогу, читацького щоденника та персональної аналітики. Основною відмінністю платформи є наявність читацького паспорта та етичного режиму доступу до цифрових матеріалів, що робить систему більш завершеною з точки зору сучасного вебзастосунку.

1.6 Загальна характеристика вебплатформи ShelfMate

ShelfMate — це вебплатформа для ведення домашньої бібліотеки, контролю читацького прогресу та організації інформації про книги. Платформа дозволяє користувачам створювати власну цифрову бібліотеку, додавати книги, змінювати статус читання, вести нотатки, оцінювати прочитане, переглядати персональну статистику та отримувати рекомендації на основі власної бібліотеки.

Основна ідея ShelfMate полягає в тому, щоб об'єднати в одному застосунку три функціональні напрями: каталог домашніх книг, читацький щоденник і просту аналітику прогресу. Користувач може швидко зрозуміти, які книги вже прочитані, які читаються зараз, які тільки заплановані та які давно не оновлювалися [1; 6].

У системі передбачено дві основні ролі: звичайний користувач та адміністратор. Звичайний користувач працює зі своєю бібліотекою, додає та редагує книги, встановлює статуси читання, залишає нотатки, оцінює книги, редагує власний профіль і переглядає читацький паспорт. Адміністратор має

ширші можливості: він може переглядати користувачів, бачити загальну статистику платформи, контролювати книги, змінювати ролі та керувати правовим статусом цифрових матеріалів.

Платформа складається з кількох основних сторінок: сторінки входу, сторінки реєстрації, головної сторінки бібліотеки, детальної сторінки книги, сторінки профілю, читацького паспорта та адміністративної панелі. Така структура дозволяє логічно розділити функції системи та зробити взаємодію користувача з платформою зрозумілою.

Головна сторінка бібліотеки є основним робочим місцем користувача. На ній відображаються картки книг, пошук, фільтри за статусом читання та жанром, коротка статистика й кнопка додавання нової книги. Кожна книга містить основну інформацію: назву, автора, жанр, статус, оцінку, прогрес читання та обкладинку.

Детальна сторінка книги містить повну інформацію про вибране видання. Користувач може переглянути опис, статус читання, оцінку, поточний прогрес, нотатки, джерело та правовий статус цифрового матеріалу. Доступність кнопок читання або завантаження PDF залежить від правового статусу книги [2].

Читацький паспорт є однією з ключових функцій ShelfMate. Він формує персональний читацький портрет користувача на основі його бібліотеки та прогресу. У ньому відображається загальна кількість книг, кількість прочитаних книг, активні книги, улюблений жанр, середня оцінка, прогрес за сторінками, виконання читацької цілі та рекомендації.

Адміністративна панель призначена для контролю системи. Вона дозволяє адміністратору переглядати користувачів, книги, статистику та правові статуси матеріалів. Це демонструє розмежування прав доступу та дозволяє підтримувати порядок у системі [1; 6].

Таким чином, ShelfMate є не просто застосунком для збереження списку книг, а повноцінною вебплатформою для організації домашньої бібліотеки, аналізу читацького прогресу та відповідальної роботи з цифровими матеріалами.

1.7 Визначення функціональних вимог до вебплатформи

Функціональні вимоги визначають основні можливості, які повинна забезпечувати вебплатформа ShelfMate. Вони формуються на основі аналізу предметної галузі, потреб цільової аудиторії та недоліків існуючих рішень [2; 24].

До функціональних вимог вебплатформи ShelfMate належать:

- реєстрація користувача, що дозволяє новому користувачу створити власний акаунт у системі;
- авторизація користувача за допомогою email і пароля;
- розмежування ролей звичайного користувача та адміністратора;
- перегляд власної бібліотеки, де користувач бачить тільки додані ним книги;
- додавання нової книги до домашньої бібліотеки;
- редагування інформації про книгу, зокрема назви, автора, жанру, опису, статусу читання, оцінки та нотаток;
- видалення книги зі своєї бібліотеки;
- пошук книг за назвою або автором;
- фільтрація книг за жанром і статусом читання;
- встановлення статусу читання книги: заплановано, читаю або прочитано;
- оцінювання книги користувачем;
- додавання та збереження власних нотаток до книги;
- контроль читацького прогресу за допомогою загальної кількості сторінок і поточної сторінки;
- перегляд детальної сторінки книги з повною інформацією про вибране видання;
- перегляд і редагування профілю користувача;
- формування читацького паспорта з персональною статистикою читання;
- формування рекомендацій на основі власної бібліотеки користувача;
- доступ адміністратора до адміністративної панелі, загальної статистики, списку користувачів і книг;
- керування ролями користувачів адміністратором;

- контроль правового статусу цифрових матеріалів адміністратором.

Функціональні вимоги демонструють, що ShelfMate повинна забезпечувати повний цикл роботи з домашньою бібліотекою: від реєстрації користувача до формування персональної статистики й адміністрування прав доступу [2].

1.8 Визначення нефункціональних вимог до вебплатформи

Нефункціональні вимоги описують якісні характеристики системи: швидкість роботи, безпеку, зручність використання, адаптивність, надійність і можливість подальшого розвитку. Для вебплатформи ShelfMate такі вимоги є важливими, оскільки система працює з персональними даними користувачів і повинна забезпечувати стабільну взаємодію між клієнтською частиною, сервером і базою даних [2; 4; 5; 24].

До нефункціональних вимог вебплатформи ShelfMate належать:

- доступ до системи повинен здійснюватися через авторизацію з використанням JWT-токена;
- користувач повинен мати доступ лише до власної бібліотеки;
- основні запити до системи повинні оброблятися протягом 1–2 секунд;
- інтерфейс повинен коректно відображатися на різних розмірах екрана;
- основні дії користувача повинні бути зрозумілими без спеціальної технічної підготовки;
- система повинна коректно зберігати дані та повідомляти користувача про помилки;
- архітектура повинна дозволяти додавати нові функції без повної переробки системи;
- код системи повинен бути розділений на логічні модулі: контролери, сервіси, репозиторії, сутності та frontend-компоненти;
- система повинна працювати в сучасних веббраузерах;

- адміністративні функції повинні бути доступні тільки користувачам з роллю адміністратора;
- система повинна перевіряти введені дані перед збереженням;
- доступ до PDF або зовнішніх джерел повинен залежати від правового статусу книги;
- система повинна забезпечувати стабільну взаємодію між клієнтською частиною, сервером і базою даних;
- структура проєкту повинна бути придатною для подальшого розширення, зокрема додавання цитат, reading sessions, PDF-reader, спільних полиць або розширених рекомендацій;
- запуск серверної частини та бази даних повинен бути спрощений за допомогою Docker Compose.

Нефункціональні вимоги забезпечують якість роботи платформи та визначають обмеження, яких необхідно дотримуватися під час реалізації. Особливу увагу потрібно приділити захисту персональних даних, розмежуванню доступу та коректній роботі з цифровими матеріалами.

2 ЗАСОБИ РЕАЛІЗАЦІЇ ВЕБПЛАТФОРМИ SHELFMATE

2.1 Загальна характеристика засобів реалізації

Для реалізації вебплатформи ShelfMate було обрано клієнт-серверний підхід, за якого система поділяється на три основні частини: клієнтську частину, серверну частину та базу даних. Такий підхід є доцільним для вебзастосунку, оскільки дозволяє розділити відповідальність між інтерфейсом користувача, бізнес-логікою та збереженням даних.

Клієнтська частина відповідає за взаємодію користувача із системою. Саме через неї користувач реєструється, входить до акаунта, переглядає власну бібліотеку, додає книги, змінює статус читання, залишає нотатки, переглядає читацький паспорт і працює з профілем. Для реалізації клієнтської частини було використано React, що дозволяє створювати динамічний, адаптивний та зручний інтерфейс [18; 19].

Серверна частина відповідає за обробку запитів, авторизацію, перевірку ролей, роботу з книгами, користувачами, профілями, статистикою та адміністративними функціями. Для її реалізації було використано мову програмування Java та фреймворк Spring Boot. Такий вибір обґрунтований стабільністю, поширеністю та зручністю Spring Boot для створення REST API [9; 10; 11].

Для збереження даних використовується реляційна база даних PostgreSQL. У ній зберігається інформація про користувачів, книги, профільні дані, статуси читання, оцінки, нотатки та правовий статус цифрових матеріалів. Для керування структурою бази даних застосовується Liquibase, що дозволяє контрольовано створювати та оновлювати таблиці за допомогою міграцій [14; 15].

Для запуску окремих компонентів системи використовується Docker Compose. Він дозволяє розгорнути базу даних і серверну частину у контейнерах, що спрощує налаштування середовища та робить запуск проєкту більш стабільним.

Клієнтська частина запускається окремо через Vite, який забезпечує швидку збірку React-застосунку під час розробки [16; 17; 19].

Таким чином, обраний стек технологій забезпечує надійну роботу вебплатформи, можливість подальшого розвитку, розмежування клієнтської та серверної логіки, захист даних і зручне адміністрування системи [3; 4; 5; 10; 14; 18].

2.2 Середовище розробки

Під час розробки вебплатформи ShelfMate використовувались сучасні середовища та інструменти, які дозволяють ефективно працювати з backend- і frontend-частинами проєкту.

Для розробки серверної частини доцільним є використання IntelliJ IDEA. Це інтегроване середовище розробки добре підходить для Java-проєктів, підтримує Spring Boot, Maven або Gradle, роботу з базами даних, автодоповнення коду, навігацію між класами, запуск тестів і вбудовану перевірку помилок. Завдяки цьому розробник може швидко працювати з контролерами, сервісами, репозиторіями, сутностями та конфігураційними файлами.

Для клієнтської частини може використовуватись Visual Studio Code. Це середовище зручне для роботи з React, TypeScript, HTML, CSS і Tailwind CSS. Воно підтримує розширення для форматування коду, перевірки синтаксису, роботи з Git, запуску npm-команд і швидкого перегляду структури проєкту.

Також під час розробки використовуються додаткові інструменти:

- Postman або аналогічний інструмент для перевірки REST API;
- pgAdmin або вбудовані засоби IDE для перегляду бази даних;
- Git для контролю версій;
- Docker Desktop для запуску контейнерів;
- термінал для запуску backend-, frontend- та docker-команд.

Загалом середовище розробки було підібрано таким чином, щоб забезпечити зручну роботу з усіма частинами системи: серверною логікою, клієнтським інтерфейсом, базою даних і контейнерним запуском.

2.3 Мова програмування Java

Для реалізації серверної частини вебплатформи ShelfMate було використано мову програмування Java. Java є однією з найпоширеніших мов для створення корпоративних, серверних і веборієнтованих систем. Вона має статичну типізацію, велику кількість бібліотек, стабільну екосистему та хорошу підтримку фреймворків для розробки backend-застосунків [9].

У межах ShelfMate Java використовується для реалізації основної бізнес-логіки системи. Саме на серверній частині обробляються запити користувачів, виконується авторизація, перевіряються ролі, зберігаються книги, формується читацький паспорт, обчислюється статистика та реалізуються адміністративні функції [9; 10].

Основними перевагами Java для даного проєкту є:

- стабільність і надійність при роботі з серверними застосунками;
- підтримка об'єктно-орієнтованого програмування;
- зручна інтеграція зі Spring Boot;
- велика кількість готових бібліотек;
- підтримка роботи з базами даних;
- можливість масштабування системи в майбутньому [9; 10].

Java також добре підходить для побудови систем, у яких потрібно чітко розділяти логіку на окремі рівні: контролери, сервіси, репозиторії, сутності та конфігурації. Це відповідає архітектурі ShelfMate, оскільки система має окрему логіку для авторизації, роботи з книгами, профілем, статистикою та адмініструванням.

2.4 Фреймворк Spring Boot

Spring Boot є основним фреймворком, який використовується для реалізації backend-частини ShelfMate. Він спрощує створення Java-застосунків, автоматизує конфігурацію, дозволяє швидко запускати сервер і надає готові механізми для створення REST API, роботи з базою даних, безпеки та обробки запитів [10; 11].

У межах платформи ShelfMate Spring Boot використовується для створення серверного застосунку, який приймає запити від клієнтської частини, обробляє їх і повертає відповідь у форматі JSON. Наприклад, коли користувач відкриває головну сторінку бібліотеки, React-клієнт надсилає запит до backend-частини. Spring Boot обробляє цей запит, звертається до бази даних через відповідний сервіс і репозиторій, отримує книги користувача та повертає їх на frontend [10].

Основні задачі Spring Boot у проєкті:

- запуск серверної частини вебплатформи;
- створення REST-контролерів;
- обробка HTTP-запитів;
- взаємодія з базою даних;
- підключення Spring Security;
- обробка помилок;
- формування відповідей для frontend-частини [10; 11].

Spring Boot дозволяє зручно організувати структуру backend-проєкту. Наприклад, контролери відповідають за приймання HTTP-запитів, сервіси містять бізнес-логіку, репозиторії забезпечують доступ до бази даних, а сутності описують структуру таблиць [10].

Такий підхід робить код більш зрозумілим, підтримуваним і придатним для подальшого розширення.

2.5 Spring Security та JWT

Оскільки ShelfMate працює з персональними бібліотеками користувачів, важливо забезпечити захист даних і правильне розмежування доступу. Для цього у серверній частині використовується Spring Security у поєднанні з JWT-токенами [12; 25].

Spring Security відповідає за контроль доступу до ресурсів системи. Він дозволяє визначити, які сторінки або API-методи доступні всім користувачам, які — лише авторизованим користувачам, а які — тільки адміністраторам. Наприклад, реєстрація та вхід мають бути доступні без авторизації, а перегляд власної бібліотеки — лише після входу в систему. Адміністративна панель повинна бути доступна тільки користувачам із роллю ADMIN [12].

JWT використовується для збереження інформації про авторизованого користувача. Після успішного входу backend перевіряє email і пароль. Якщо дані правильні, система створює токен, який frontend зберігає та надсилає при наступних запитах. Backend перевіряє токен і визначає, хто саме виконує запит [25].

Такий механізм дозволяє:

- не пускати неавторизованих користувачів до закритих сторінок;
- показувати користувачу тільки його власну бібліотеку;
- захищати адміністративні функції;
- перевіряти роль користувача;
- уникати постійного повторного введення логіна й пароля.

У контексті ShelfMate це особливо важливо, оскільки кожен користувач має працювати лише зі своїми книгами, нотатками, оцінками та профільними даними. Адміністратор, у свою чергу, має доступ до ширших функцій, але ці можливості повинні бути чітко обмежені роллю [24].

2.6 Spring Data JPA

Для взаємодії серверної частини з базою даних використовується Spring Data JPA. Цей інструмент спрощує роботу з реляційними базами даних і дозволяє працювати з таблицями через Java-класи та репозиторії [13].

У ShelfMate основними сутностями є користувач і книга. Користувач має власну бібліотеку, профільні дані та роль у системі. Книга містить назву, автора, жанр, опис, статус читання, оцінку, нотатки, кількість сторінок, поточну сторінку, обкладинку, джерело та правовий статус цифрового матеріалу.

Spring Data JPA дозволяє описати ці сутності у вигляді Java-класів, а потім автоматично виконувати типові операції з базою даних:

- створення нового запису;
- отримання запису за ідентифікатором;
- пошук записів за певними параметрами;
- оновлення даних;
- видалення записів;
- фільтрація книг за користувачем, жанром або статусом [13].

Наприклад, для отримання книг конкретного користувача можна створити метод у репозиторії, який знаходить усі книги за ідентифікатором власника. Це зручно для реалізації приватності даних, оскільки користувач бачить тільки власні записи [13].

Використання Spring Data JPA зменшує кількість шаблонного SQL-коду та робить роботу з базою даних більш структурованою.

2.7 Система управління базами даних PostgreSQL

Для збереження даних вебплатформи ShelfMate використовується PostgreSQL. Це реляційна система управління базами даних, яка підходить для зберігання структурованої інформації та підтримує зв'язки між таблицями [14].

У базі даних ShelfMate зберігаються:

- користувачі;
- книги;
- профільні дані;
- ролі користувачів;
- статуси читання;
- оцінки;
- нотатки;
- інформація про правовий статус цифрових матеріалів [14].

Основними таблицями системи є `users` та `books`. Таблиця `users` зберігає інформацію про користувачів: `email`, пароль, ім'я, роль та інші дані, необхідні для авторизації. Таблиця `books` містить інформацію про книги, які користувач додає до своєї бібліотеки. Кожна книга прив'язана до конкретного користувача, що дозволяє розмежувати бібліотеки різних акаунтів [14; 17].

PostgreSQL було обрано через такі переваги:

- підтримка реляційної моделі даних;
- стабільність і надійність;
- можливість створення зв'язків між таблицями;
- підтримка складних запитів;
- сумісність зі Spring Boot і Spring Data JPA;
- зручність запуску через Docker Compose [14].

Для ShelfMate реляційна база даних є доцільною, оскільки дані мають чітку структуру: користувач має багато книг, кожна книга має власні параметри, статуси, оцінки та нотатки. У майбутньому структуру бази можна розширити, додавши окремі таблиці для цитат, сеансів читання, жанрів, файлів або спільних книжкових полиць.

2.8 Liquibase

Для керування структурою бази даних у ShelfMate використовується Liquibase. Цей інструмент дозволяє створювати, змінювати та контролювати структуру таблиць за допомогою міграцій [15].

Під час розробки вебзастосунку структура бази даних може змінюватися. Наприклад, спочатку таблиця books може містити лише базові поля: назву, автора та жанр. Пізніше до неї потрібно додати статус читання, оцінку, нотатки, кількість сторінок, ISBN, опис, обкладинку або правовий статус доступу до цифрового матеріалу. Якщо виконувати такі зміни вручну, можна легко втратити контроль над версіями бази даних.

Liquibase вирішує цю проблему, оскільки кожна зміна описується як окрема міграція. Це дозволяє:

- відстежувати історію змін структури бази даних;
- автоматично створювати таблиці під час запуску застосунку;
- безпечно додавати нові поля;
- уникати ручного налаштування бази даних;
- синхронізувати структуру бази між різними середовищами [15].

У межах ShelfMate Liquibase використовується для створення таблиць користувачів і книг, а також для подальшого розширення структури бази даних. Це робить розробку більш контрольованою та зручною [15].

2.9 React

Для реалізації клієнтської частини вебплатформи ShelfMate використовується React. Це JavaScript-бібліотека для створення користувацьких інтерфейсів, яка дозволяє будувати застосунок із окремих компонентів [18].

У ShelfMate React відповідає за відображення сторінок, форм, кнопок, карток книг, профілю користувача, читацького паспорта та адміністративної панелі.

Компонентний підхід дозволяє розділити інтерфейс на окремі частини та повторно використовувати їх у різних місцях застосунку [18].

Наприклад, картка книги може бути окремим компонентом, який використовується на головній сторінці бібліотеки, у рекомендаціях або в адміністративній панелі. Форма додавання книги також може бути окремим компонентом, який відповідає за введення назви, автора, жанру, статусу, оцінки, нотаток та інших параметрів.

React забезпечує швидке оновлення інтерфейсу без повного перезавантаження сторінки. Якщо користувач змінює статус книги або додає нову книгу, відповідна частина інтерфейсу може оновитися динамічно. Це робить взаємодію з платформою зручнішою [18].

Основні задачі React у ShelfMate:

- реалізація сторінки входу та реєстрації;
- відображення головної бібліотеки;
- створення форм додавання та редагування книг;
- реалізація пошуку і фільтрації;
- відображення детальної сторінки книги;
- реалізація профілю користувача;
- формування читацького паспорта;
- створення адміністративної панелі;
- взаємодія з backend через REST API [18].

Таким чином, React є основним засобом створення клієнтської частини ShelfMate.

2.10 TypeScript

Для підвищення надійності клієнтської частини може використовуватися TypeScript. TypeScript є розширенням JavaScript, яке додає статичну типізацію. Це

дозволяє краще контролювати структуру даних, які надходять від backend-частини, і зменшує кількість помилок під час розробки [20].

У ShelfMate TypeScript доцільно використовувати для опису типів користувача, книги, профілю, статусу читання, правового статусу та відповідей API. Наприклад, можна описати тип Book, який містить поля id, title, author, genre, status, rating, notes, totalPages, currentPage, coverUrl та legalAccessStatus [20].

Переваги використання TypeScript:

- кращий контроль над структурою даних;
- виявлення частини помилок ще до запуску застосунку;
- зручніше автодоповнення коду;
- простіша підтримка великих frontend-проектів;
- зрозуміліші інтерфейси між компонентами [20].

Для ShelfMate це важливо, оскільки frontend працює з великою кількістю даних про книги, користувачів, статуси, статистику та рекомендації. Типізація допомагає зробити код більш передбачуваним і підтримуваним.

2.11 REST API

Взаємодія між клієнтською та серверною частинами ShelfMate здійснюється через REST API. Це означає, що frontend надсилає HTTP-запити до backend-частини, а сервер повертає відповідь у форматі JSON.

REST API використовується для таких дій:

- реєстрація користувача;
- авторизація користувача;
- отримання списку книг;
- додавання нової книги;
- редагування книги;
- видалення книги;
- отримання детальної інформації про книгу;
- отримання профілю користувача;

- оновлення профілю;
- формування читацького паспорта;
- отримання рекомендацій;
- робота з адміністративною панеллю [25].

Наприклад, коли користувач додає нову книгу, frontend надсилає POST-запит із даними книги на backend. Сервер перевіряє токен користувача, обробляє дані, зберігає книгу в базі даних і повертає результат. Якщо користувач не авторизований або не має відповідних прав, сервер повертає помилку доступу.

REST API дозволяє чітко розділити клієнтську та серверну частини. Це спрощує подальший розвиток системи, оскільки frontend і backend можуть змінюватися незалежно один від одного за умови збереження узгодженого API.

2.12 Docker та Docker Compose

Для спрощення запуску серверної частини та бази даних використовується Docker Compose. Docker дозволяє запускати окремі частини системи у контейнерах, а Docker Compose — описати декілька сервісів в одному конфігураційному файлі [16; 17].

Це дозволяє не встановлювати базу даних вручну на кожному комп'ютері, а запускати її в контейнері з уже заданими параметрами: назвою бази, користувачем, паролем і портом. Backend також може запускатися у власному контейнері та підключатися до бази даних [16; 17].

Frontend-частина під час розробки може запускатися окремо через Vite, оскільки це зручніше для швидкої перевірки змін в інтерфейсі. У майбутньому її також можна контейнеризувати.

3 ПРОЄКТУВАННЯ ВЕБПЛАТФОРМИ SHELFMATE

3.1 Логічна структура вебплатформи ShelfMate

Логічна структура вебплатформи ShelfMate відображає основні функціональні блоки системи та зв'язки між ними. Центральним елементом платформи є користувач, який взаємодіє з бібліотекою через вебінтерфейс. Після авторизації користувач отримує доступ до власної бібліотеки, профілю, читацького паспорта та рекомендацій [10; 18; 14; 25].

Основними логічними модулями ShelfMate є:

- модуль авторизації та реєстрації;
- модуль керування книгами;
- модуль пошуку та фільтрації;
- модуль читацького прогресу;
- модуль нотаток та оцінок;
- модуль профілю користувача;
- модуль читацького паспорта;
- модуль рекомендацій;
- модуль контролю правового статусу цифрових матеріалів;
- адміністративний модуль [25].

Модуль авторизації та реєстрації забезпечує створення акаунта, вхід до системи та перевірку користувача за допомогою JWT-токена. Модуль керування книгами відповідає за додавання, редагування, видалення та перегляд книг. Модуль пошуку та фільтрації дозволяє швидко знаходити книги за назвою, автором, жанром або статусом читання.

Модуль читацького прогресу дає змогу користувачу вказувати загальну кількість сторінок і поточну сторінку, а система на основі цих даних може

розраховувати відсоток виконання. Модуль нотаток і оцінок дозволяє залишати власні записи щодо книги та фіксувати особисте враження після прочитання.

Модуль профілю користувача зберігає персональні дані, улюблений жанр, читацьку ціль і коротку інформацію про користувача. Модуль читацького паспорта формує узагальнену статистику: кількість книг, кількість прочитаних книг, активні книги, середню оцінку, улюблений жанр і прогрес до читацької цілі.

Модуль рекомендацій працює на основі простих правил і пропонує користувачу книги зі статусом “Заплановано”, книги з улюбленого жанру або книги, які давно не оновлювалися. Модуль контролю правового статусу цифрових матеріалів визначає, чи доступні користувачу функції читання або завантаження PDF.

Адміністративний модуль доступний лише користувачам із роллю адміністратора. Він дозволяє переглядати користувачів, книги, загальну статистику, керувати ролями та контролювати правовий статус матеріалів [2; 24].

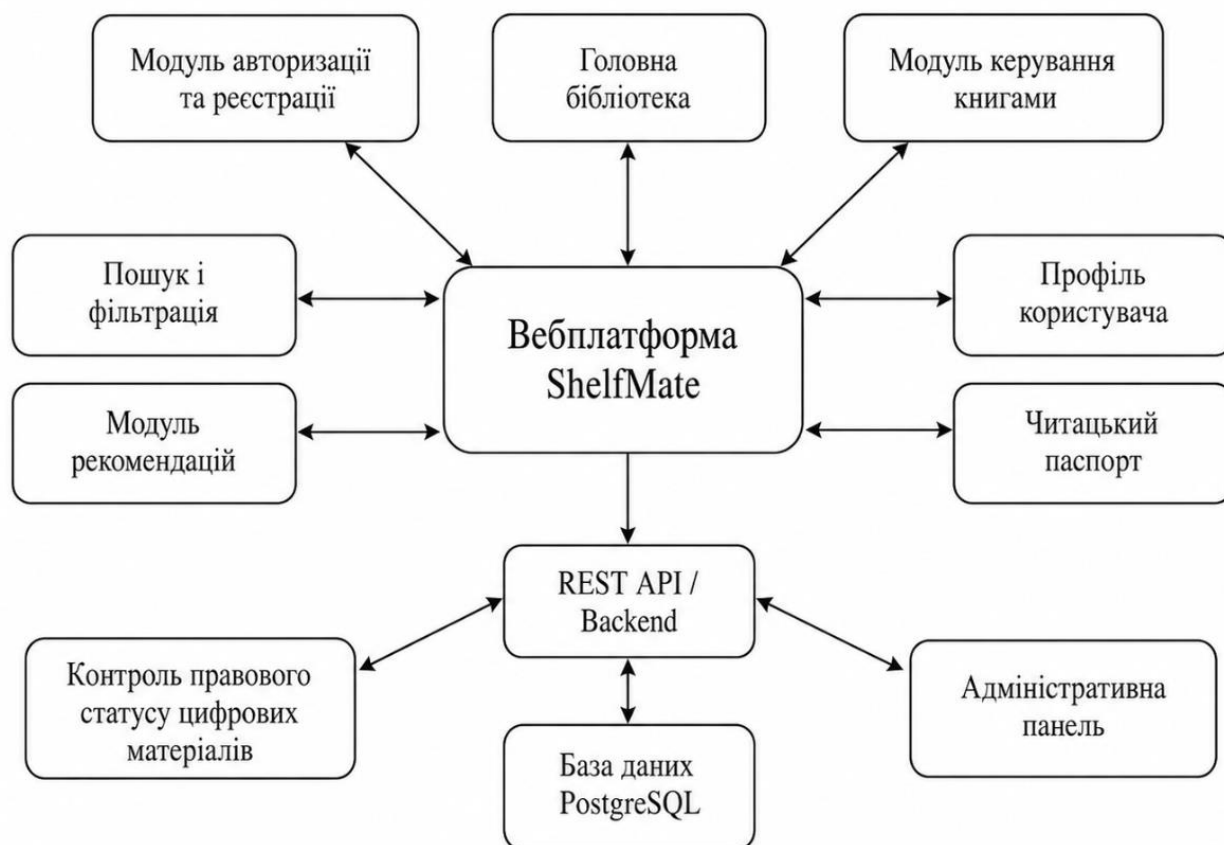


Рис. 3.1 Логічна структура вебплатформи ShelfMate

3.2 Клієнт-серверна архітектура системи

Вебплатформа ShelfMate побудована за клієнт-серверною архітектурою. Такий підхід дозволяє розділити систему на окремі рівні, кожен із яких виконує власні задачі. Клієнтська частина відповідає за інтерфейс користувача, серверна частина — за обробку запитів і бізнес-логіку, а база даних — за збереження інформації [18; 10; 14].

Клієнтська частина реалізована за допомогою React. Вона відображає сторінки входу, реєстрації, бібліотеки, детальної інформації про книгу, профілю, читацького паспорта та адміністративної панелі. Користувач взаємодіє саме з клієнтською частиною: заповнює форми, натискає кнопки, обирає фільтри, переглядає книги та змінює їхній статус.

Серверна частина реалізована на Java з використанням Spring Boot. Вона приймає HTTP-запити від frontend-частини, перевіряє JWT-токен, визначає роль користувача, виконує необхідну бізнес-логіку та звертається до бази даних. Наприклад, якщо користувач додає книгу, backend перевіряє авторизацію, обробляє передані дані, створює новий запис у базі даних і повертає результат на frontend [10; 13; 14].

База даних PostgreSQL використовується для збереження структурованої інформації. У ній зберігаються користувачі, книги, профільні дані, статуси, оцінки, нотатки та інформація про правовий доступ до цифрових матеріалів.

Такий підхід забезпечує чітке розділення логіки та дозволяє розвивати кожну частину системи окремо.

3.3 Діаграма варіантів використання

Діаграма варіантів використання відображає основні сценарії взаємодії користувачів із системою. У ShelfMate основними акторами є звичайний користувач і адміністратор [12; 25].

Звичайний користувач може виконувати такі дії:

- зареєструватися в системі;
- увійти до акаунта;
- переглянути власну бібліотеку;
- додати нову книгу;
- редагувати інформацію про книгу;
- видалити книгу;
- змінити статус читання;
- додати нотатку;
- оцінити книгу;
- переглянути детальну сторінку книги;
- редагувати профіль;
- переглянути читацький паспорт;
- переглянути рекомендації;
- відкрити легальне джерело або PDF, якщо доступ дозволений.

Адміністратор може виконувати додатково ще такі дії:

- увійти до системи;
- переглянути список користувачів;
- переглянути список книг;
- переглянути загальну статистику;
- змінити роль користувача;
- змінити правовий статус книги;
- обмежити доступ до сумнівного цифрового матеріалу.

Діаграма варіантів використання дозволяє наочно показати, які функції доступні кожній ролі, та визначити межі відповідальності користувача й адміністратора [12; 2].

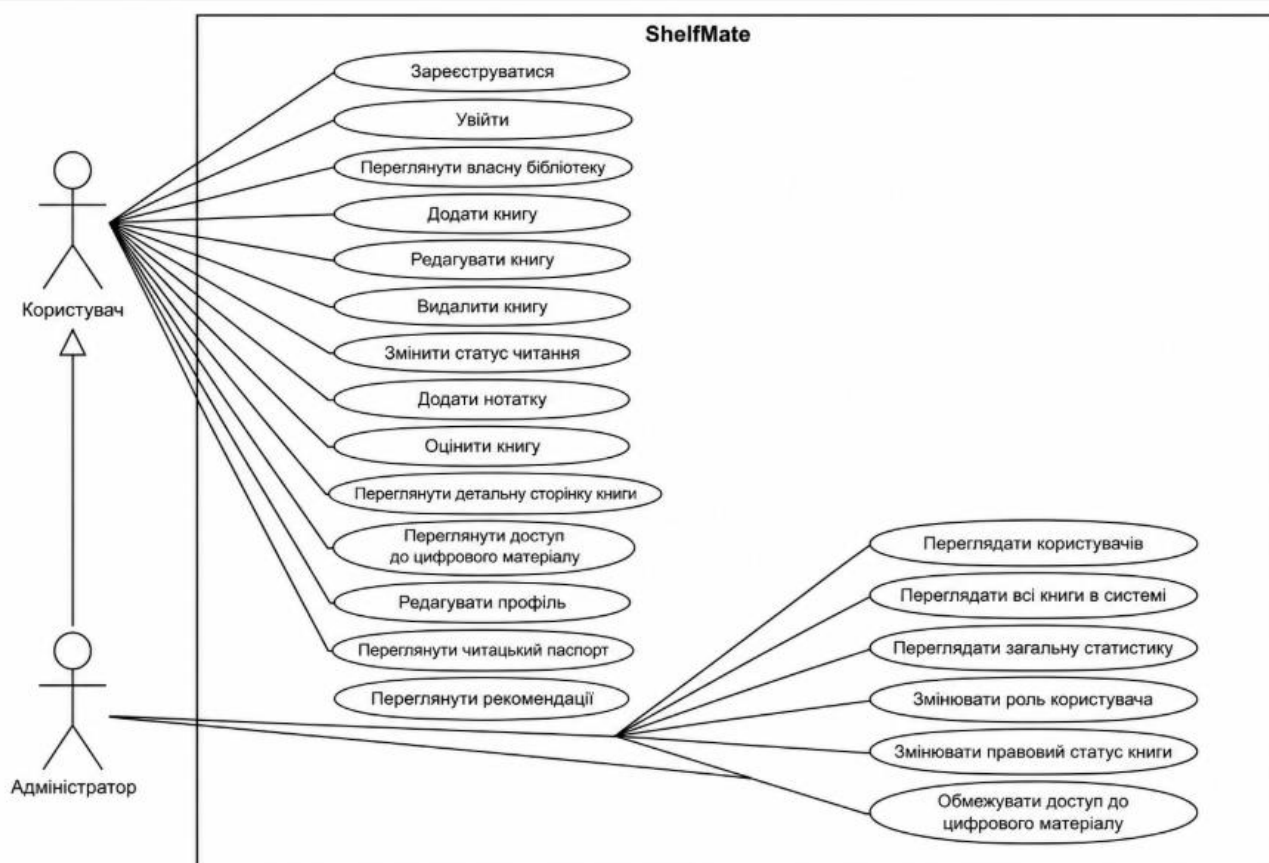


Рис. 3.2 Діаграма варіантів використання вебплатформи ShelfMate

3.4 Проєктування бази даних

База даних є важливою частиною вебплатформи ShelfMate, оскільки саме вона забезпечує збереження інформації про користувачів, книги, профілі, статуси читання та правовий доступ до цифрових матеріалів [14].

У поточній реалізації основними сутностями системи є користувач і книга. Користувач має власну бібліотеку, а кожна книга належить конкретному користувачу. Такий зв'язок дозволяє забезпечити приватність даних: один користувач не має доступу до книг іншого користувача.

Основна сутність users зберігає дані про користувача. До неї можуть входити такі поля: ідентифікатор, email, пароль, ім'я, роль, місто, аватар, коротка інформація про себе, улюблений жанр і читацька ціль на місяць.

Сутність books зберігає інформацію про книги. До неї можуть входити такі поля: ідентифікатор, назва, автор, жанр, опис, ISBN, видавництво, рік публікації, мова, статус читання, оцінка, нотатки, загальна кількість сторінок, поточна сторінка, посилання на обкладинку, правовий статус доступу та посилання на легальне джерело [2].

Зв'язок між users і books є зв'язком типу “один до багатьох”. Один користувач може мати багато книг, але кожна книга належить одному користувачу. Це є базовою структурою для персональної бібліотеки.

У подальшому розвитку системи структуру бази даних можна розширити. Наприклад, можна додати окремі таблиці для нотаток, цитат, жанрів, сеансів читання, файлів книг або спільних бібліотек. Проте для базової реалізації достатньо основних таблиць користувачів і книг [14].

Таблиця 3.1

Основні сутності бази даних ShelfMate

Сутність	Призначення	Основні поля
users	Збереження даних користувачів	id, email, password, name, role, city, avatarUrl, favoriteGenre, monthlyGoal
books	Збереження інформації про книги	id, title, author, genre, description, isbn, publisher, publicationYear, language, readingStatus, rating, notes, totalPages, currentPage, coverUrl, legalAccessStatus, legalSourceUrl, user_id

Така структура бази даних дозволяє реалізувати основні функції ShelfMate та залишає можливість для подальшого масштабування [14].

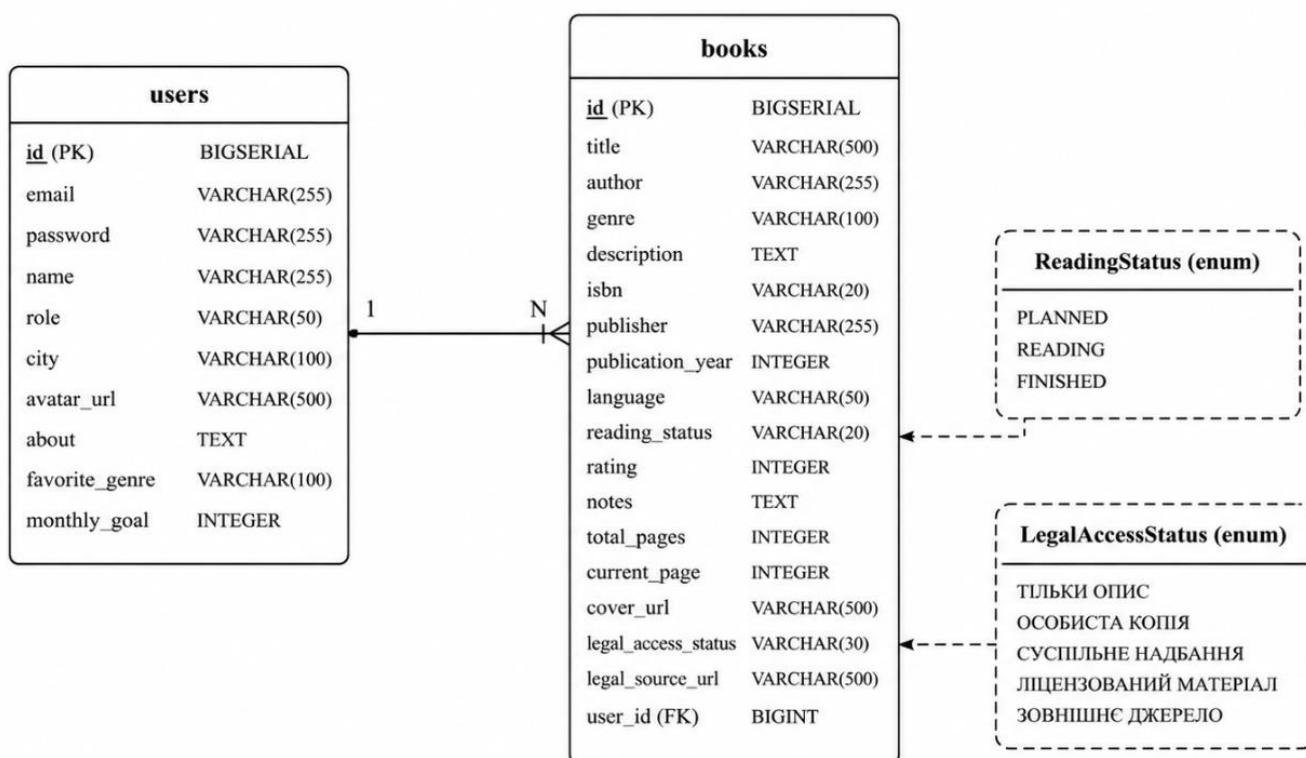


Рис. 3.3 Схеми бази даних вебплатформи ShelfMate

3.5 Проектування основних сутностей системи

Під час проектування серверної частини було визначено основні сутності, які відображають предметну область ShelfMate. Найважливішими з них є User та Book [10; 13].

Сутність User описує користувача системи. Вона містить дані, необхідні для авторизації, ідентифікації та персоналізації профілю. Кожен користувач має email, пароль, роль і додаткові профільні дані. Роль визначає, які функції доступні користувачу: звичайні функції бібліотеки або адміністративні можливості.

Сутність Book описує книгу, яка зберігається у персональній бібліотеці користувача. Вона містить бібліографічні дані, читацький статус, оцінку, нотатки, прогрес за сторінками, обкладинку та інформацію про правовий доступ. Саме ця сутність є центральною для роботи платформи, оскільки більшість функцій пов'язані з додаванням, редагуванням, пошуком і аналізом книг [10; 12; 25].

Окрім сутностей, у backend-частині проєктуються контролери, сервіси та репозиторії. Контролери приймають HTTP-запити від frontend-частини. Сервіси містять бізнес-логіку, наприклад перевірку прав доступу, формування статистики або рекомендацій. Репозиторії відповідають за взаємодію з базою даних.

Клас `AuthController` відповідає за реєстрацію та авторизацію користувачів. `BookController` обробляє запити, пов'язані з книгами. `AdminController` забезпечує доступ до адміністративних функцій. `AuthService` реалізує логіку входу, реєстрації та створення JWT-токена. `BookService` відповідає за роботу з книгами, а `JwtService` — за створення та перевірку токенів [10; 13].

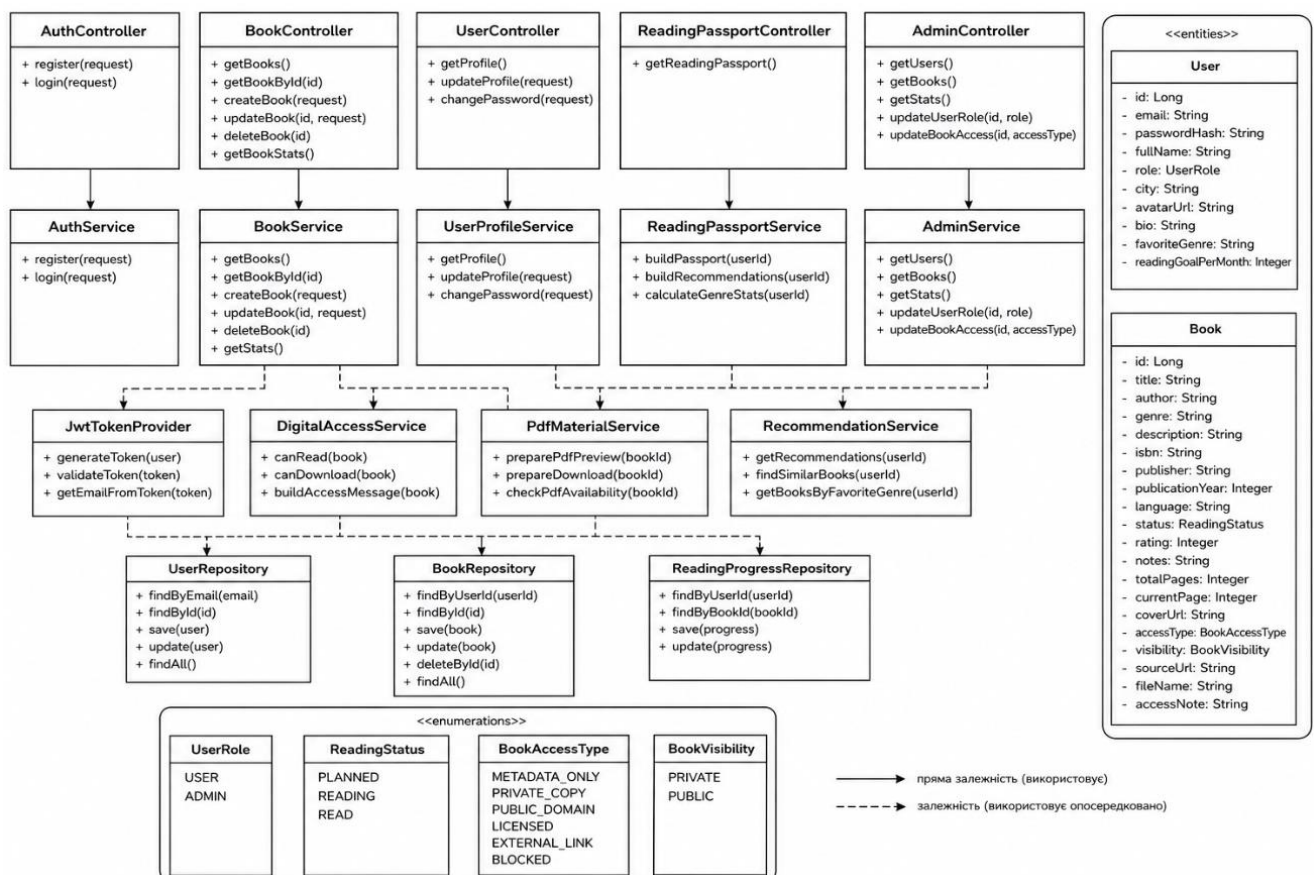


Рис. 3.4 Діаграма класів серверної частини ShelfMate

3.6 Проєктування процесу авторизації

Авторизація є одним із ключових процесів у ShelfMate, оскільки система працює з персональними даними користувачів. Без авторизації користувач не повинен мати доступ до бібліотеки, профілю, читацького паспорта або адміністративної панелі [12; 25].

Процес авторизації починається з того, що користувач вводить email і пароль на сторінці входу. Frontend надсилає ці дані на backend. Сервер перевіряє, чи існує користувач із таким email, і чи збігається пароль. Якщо дані правильні, backend формує JWT-токен і повертає його на клієнтську частину [25].

Frontend зберігає токен і додає його до наступних запитів. Коли користувач відкриває бібліотеку або виконує іншу захищену дію, backend перевіряє токен, визначає користувача та його роль. Якщо токен коректний, запит виконується. Якщо токен відсутній, неправильний або прострочений, система не надає доступ.

Загальна послідовність авторизації:

- користувач вводить email і пароль;
- frontend надсилає запит на backend;
- backend перевіряє облікові дані;
- backend створює JWT-токен;
- frontend зберігає токен;
- frontend надсилає токен при наступних запитах;
- backend перевіряє токен і роль користувача;
- система надає або блокує доступ.

Такий механізм дозволяє реалізувати безпечну роботу з персональними бібліотеками та адміністративними функціями.

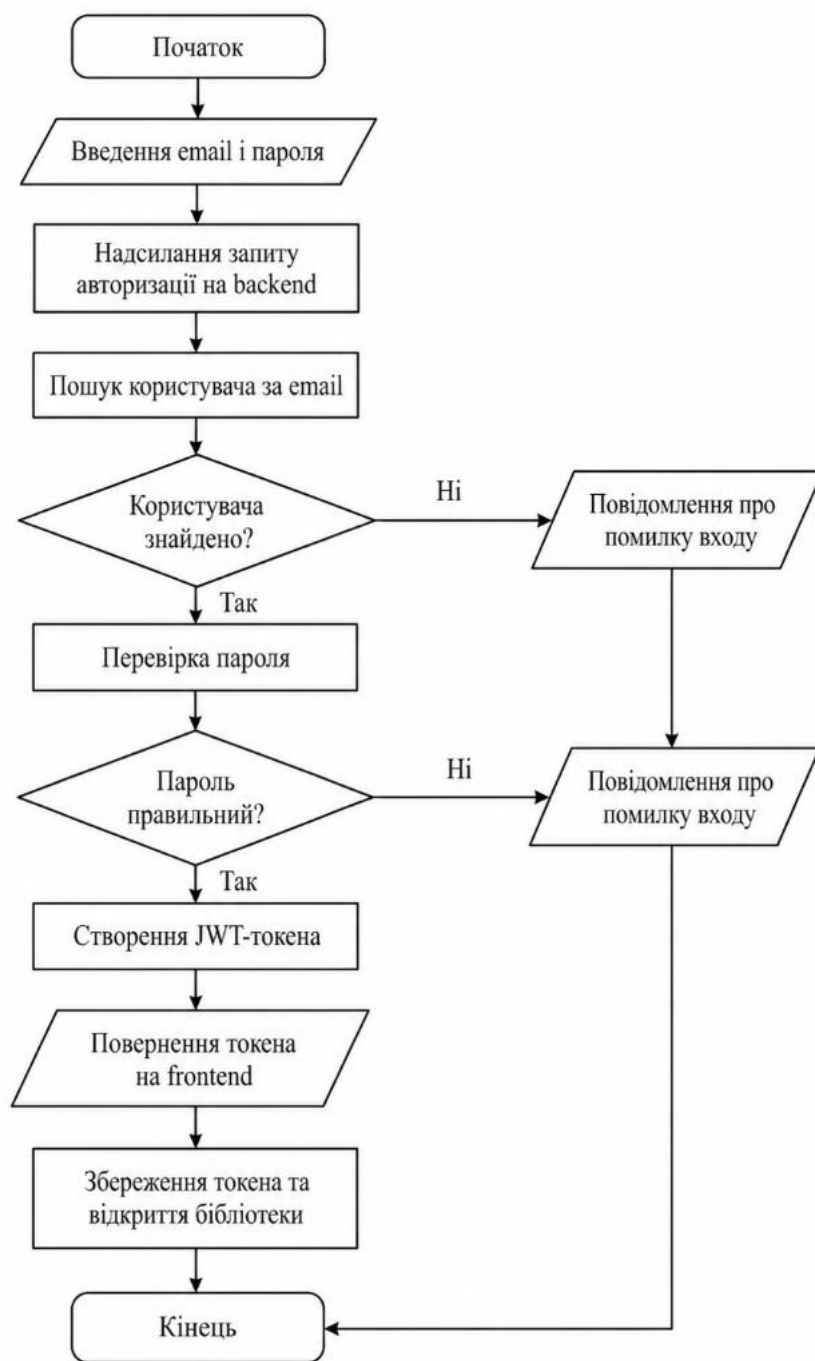


Рис. 3.5 Блок-схема процесу авторизації користувача

3.7 Проєктування адміністративної панелі

Адміністративна панель ShelfMate призначена для користувачів із роллю адміністратора. Її основна задача — забезпечити контроль платформи, перегляд загальної статистики та керування окремими даними [12].

У списку користувачів адміністратор може бачити основні дані акаунтів: ім'я, email, роль і кількість доданих книг. У списку книг адміністратор може переглядати книги, додані користувачами, а також перевіряти їхній правовий статус. Загальна статистика може містити кількість зареєстрованих користувачів, кількість книг, кількість прочитаних книг, кількість матеріалів з цифровим доступом і кількість книг, які потребують перевірки.

Керування ролями дозволяє адміністратору змінювати роль користувача, наприклад призначати адміністратора або повертати роль звичайного користувача. Ця функція повинна бути захищена, оскільки вона впливає на права доступу [2].

Контроль правового статусу цифрових матеріалів дозволяє адміністратору змінювати статус книги, якщо матеріал є сумнівним або не повинен бути доступний для завантаження. Наприклад, адміністратор може змінити статус книги на “Тільки опис”, якщо цифровий файл не має підтвердженого легального джерела.

Адміністративна панель демонструє не лише додаткову функціональність, а й важливий принцип сучасних вебзастосунків — розмежування прав доступу [12; 24].



Рис. 3.6 Структура адміністративної панелі ShelfMate

3.8 Діаграма послідовності додавання книги

Додавання книги є одним із базових сценаріїв роботи користувача у ShelfMate. Цей процес включає взаємодію між користувачем, frontend-частиною, backend-контролером, сервісом, репозиторієм і базою даних [18].

Спочатку користувач відкриває форму додавання книги та заповнює необхідні поля: назву, автора, жанр, статус читання, кількість сторінок, поточну сторінку, оцінку, нотатки, обкладинку та правовий статус. Після натискання кнопки збереження frontend перевіряє базову коректність даних і надсилає POST-запит на backend.

Backend приймає запит у BookController. Далі запит передається до BookService, де виконується бізнес-логіка: перевіряється авторизований користувач, обробляються введені дані, встановлюється зв'язок книги з власником. Після цього BookRepository зберігає книгу в PostgreSQL. Якщо операція успішна, backend повертає відповідь, а frontend оновлює список книг [10; 13; 14].

Загальна послідовність:

- користувач заповнює форму;
- frontend надсилає POST-запит;
- BookController приймає запит;
- BookService перевіряє дані та користувача;
- BookRepository зберігає книгу;
- PostgreSQL додає новий запис;
- backend повертає відповідь;
- frontend оновлює бібліотеку.

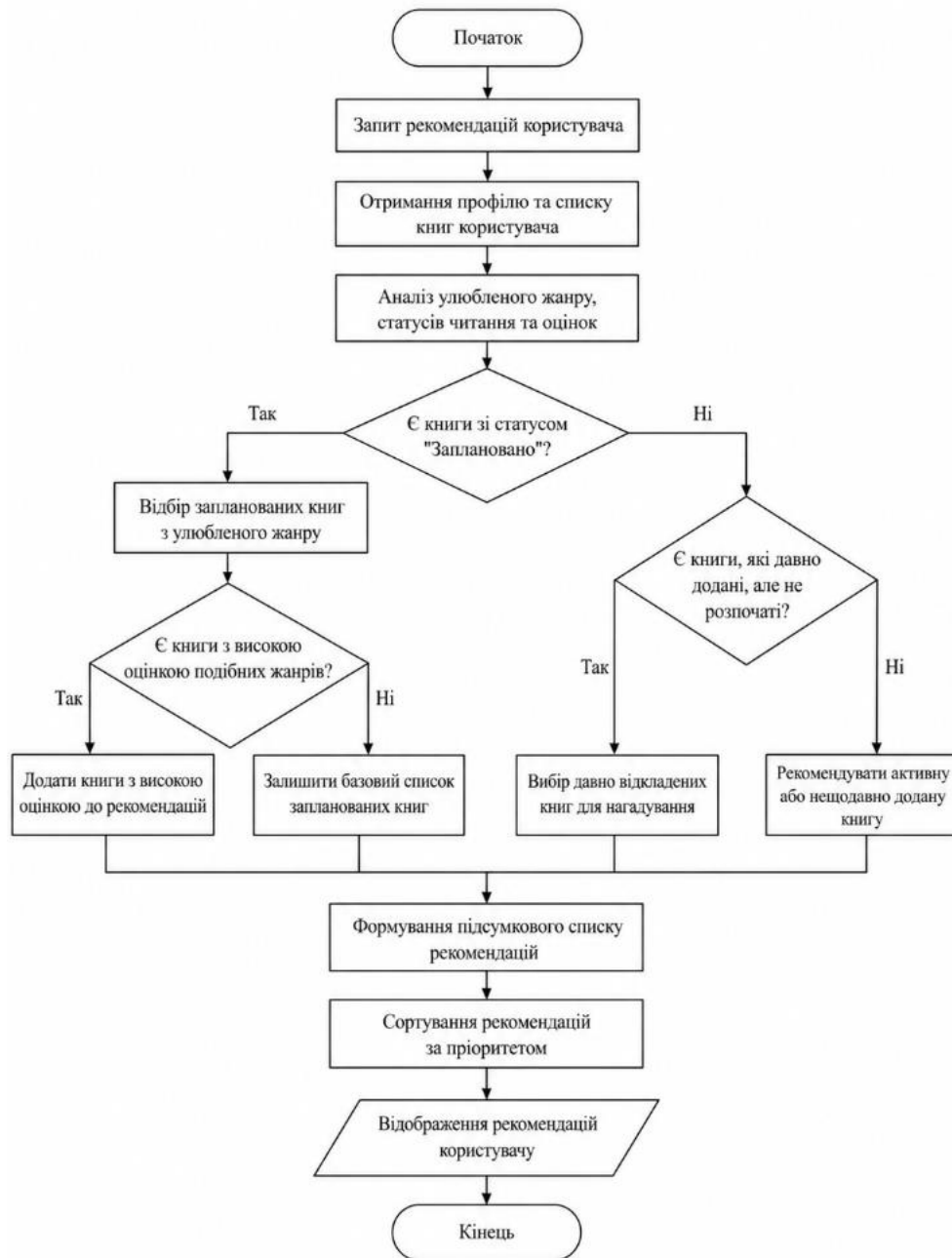


Рис. 3.7 Діаграма послідовності додавання книги до бібліотеки

3.9 Діаграма розгортання системи

Діаграма розгортання показує, як компоненти ShelfMate розміщуються у середовищі виконання. У базовому варіанті система складається з клієнтської частини, серверної частини та бази даних [16; 17].

Користувач працює з платформою через веббраузер. У браузері відкривається React-застосунок, який надсилає запити до backend-частини.

Backend-застосунок працює на Spring Boot і обробляє запити через REST API. Для збереження даних backend підключається до PostgreSQL.

Для запуску серверної частини та бази даних використовується Docker Compose. Це дозволяє створити окремий контейнер для PostgreSQL і окремий контейнер для backend. Frontend під час розробки може запускатися окремо через Vite, а у майбутньому також може бути розгорнутий як окремий контейнер або розміщений на вебсервері.

Основні вузли розгортання:

- клієнтський браузер;
- frontend-застосунок React;
- backend-застосунок Spring Boot;
- база даних PostgreSQL;
- Docker-середовище [16; 17].

Така структура є достатньою для демонстрації роботи платформи та може бути розширена в майбутньому, наприклад шляхом додавання окремого файлового сховища для обкладинок або PDF, сервісу сповіщень, хмарного розгортання або окремого модуля рекомендацій.

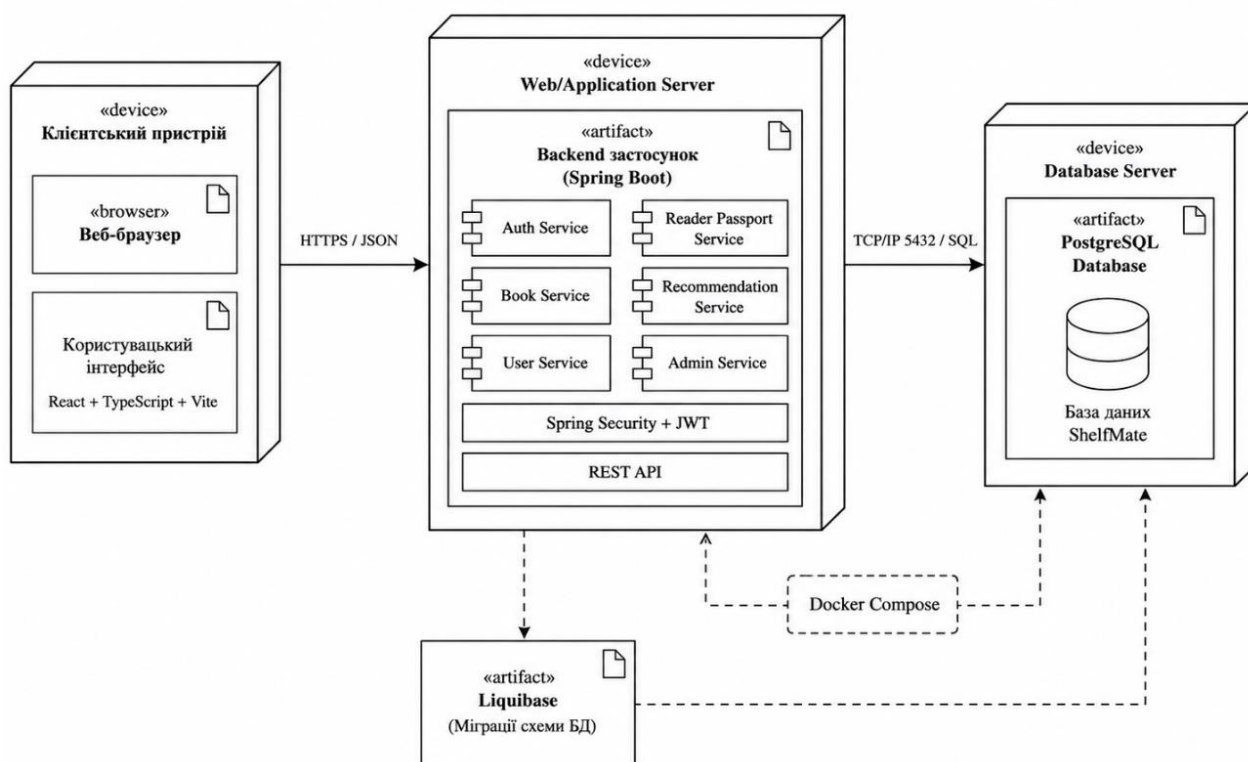


Рис. 3.8 Діаграма розгортання вебплатформи ShelfMate

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБПЛАТФОРМИ SHELFMATE

4.1 Загальна характеристика реалізації вебплатформи

Реалізація вебплатформи ShelfMate виконувалася відповідно до спроектованої клієнт-серверної архітектури. Система складається з клієнтської частини, серверної частини та бази даних. Клієнтська частина забезпечує взаємодію користувача із системою через вебінтерфейс, серверна частина обробляє запити, виконує бізнес-логіку та контролює доступ, а база даних зберігає інформацію про користувачів, книги, профільні дані та читацький прогрес [9; 10; 18; 14].

Основною метою реалізації було створення зручної платформи для ведення домашньої бібліотеки, у якій користувач може додавати книги, змінювати статус читання, вести нотатки, оцінювати прочитане, переглядати статистику та працювати з персональним читацьким паспортом. Окремо було реалізовано адміністративну частину, яка дозволяє контролювати користувачів, книги та правовий статус цифрових матеріалів.

Backend-частина реалізована за допомогою Java та Spring Boot. Вона містить контролери, сервіси, репозиторії, сутності та конфігурацію безпеки. Frontend-частина реалізована за допомогою React, TypeScript, Vite та Tailwind CSS. Для збереження даних використовується PostgreSQL, а для керування структурою бази даних — Liquibase.

4.2 Реалізація серверної частини

Серверна частина ShelfMate реалізована на основі Spring Boot. Вона відповідає за обробку HTTP-запитів, авторизацію користувачів, взаємодію з базою даних, перевірку ролей, формування відповідей для клієнтської частини та виконання бізнес-логіки [10].

Контролери приймають запити від frontend-частини та передають їх до відповідних сервісів. Наприклад, AuthController відповідає за реєстрацію та авторизацію, BookController — за роботу з книгами, UserController — за профіль користувача, а AdminController — за адміністративні функції.

Сервіси містять основну бізнес-логіку системи. Наприклад, BookService відповідає за створення, редагування, видалення та пошук книг. AuthService виконує перевірку облікових даних користувача та створення JWT-токена. AdminService забезпечує роботу з адміністративними функціями, зокрема переглядом користувачів, книг і зміною правових статусів [13].

Репозиторії забезпечують взаємодію з базою даних. Вони дозволяють отримувати, зберігати, оновлювати та видаляти записи без написання великої кількості SQL-коду. Для цього використовується Spring Data JPA.

Основними сутностями серверної частини є User та Book. Сутність User зберігає дані користувача, роль і профільну інформацію. Сутність Book містить інформацію про книгу, її статус читання, оцінку, нотатки, прогрес і правовий статус доступу [13].

4.3 Реалізація авторизації та розмежування доступу

Для захисту персональних даних у ShelfMate реалізовано механізм авторизації з використанням JWT-токена. Користувач отримує доступ до власної бібліотеки лише після успішного входу в систему. Без авторизації користувач може перейти тільки на сторінку входу або реєстрації [12; 25].

Процес авторизації починається з введення email і пароля. Після цього frontend надсилає запит на backend. Сервер перевіряє, чи існує користувач із таким email, і чи правильний пароль. Якщо перевірка проходить успішно, backend створює JWT-токен і повертає його клієнтській частині [25].

Frontend зберігає отриманий токен і додає його до наступних запитів. Кожного разу, коли користувач звертається до захищеного ресурсу, backend

перевіряє токен і визначає користувача. Якщо токен коректний, запит виконується. Якщо токен відсутній або неправильний, система блокує доступ.

У системі передбачено дві ролі:

- USER — звичайний користувач;
- ADMIN — адміністратор [12].

Користувач із роллю USER має доступ тільки до власної бібліотеки, профілю, читацького паспорта та рекомендацій. Користувач із роллю ADMIN має доступ до адміністративної панелі, списку користувачів, загальної статистики та функцій контролю правового статусу книг.

Розмежування доступу реалізовано на backend-рівні. Це означає, що навіть у разі прямого звернення до адміністративного API система перевіряє роль користувача і не дозволяє виконати дію без відповідних прав [12; 24].

4.4 Реалізація клієнтської частини

Клієнтська частина ShelfMate реалізована за допомогою React. Вона відповідає за відображення сторінок, форм, карток книг, фільтрів, статистики, профілю, читацького паспорта та адміністративної панелі [18].

Інтерфейс побудований за компонентним принципом. Кожен функціональний елемент системи реалізований як окремий компонент. Наприклад, картка книги, форма додавання книги, блок статистики, фільтри, профіль користувача та елементи адміністративної панелі можуть бути окремими компонентами. Це спрощує підтримку коду та дозволяє повторно використовувати елементи інтерфейсу [18].

Основними сторінками клієнтської частини є:

- сторінка входу;
- сторінка реєстрації;
- головна сторінка бібліотеки;
- сторінка детальної інформації про книгу;

- сторінка профілю;
- сторінка читацького паспорта;
- адміністративна панель.

Сторінка входу дозволяє користувачу ввести email і пароль. Сторінка реєстрації призначена для створення нового акаунта. Після входу користувач переходить на головну сторінку бібліотеки, де може працювати з власними книгами.

Інтерфейс платформи зроблено простим і зрозумілим. Основний акцент зроблено на картках книг, пошуку, фільтрах і швидкому доступі до основних дій. Це дозволяє користувачу швидко додати книгу, знайти потрібний запис або змінити статус читання [18].

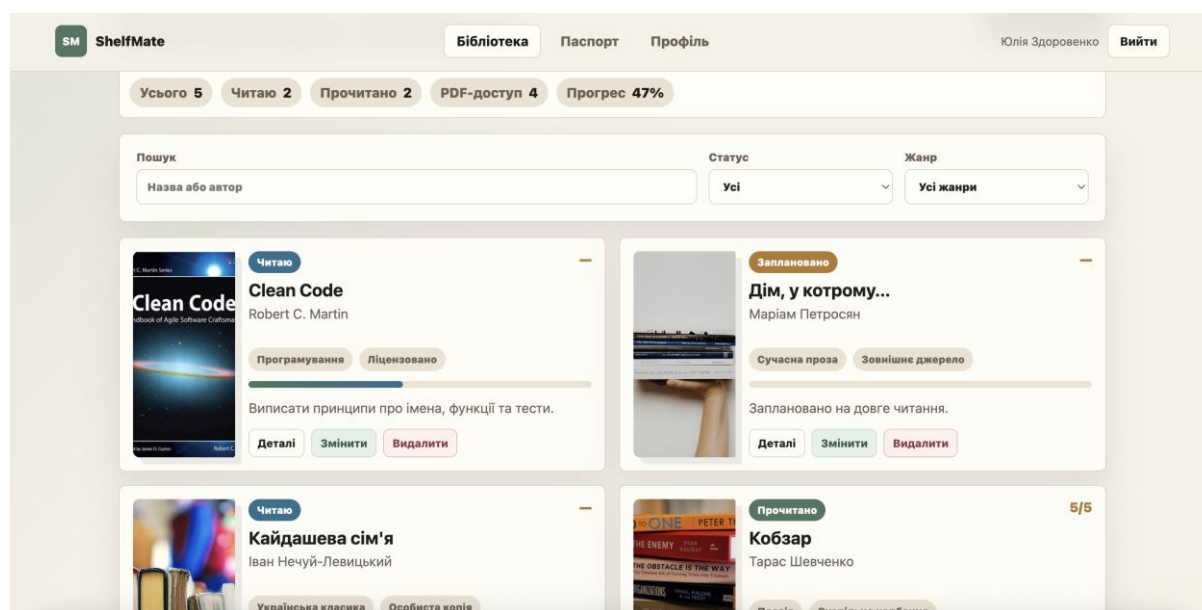


Рис. 4.1 Головна сторінка бібліотеки користувача ShelfMate

4.5 Реалізація сторінки входу та реєстрації

Сторінка входу є початковою точкою взаємодії користувача із системою. На ній користувач вводить email і пароль. Після натискання кнопки входу frontend надсилає запит до backend-частини. Якщо дані правильні, користувач отримує JWT-токен і переходить до власної бібліотеки [12; 25].

Сторінка реєстрації дозволяє створити новий акаунт. Користувач вводить необхідні дані, після чого система створює новий запис у базі даних. За замовчуванням новому користувачу надається роль USER. Це означає, що він може працювати тільки зі своєю бібліотекою та не має доступу до адміністративних функцій [12].

Для демонстрації роботи системи можуть бути передбачені тестові акаунти: акаунт адміністратора, акаунт звичайного користувача та додатковий акаунт читача. Це дозволяє показати різницю між ролями та перевірити, що кожен користувач бачить лише дозволені йому дані [25; 12].

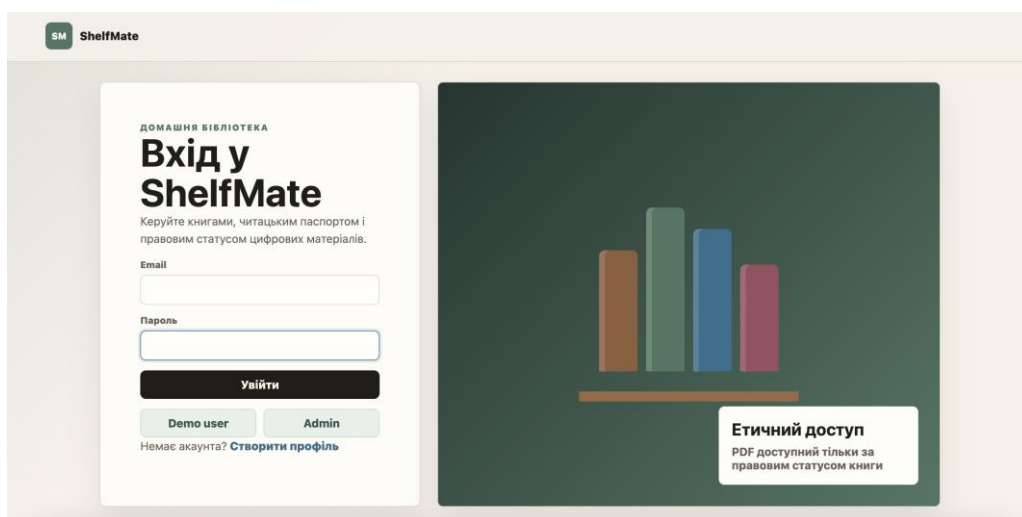


Рис. 4.2 Сторінка входу до вебплатформи ShelfMate

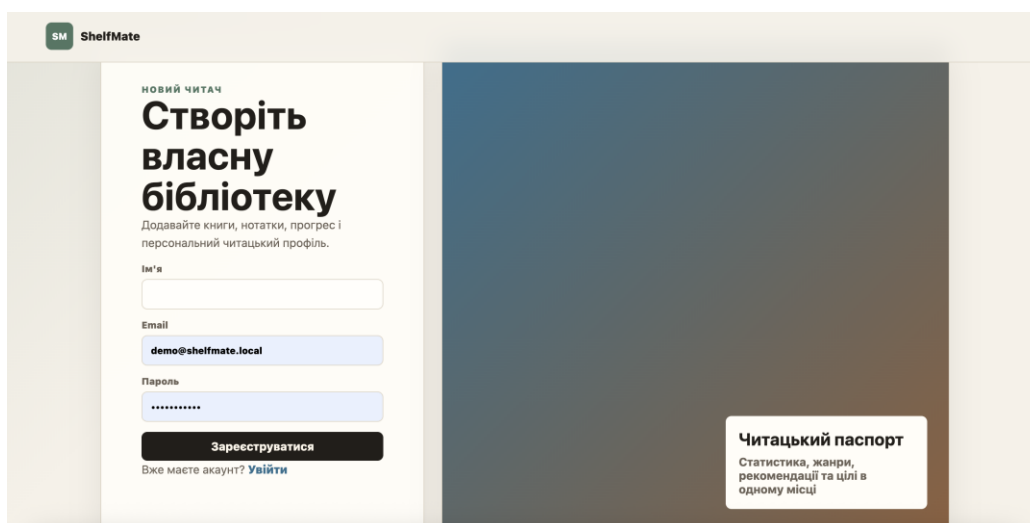


Рис. 4.3 Сторінка реєстрації користувача у ShelfMate

4.6 Реалізація головної сторінки бібліотеки

Головна сторінка бібліотеки є основним робочим простором користувача. Саме на цій сторінці відображається список книг, які користувач додав до власної бібліотеки. Кожна книга представлена у вигляді картки, що дозволяє швидко переглянути основну інформацію [18].

Головна сторінка також містить інструменти пошуку та фільтрації. Користувач може знайти книгу за назвою або автором, а також відфільтрувати записи за статусом читання або жанром. Це особливо важливо для користувачів, які мають велику кількість книг.

На сторінці бібліотеки також може відображатися коротка статистика: загальна кількість книг, кількість прочитаних книг, кількість книг у процесі читання та кількість запланованих книг. Така статистика допомагає швидко оцінити поточний стан бібліотеки без переходу до читацького паспорта.

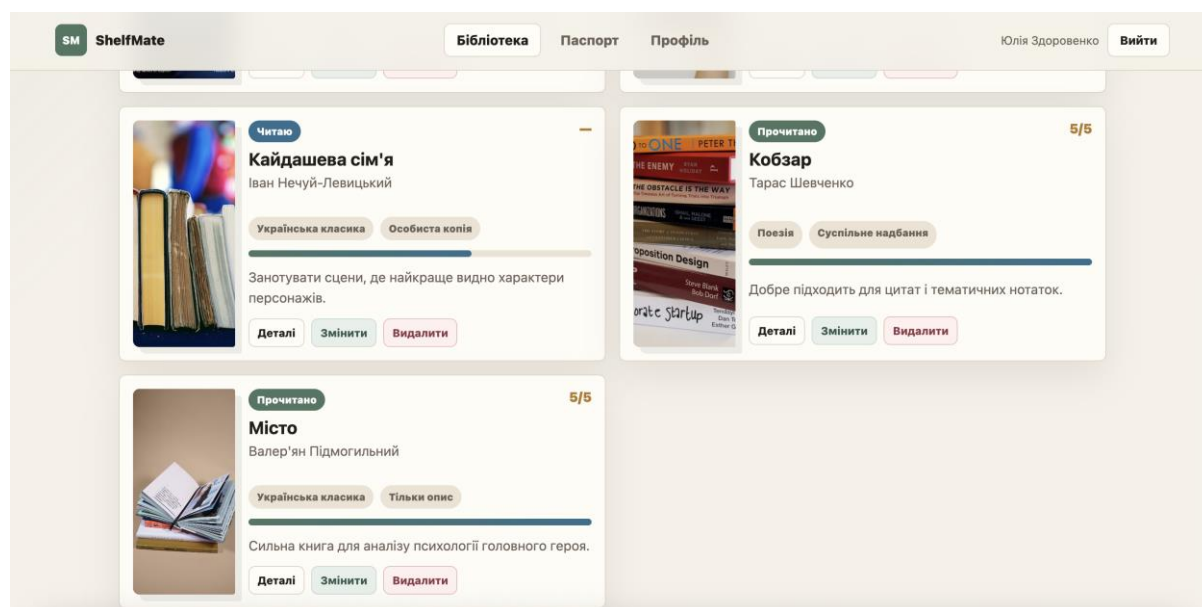


Рис. 4.4 Відображення карток книг на головній сторінці бібліотеки

4.7 Реалізація додавання та редагування книги

Додавання книги є однією з основних функцій ShelfMate. Користувач може створити новий запис і заповнити інформацію про книгу. Для цього використовується форма, у якій передбачено поля для введення основних бібліографічних і користувацьких даних [13; 14].

Після заповнення форми frontend надсилає дані на backend. Сервер перевіряє авторизованого користувача, обробляє дані та зберігає книгу в базі даних. Книга автоматично прив'язується до поточного користувача, що забезпечує приватність бібліотеки [13; 14].

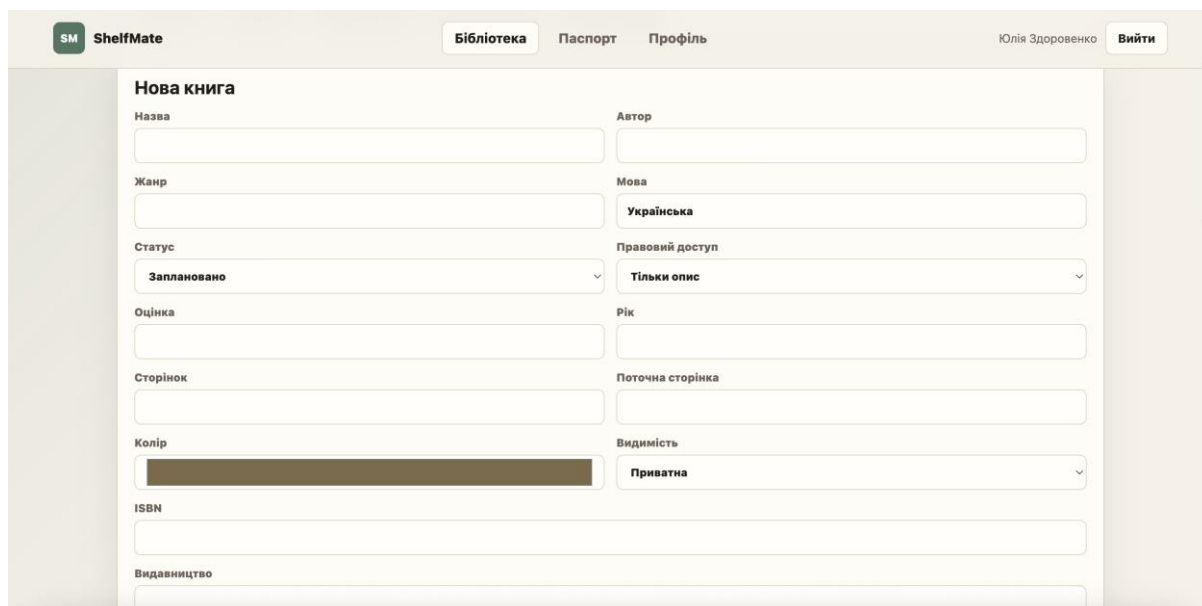


Рис. 4.5 Форма додавання нової книги до бібліотеки

4.8 Реалізація детальної сторінки книги

Детальна сторінка книги призначена для перегляду повної інформації про вибране видання. Якщо на головній сторінці користувач бачить лише скорочену картку, то на детальній сторінці відображаються всі дані, пов'язані з книгою [18; 13].

Також на сторінці передбачені кнопки для дій із книгою. Користувач може редагувати запис, видалити книгу, перейти до легального джерела, прочитати PDF або завантажити PDF, якщо це дозволено правовим статусом [1; 6].

Особливістю цієї сторінки є те, що доступність кнопок читання та завантаження залежить від правового статусу книги. Якщо книга має статус “Тільки опис”, відповідні дії будуть неактивними. Якщо вказано зовнішнє легальне джерело, користувач може перейти за посиланням. Якщо книга має статус “Особиста копія”, доступ надається лише власнику бібліотеки.

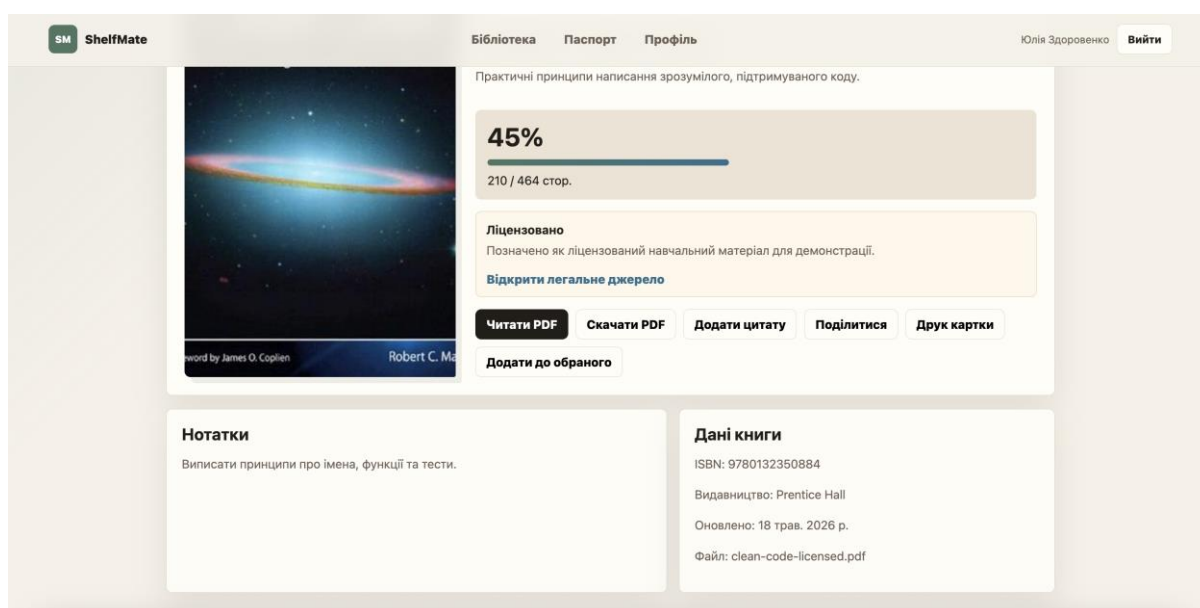


Рис. 4.6 Детальна сторінка книги у вебплатформі ShelfMate

4.9 Реалізація профілю користувача

Сторінка профілю призначена для персоналізації платформи. У профілі користувач може переглядати та редагувати власні дані, які використовуються для відображення інформації в системі та формування читацької статистики [13].

Також у профілі може відображатися коротка статистика: загальна кількість книг, кількість прочитаних книг, кількість активних книг, середня оцінка та прогрес до місячної цілі.

Профіль дозволяє зробити платформу більш персоналізованою. Наприклад, улюблений жанр може використовуватися для формування рекомендацій, а читацька ціль — для розрахунку прогресу в читацькому паспорті [13; 14].

Редагування профілю реалізовано через форму, у якій користувач може змінити доступні поля. Після збереження frontend надсилає оновлені дані на backend, а сервер записує зміни в базу даних.

Рис. 4.7 Сторінка профілю користувача ShelfMate

4.10 Реалізація читацького паспорта

Читацький паспорт є однією з основних особливостей ShelfMate. Він формує персональний читацький портрет користувача на основі інформації з його бібліотеки [13; 14].

Під час відкриття читацького паспорта система отримує всі книги поточного користувача та розраховує основні показники. До них належать загальна кількість книг, кількість прочитаних, кількість активних книг, кількість запланованих книг, середня оцінка, улюблений жанр, прогрес за сторінками та виконання читацької цілі.

Читацький паспорт дозволяє користувачу побачити не просто список книг, а загальну картину власної читацької активності. Це робить платформу більш аналітичною та корисною для довгострокового використання [13].

Формування читацького паспорта реалізовано на основі даних, які вже зберігаються у системі. Це дозволяє не вводити додаткові дані вручну, а автоматично формувати аналітику після оновлення бібліотеки [13].

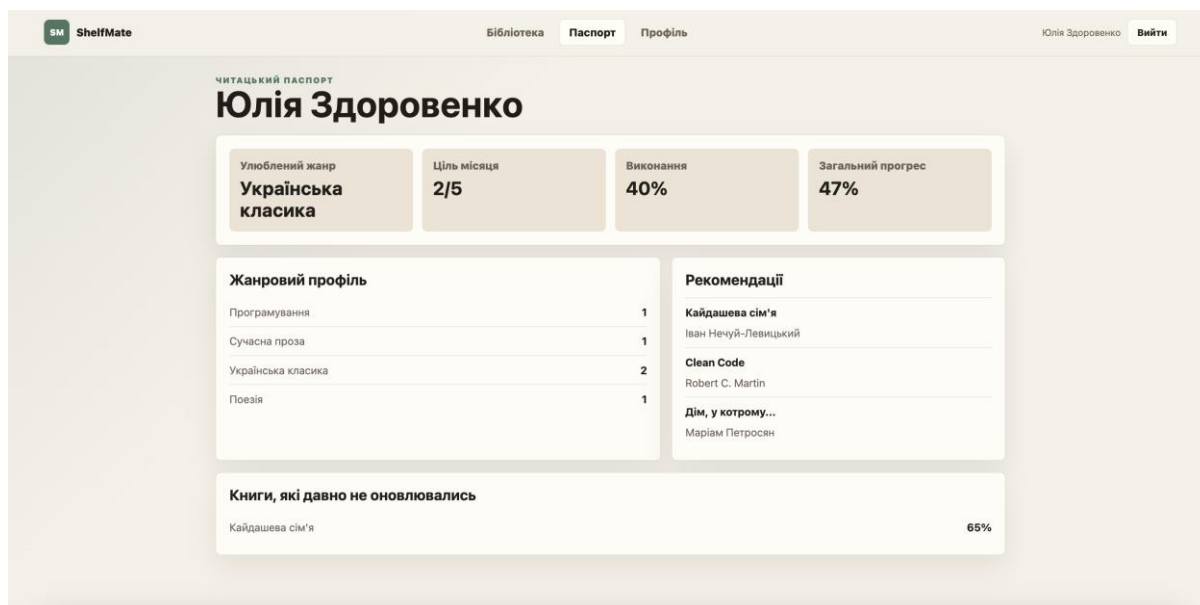


Рис. 4.8 Сторінка читацького паспорта користувача

4.11 Реалізація контролю правового статусу цифрових матеріалів

Однією з особливостей ShelfMate є контроль правового статусу цифрових матеріалів. У системі передбачено, що книга може бути додана до бібліотеки як інформаційний запис, але це не означає, що її PDF-файл можна вільно читати або завантажувати [1; 6].

Для кожної книги може бути встановлений один із таких статусів:

- тільки опис;
- особиста копія;
- суспільне надбання;

- ліцензований матеріал;
- зовнішнє легальне джерело [1; 6].

Якщо книга має статус “Тільки опис”, система відображає інформацію про книгу, але блокує кнопки читання та завантаження PDF. Якщо встановлено статус “Особиста копія”, доступ може бути наданий тільки власнику бібліотеки. Якщо книга належить до суспільного надбання або є ліцензованою, система може дозволити читання або завантаження. Якщо вказано зовнішнє легальне джерело, користувач може перейти за відповідним посиланням [1; 6].

Цей механізм дозволяє враховувати авторське право та робить платформу більш відповідальною. ShelfMate не позиціонується як незаконне сховище електронних книг, а працює як система організації бібліотеки з контрольованим доступом до цифрових матеріалів.

4.12 Реалізація адміністративної панелі

Адміністративна панель доступна тільки користувачам із роллю ADMIN. Вона призначена для контролю системи, перегляду загальної статистики, роботи з користувачами та керування правовими статусами книг [12].

Адміністративна панель містить такі блоки:

- список користувачів;
- список книг;
- загальна статистика;
- керування ролями;
- керування правовими статусами цифрових матеріалів.

У списку користувачів адміністратор може переглядати зареєстрованих користувачів, їхні ролі та основну інформацію. У списку книг адміністратор може бачити книги, додані користувачами, та перевіряти їхній правовий статус.

Загальна статистика дозволяє оцінити роботу платформи: кількість користувачів, кількість книг, кількість прочитаних книг, кількість цифрових матеріалів і кількість записів, які потребують перевірки [13; 14].

Керування ролями дозволяє адміністратору змінювати роль користувача. Керування правовими статусами дозволяє змінити доступ до цифрового матеріалу, якщо книга не має підтвердженого легального джерела або повинна бути доступна лише як опис [1; 6].

Адміністративна панель демонструє важливий принцип розмежування прав доступу. Звичайний користувач не має доступу до цих функцій, а всі адміністративні запити перевіряються на backend-рівні [12; 24].

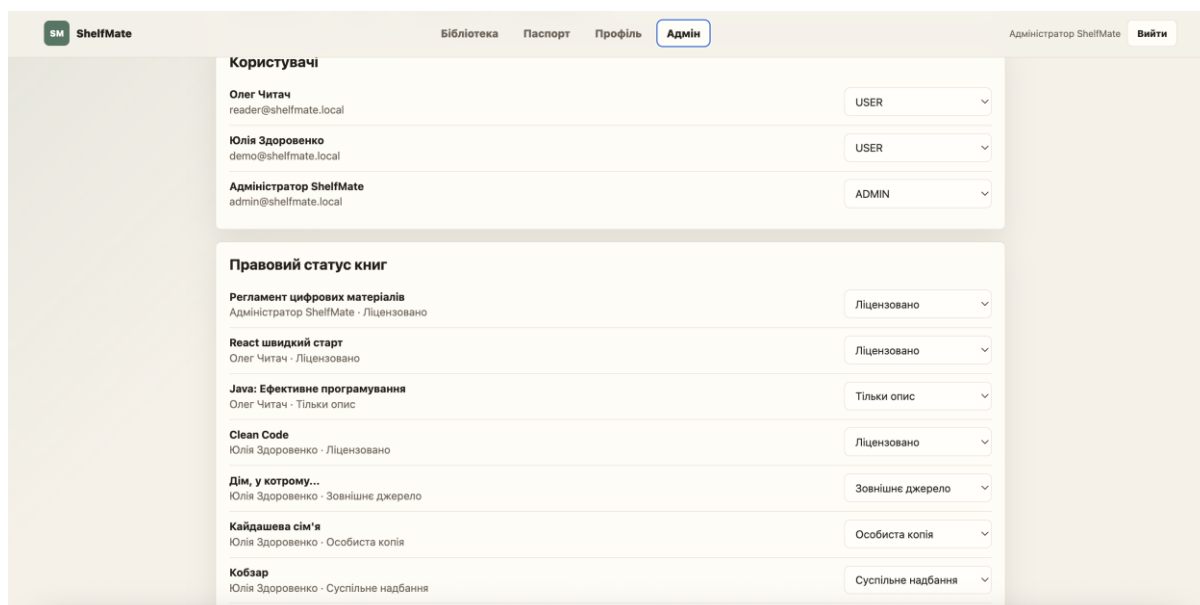


Рис. 4.9 Адміністративна панель вебплатформи ShelfMate

4.13 Тестування вебплатформи

Після реалізації основних функцій було проведено тестування вебплатформи ShelfMate. Метою тестування було перевірити працездатність основних сценаріїв, правильність збереження даних, коректність розмежування доступу та стабільність взаємодії між frontend-, backend-частиною і базою даних [13; 14].

Тестування проводилося вручну через вебінтерфейс, а також шляхом перевірки API-запитів. Для перевірки backend-частини використовувалися запити до REST API, що дозволило переконатися у правильності відповідей сервера.

Особлива увага приділялася перевірці приватності даних. Потрібно було переконатися, що користувач бачить тільки власні книги та не може отримати доступ до книг іншого користувача. Також перевірялося, що адміністративна панель доступна лише користувачу з роллю ADMIN [2; 12; 24].

4.14 Результати функціонального тестування

Результати функціонального тестування наведено в таблиці 4.1.

Таблиця 4.1

Результати функціонального тестування вебплатформи ShelfMate

№	Тестовий сценарій	Очікуваний результат	Фактичний результат	Статус
1	Реєстрація нового користувача	Користувач створюється в системі	Акаунт успішно створено	Успішно
2	Авторизація користувача	Користувач входить до системи	Вхід виконано, токен отримано	Успішно
3	Вхід із неправильним паролем	Система показує помилку	Доступ заблоковано	Успішно
4	Перегляд власної бібліотеки	Відображаються книги поточного користувача	Бібліотека відображена коректно	Успішно
5	Додавання нової книги	Книга зберігається в базі даних	Книга додана та показана у списку	Успішно
6	Редагування книги	Дані книги оновлюються	Зміни збережено	Успішно
7	Видалення книги	Книга видаляється з бібліотеки	Запис видалено	Успішно
8	Пошук за назвою	Відображаються відповідні книги	Пошук працює коректно	Успішно

Продовження таблиці 4.1

Результати функціонального тестування вебплатформи ShelfMate

№	Тестовий сценарій	Очікуваний результат	Фактичний результат	Статус
9	Фільтрація за статусом	Показуються книги з обраним статусом	Фільтр працює коректно	Успішно
10	Зміна статусу читання	Статус книги оновлюється	Статус змінено	Успішно
11	Додавання нотатки	Нотатка зберігається до книги	Нотатку збережено	Успішно
12	Додавання оцінки	Оцінка відображається у книзі	Оцінку збережено	Успішно
13	Перегляд профілю	Дані користувача відображаються	Профіль відкрито	Успішно
14	Редагування профілю	Дані профілю оновлюються	Зміни збережено	Успішно
15	Перегляд читацького паспорта	Статистика формується автоматично	Дані паспорта відображено	Успішно
16	Формування рекомендацій	Система пропонує книги з бібліотеки	Рекомендації відображено	Успішно
17	Доступ USER до адмінпанелі	Доступ має бути заборонений	Доступ заблоковано	Успішно
18	Доступ ADMIN до адмінпанелі	Панель відкривається	Адмінпанель доступна	Успішно
19	Зміна правового статусу книги	Статус оновлюється адміністратором	Статус змінено	Успішно
20	Книга зі статусом “Тільки опис”	PDF недоступний	Кнопки читання/завантаження неактивні	Успішно

Проведене тестування показало, що основні функції вебплатформи працюють коректно. Система дозволяє реєструвати та авторизувати користувачів, працювати з книгами, переглядати читацький паспорт, формувати рекомендації та контролювати доступ до адміністративних функцій.

4.15 Розгортання та запуск системи

Для запуску вебплатформи ShelfMate використовується поєднання Docker Compose, backend-застосунку Spring Boot, бази даних PostgreSQL і frontend-застосунку React [16; 17; 10; 14; 18].

Docker Compose дозволяє запустити базу даних і серверну частину в окремих контейнерах. Це спрощує налаштування середовища, оскільки розробнику не потрібно вручну встановлювати PostgreSQL і налаштовувати підключення. Усі основні параметри можна описати в конфігураційному файлі [16; 17].

Frontend-частина запускається через Vite. Після запуску вона відкривається у браузері та надсилає запити до backend-частини через REST API [19].

Загальний порядок запуску системи:

- запуск Docker Compose для PostgreSQL і backend;
- перевірка підключення backend до бази даних;
- виконання міграцій Liquibase;
- запуск frontend-частини через Vite;
- відкриття вебзастосунку у браузері;
- авторизація тестового користувача;
- перевірка основних функцій системи.

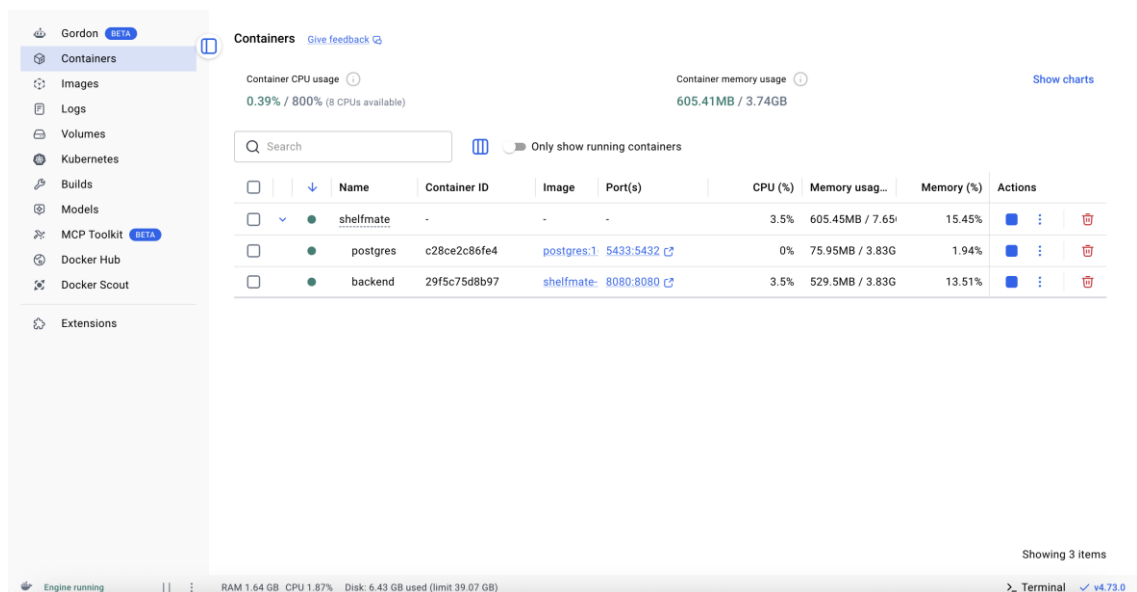


Рис. 4.10 Запуск застосунку у Docker

ВИСНОВКИ

У кваліфікаційній роботі було розроблено вебплатформу **ShelfMate** для ведення домашньої бібліотеки, організації інформації про книги, контролю читацького прогресу та формування персональної статистики користувача.

1. Було проаналізовано предметну галузь ведення домашньої бібліотеки та визначено основні проблеми користувачів, зокрема неструктуроване зберігання інформації про книги, складність відстеження читацького прогресу, розрізненість нотаток та відсутність єдиного інструменту для персонального аналізу читання [26; 27; 28; 29].
2. Було досліджено існуючі програмні засоби для організації книжкових колекцій, зокрема електронні таблиці, Goodreads, The StoryGraph, LibraryThing та Libib. У результаті аналізу встановлено, що наявні рішення частково вирішують задачі обліку книг, однак не забезпечують повного поєднання домашнього каталогу, читацького паспорта, рекомендацій, адміністративного контролю та перевірки правового статусу цифрових матеріалів [26; 27; 28; 29].
3. Було сформовано функціональні та нефункціональні вимоги до вебплатформи ShelfMate. До основних функцій системи віднесено реєстрацію та авторизацію користувачів, додавання, редагування й видалення книг, пошук і фільтрацію, зміну статусу читання, ведення нотаток, оцінювання книг, перегляд профілю, формування читацького паспорта, рекомендації та роботу адміністративної панелі [2; 12; 24].
4. Було спроектовано архітектуру вебплатформи ShelfMate. Система побудована за клієнт-серверним принципом: клієнтська частина реалізує інтерфейс користувача, серверна частина відповідає за бізнес-логіку та обробку запитів, а база даних забезпечує збереження інформації про користувачів і книги. Також було визначено основні ролі системи — звичайний користувач та адміністратор [10; 18; 14].

5. Було розроблено структуру бази даних, основні сутності системи та логіку взаємодії між ними. Основними сутностями стали користувач і книга, між якими реалізовано зв'язок “один до багатьох”. Такий підхід дозволяє кожному користувачу працювати лише з власною бібліотекою та забезпечує приватність персональних даних [14].
6. Було реалізовано серверну частину вебплатформи з використанням Java, Spring Boot, Spring Security, JWT, Spring Data JPA, PostgreSQL та Liquibase. Backend забезпечує авторизацію, розмежування доступу, роботу з книгами, профілем [9; 10; 12; 13; 14; 15; 25].
7. Було реалізовано клієнтську частину вебплатформи з використанням React, TypeScript, Vite та Tailwind CSS. Користувацький інтерфейс включає сторінки входу та реєстрації, головну сторінку бібліотеки, детальну сторінку книги, профіль користувача, читацький паспорт і адміністративну панель [18; 19; 20; 21].
8. Окремо було реалізовано механізм контролю правового статусу цифрових матеріалів. Для кожної книги може бути визначено статус доступу: тільки опис, особиста копія, суспільне надбання, ліцензований матеріал або зовнішнє легальне джерело [1; 6].
9. Було проведено функціональне тестування основних сценаріїв роботи вебплатформи. Перевірено реєстрацію, авторизацію, додавання, редагування та видалення книг, пошук, фільтрацію, зміну статусу читання, роботу профілю, читацького паспорта, рекомендацій, адміністративної панелі та обмеження доступу для різних ролей користувачів.

У результаті виконання роботи було створено вебплатформу ShelfMate, яка поєднує функції електронного каталогу домашньої бібліотеки, читацького щоденника, персональної аналітики та відповідального керування цифровими матеріалами. Розроблена система відповідає поставленій меті та може бути розширена в майбутньому шляхом додавання PDF-reader, цитат, календаря читання, імпорту книг з відкритих бібліотек, спільних книжкових полиць і розширених рекомендацій [16; 17].

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей;

1. Здоровенко Ю.С., Корецька В.О. Розробка web-застосунку для відстеження процесу тренувань. II Всеукраїнська науково-технічна конференція "Виклики та рішення в програмній інженерії", 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025.С.459-461
2. Здоровенко Ю.С., Корецька В.О. Особливості ведення домашньої бібліотеки та трекінгу прочитаного матеріалу за допомогою веб-застосунку Shelfmate Всеукраїнська науково-технічна конференція «Технології цифрового розвитку: програмні системи та децентралізація», 1-3 травня 2026 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. (подано до друку).

ПЕРЕЛІК ПОСИЛАНЬ

1. Верховна Рада України. “Про авторське право і суміжні права”. URL: <https://zakon.rada.gov.ua/laws/show/2811-20#Text> (дата звернення 10.05.2026).
2. Верховна Рада України. “Про захист персональних даних”. URL: <https://zakon.rada.gov.ua/laws/show/2297-17#Text> (дата звернення 10.05.2026).
3. Міністерство цифрової трансформації України. “Дія.Освіта”. URL: <https://osvita.diia.gov.ua/> (дата звернення 10.05.2026).
4. Дія.Освіта. “Кібергігієна та онлайн-безпека”. URL: <https://osvita.diia.gov.ua/> (дата звернення 10.05.2026).
5. Державна служба спеціального зв'язку та захисту інформації України. “Кіберзахист”. URL: <https://cip.gov.ua/ua> (дата звернення 10.05.2026).
6. Український інститут інтелектуальної власності. “Авторське право”. URL: <https://ukrpatent.org/uk> (дата звернення 10.05.2026).
7. Державна науково-технічна бібліотека України. “Електронні інформаційні ресурси”. URL: <https://dntb.gov.ua/> (дата звернення 10.05.2026).
8. Національна бібліотека України імені В. І. Вернадського. “Електронні ресурси”. URL: <http://www.nbuv.gov.ua/> (дата звернення 10.05.2026).
9. Java Documentation. “Java Platform, Standard Edition Documentation”. URL: <https://docs.oracle.com/en/java/> (дата звернення 11.05.2026).
- 10.Spring. “Spring Boot Documentation”. URL: <https://docs.spring.io/spring-boot/index.html> (дата звернення 11.05.2026).
- 11.Spring. “Spring Framework Documentation”. URL: <https://docs.spring.io/spring-framework/reference/index.html> (дата звернення 11.05.2026).
- 12.Spring Security. “Spring Security Reference”. URL: <https://docs.spring.io/spring-security/reference/> (дата звернення 11.05.2026).
- 13.Spring Data. “Spring Data JPA Documentation”. URL: <https://docs.spring.io/spring-data/jpa/reference/> (дата звернення 11.05.2026).
- 14.PostgreSQL Global Development Group. “PostgreSQL Documentation”. URL: <https://www.postgresql.org/docs/> (дата звернення 11.05.2026).

- 15.Liquibase. “Liquibase Documentation”. URL: <https://docs.liquibase.com/> (дата звернення 11.05.2026).
- 16.Docker Docs. “Docker Documentation”. URL: <https://docs.docker.com/> (дата звернення 11.05.2026).
- 17.Docker Docs. “Docker Compose Documentation”. URL: <https://docs.docker.com/compose/> (дата звернення 11.05.2026).
- 18.React. “React Documentation”. URL: <https://react.dev/> (дата звернення 11.05.2026).
- 19.Vite. “Vite Documentation”. URL: <https://vite.dev/guide/> (дата звернення 11.05.2026).
- 20.TypeScript. “TypeScript Documentation”. URL: <https://www.typescriptlang.org/docs/> (дата звернення 11.05.2026).
- 21.Tailwind CSS. “Tailwind CSS Documentation”. URL: <https://tailwindcss.com/docs> (дата звернення 11.05.2026).
- 22.Git SCM. “Git Documentation”. URL: <https://git-scm.com/doc> (дата звернення 11.05.2026).
- 23.GitHub Docs. “GitHub Docs”. URL: <https://docs.github.com/> (дата звернення 11.05.2026).
- 24.OWASP Foundation. “OWASP Top Ten”. URL: <https://owasp.org/www-project-top-ten/> (дата звернення 11.05.2026).
- 25.JWT.IO. “Introduction to JSON Web Tokens”. URL: <https://jwt.io/introduction> (дата звернення 11.05.2026).
- 26.Goodreads. “Goodreads”. URL: <https://www.goodreads.com/> (дата звернення 10.05.2026).
- 27.The StoryGraph. “The StoryGraph”. URL: <https://app.thestorygraph.com/> (дата звернення 10.05.2026).
- 28.LibraryThing. “LibraryThing”. URL: <https://www.librarything.com/> (дата звернення 10.05.2026).
- 29.Libib. “Libib”. URL: <https://www.libib.com/> (дата звернення 10.05.2026).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ



ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



Розробка web-застосунку для ведення домашньої бібліотеки та трекінгу прочитаного матеріалу

Виконала студентка 4 курсу

групи ПД-44

Здоровенко Юлія Сергіївна

Керівник роботи

к.п.н., доцент, професор кафедри ІТ Корецька Вікторія Олександрівна

Київ - 2026

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Проаналізувати предметну галузь ведення домашньої бібліотеки та визначити основні проблеми організації, пошуку, систематизації й контролю читачького прогресу.
2. Дослідити існуючі програмні засоби для ведення книжкових колекцій, читачьких списків та обліку книг, визначити їх переваги, недоліки й функціональні обмеження.
3. Сформувати функціональні та нефункціональні вимоги до вебплатформи ShelfMate для ведення домашньої бібліотеки.
4. Обґрунтувати вибір технологій для реалізації клієнтської частини, серверної частини, бази даних, авторизації та контейнерного запуску системи.
5. Спроектувати логічну структуру вебплатформи ShelfMate, базу даних, UML-діаграми та основні алгоритми роботи системи.
6. Реалізувати основні модулі вебплатформи: авторизацію, профіль користувача, керування книгами, пошук, фільтрацію, читачький паспорт, рекомендації та адміністративну панель.
7. Розробити механізм контролю правового статусу цифрових матеріалів для обмеження доступу до PDF-файлів відповідно до типу доступу книги.
8. Провести тестування вебплатформи ShelfMate та проаналізувати результати перевірки основних функцій системи.

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: спрощення ведення домашньої бібліотеки за рахунок розробки вебплатформи ShelfMate для обліку книг, контролю читацького прогресу, формування персональної статистики, рекомендацій та керування доступом до цифрових матеріалів.

Об'єкт дослідження: процес організації, систематизації та контролю читацького прогресу в межах домашньої бібліотеки.

Предмет дослідження: засоби, методи та технології реалізації вебплатформи ShelfMate для ведення домашньої бібліотеки, обліку книг, формування читацького паспорта та контролю правового статусу цифрових матеріалів.

3

АНАЛІЗ АНАЛОГІВ

Критерій	Goodreads	The StoryGraph	LibraryThing	Libib	ShelfMate
Додавання книг	+	+	+	+	+
Редагування інформації про книгу	обмежене редагування	обмежене редагування	+	+	+
Пошук за назвою або автором	+	+	+	+	+
Фільтрація за жанром	базова фільтрація	+	+	+	+
Статуси читання	+	+	часткова підтримка	часткова підтримка	+
Збереження нотаток	+	обмежені нотатки	+	обмежені нотатки	+
Оцінювання книг	+	+	+	базове оцінювання	+
Відстеження прогресу за сторінками	часткове відстеження	+	часткове відстеження	-	+
Персональна статистика	базова статистика	+	базова статистика	базова статистика	+
Читацький паспорт	-	окремні елементи статистики	-	-	+
Рекомендації з власної бібліотеки	+	+	часткові рекомендації	-	+

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги

1. Реєстрація, авторизація та розмежування ролей користувача й адміністратора.
2. Перегляд, додавання, редагування та видалення книг у власній бібліотеці.
3. Пошук і фільтрація книг за назвою, автором, жанром та статусом читання.
4. Зміна статусу читання, додавання нотаток, оцінки та відстеження прогресу за сторінками.
5. Перегляд детальної сторінки книги та редагування профілю користувача.
6. Формування читачького паспорта та персональних рекомендацій.
7. Контроль доступу до цифрових матеріалів залежно від правового статусу книги.
8. Перегляд адміністративної панелі, списку користувачів, усіх книг, зміна ролей і правового статусу матеріалів адміністратором.

Нефункціональні вимоги

1. Час завантаження основних сторінок — не більше 3 секунд, відповідь сервера — 1–2 секунди.
2. Захист доступу до системи за допомогою JWT-токена.
3. Користувач має доступ лише до власної бібліотеки, а адміністративні функції доступні тільки ролі ADMIN.
4. Коректна робота в сучасних браузерях: Google Chrome, Mozilla Firefox, Microsoft Edge.
5. Адаптивність інтерфейсу для екранів від 360 px до 1920 px.
6. Можливість подальшого розширення системи новими сутностями: цитати, PDF-файли, сеанси читання, спільні полиці.
7. Стабільна робота при одночасному використанні не менше ніж 50 користувачами.
8. Повідомлення користувача про помилки введення даних або відсутність доступу.

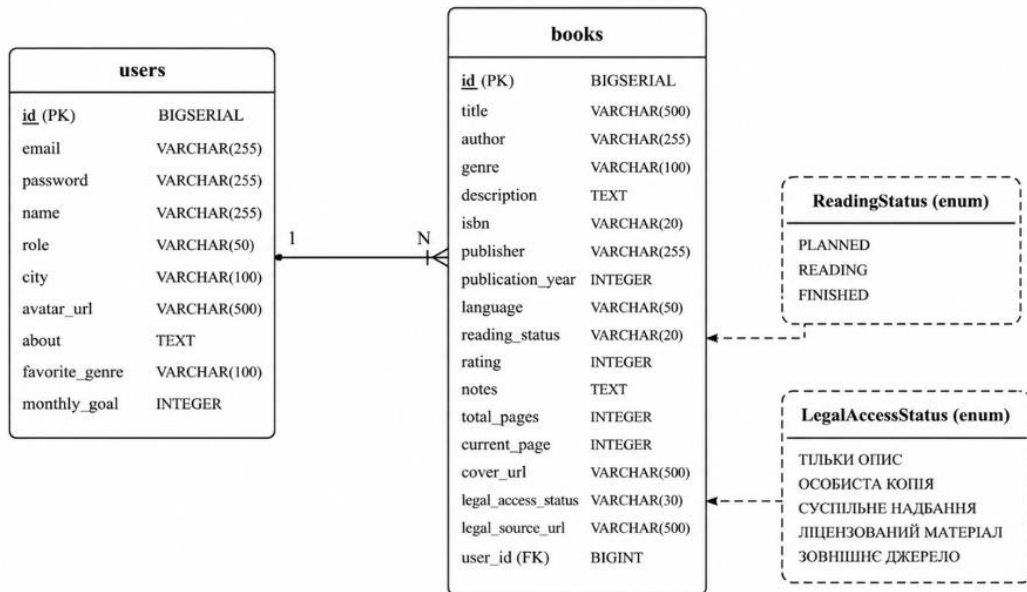
5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



6

СТРУКТУРА БАЗИ ДАНИХ



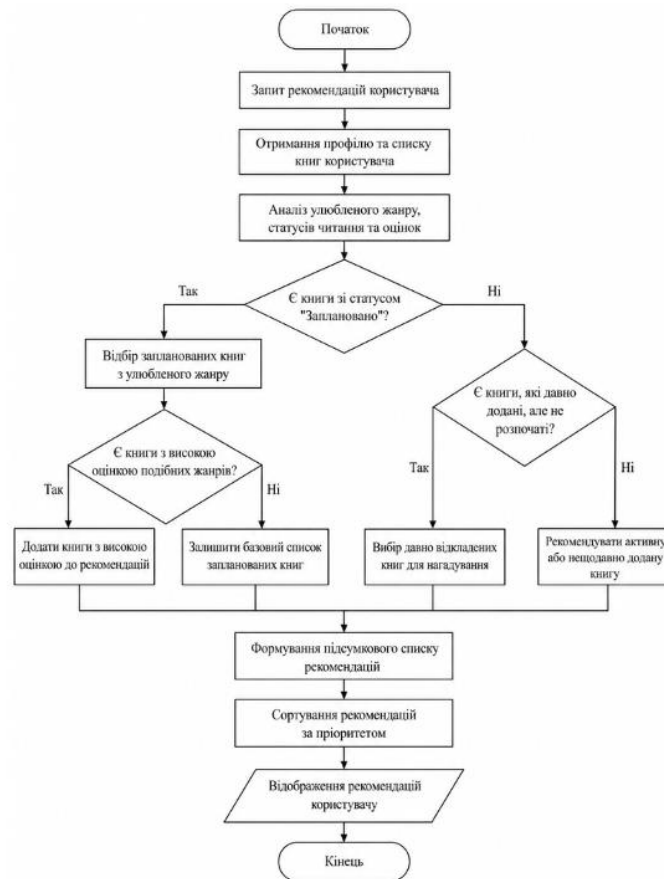
7

СТРУКТУРА АДМІНІСТРАТИВНОЇ ПАНЕЛІ



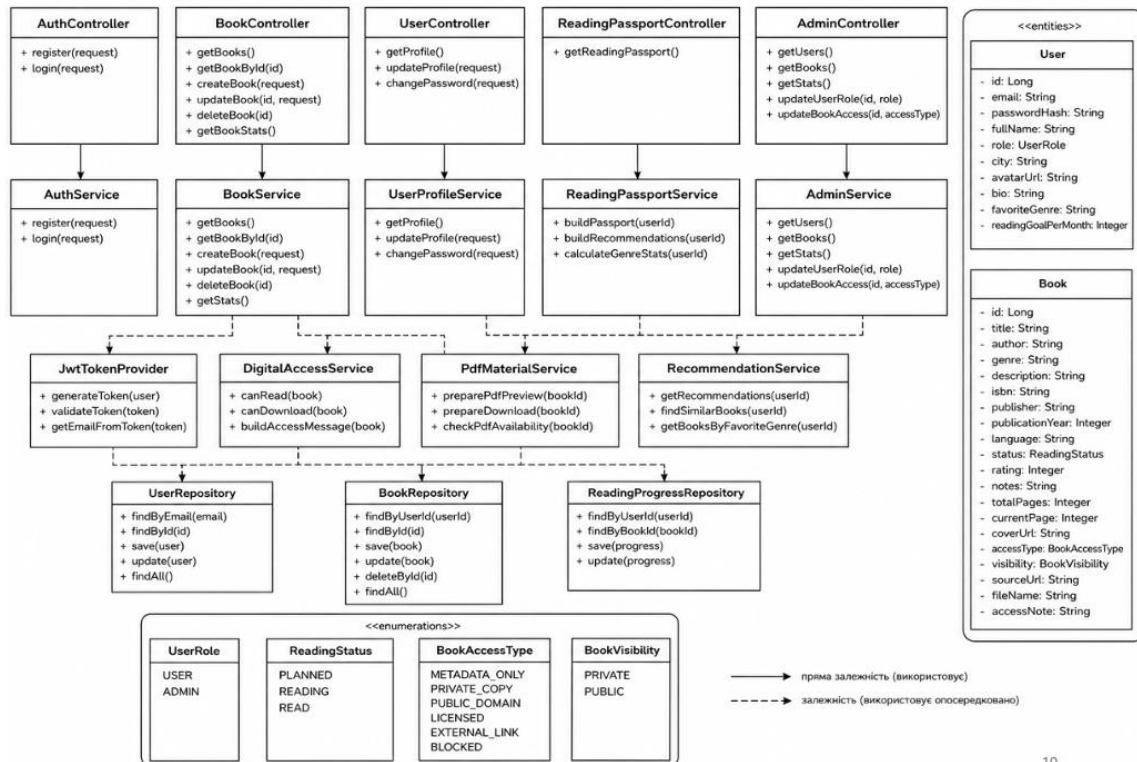
8

ДІАГРАМА ПОСЛІДОВНОСТІ ДОДАВАННЯ КНИГИ ДО БІБЛІОТЕКИ



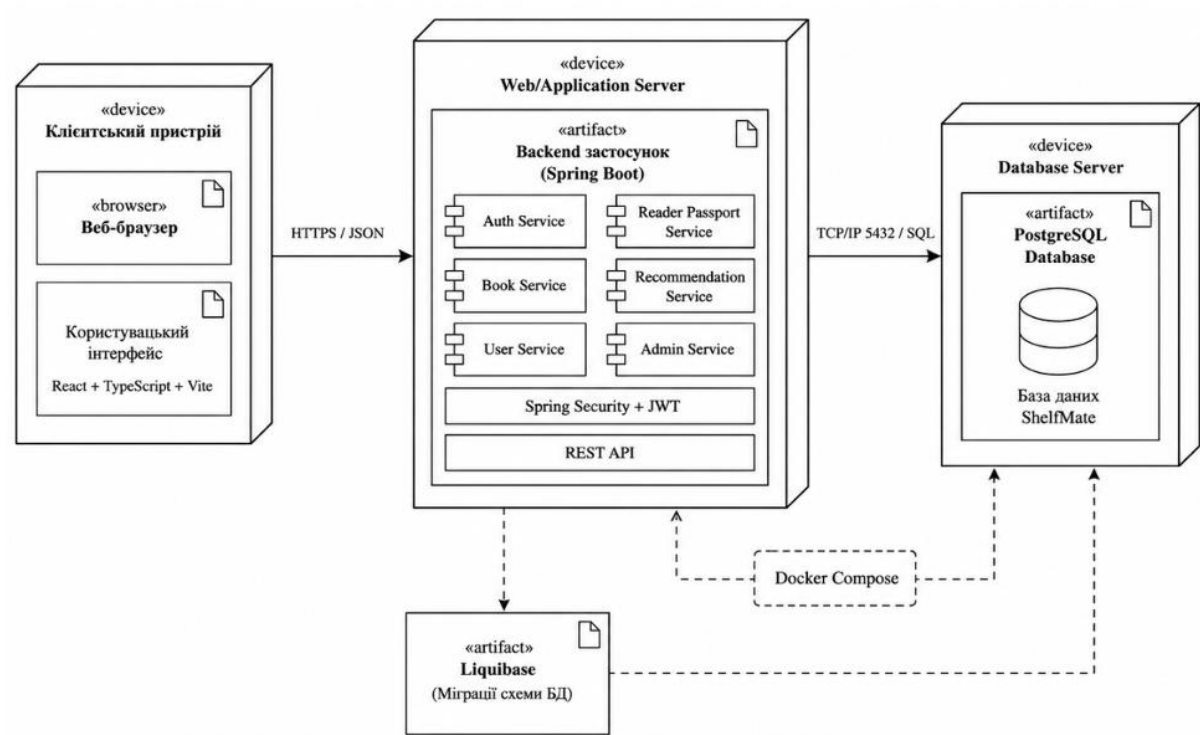
9

ДІАГРАМА КЛАСІВ



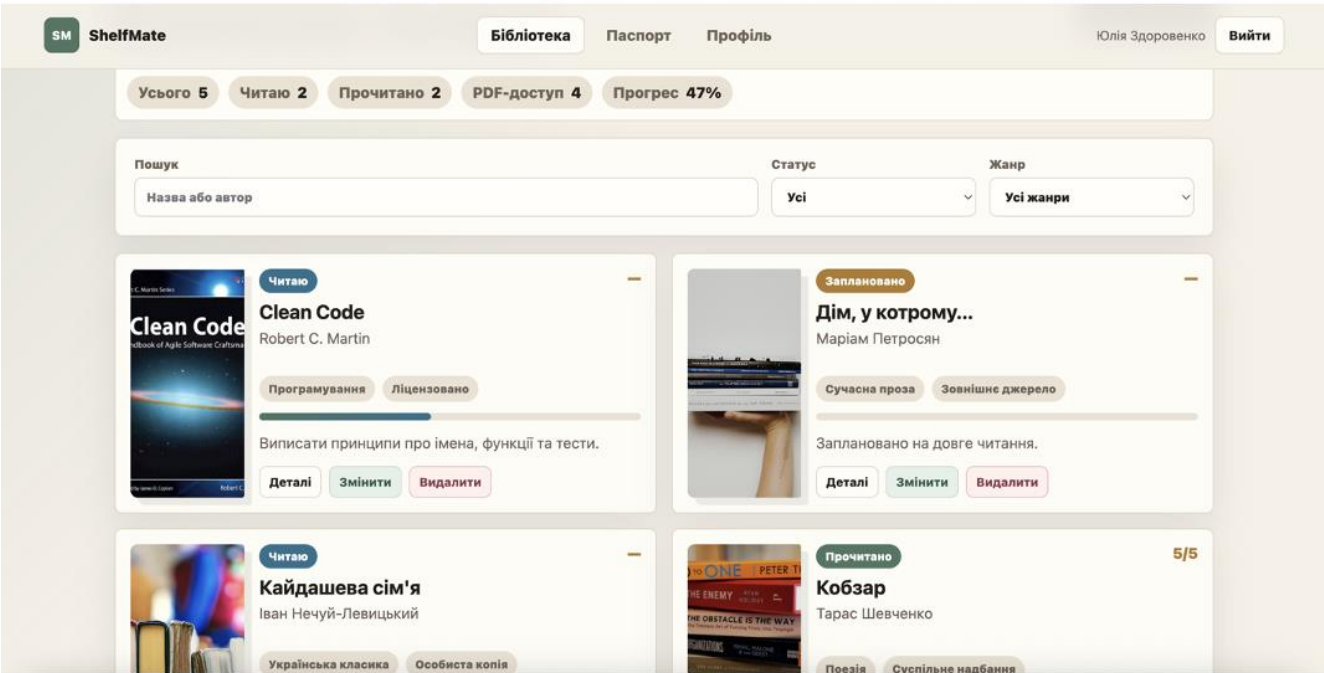
10

ДІАГРАМА РОЗГОРТАННЯ



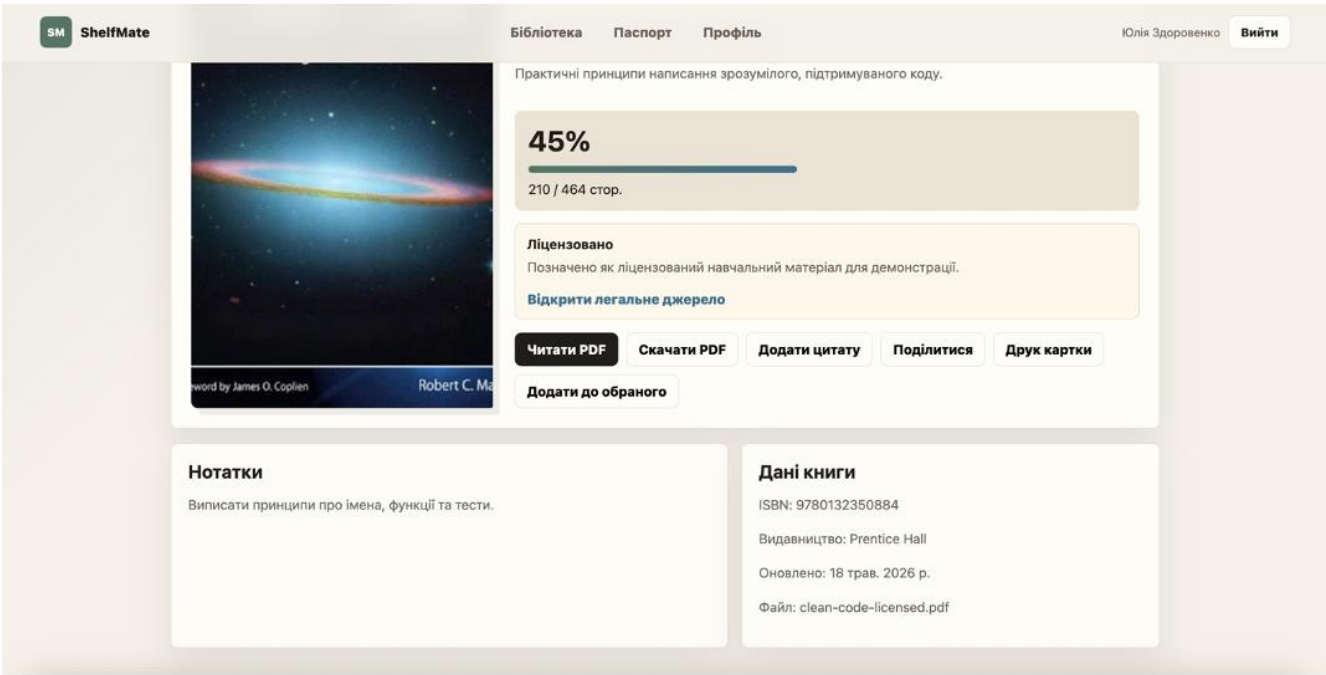
11

ЕКРАННІ ФОРМИ



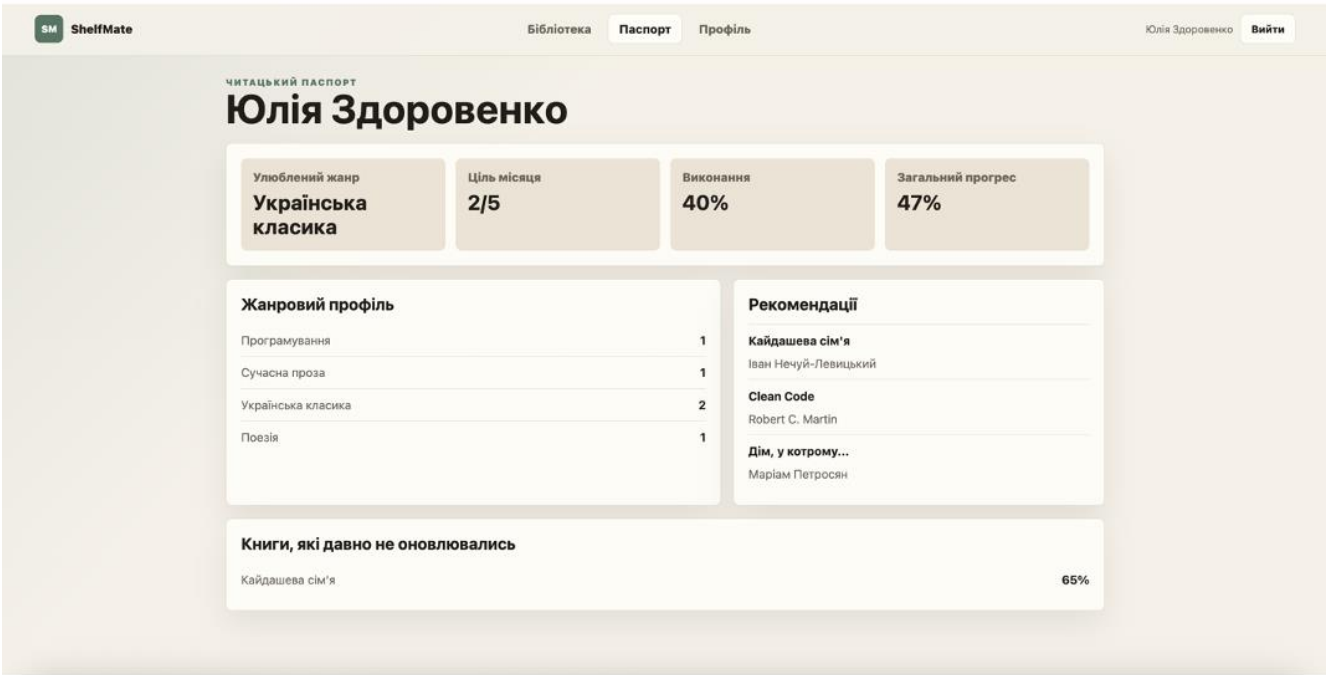
Головна сторінка

ЕКРАННІ ФОРМИ



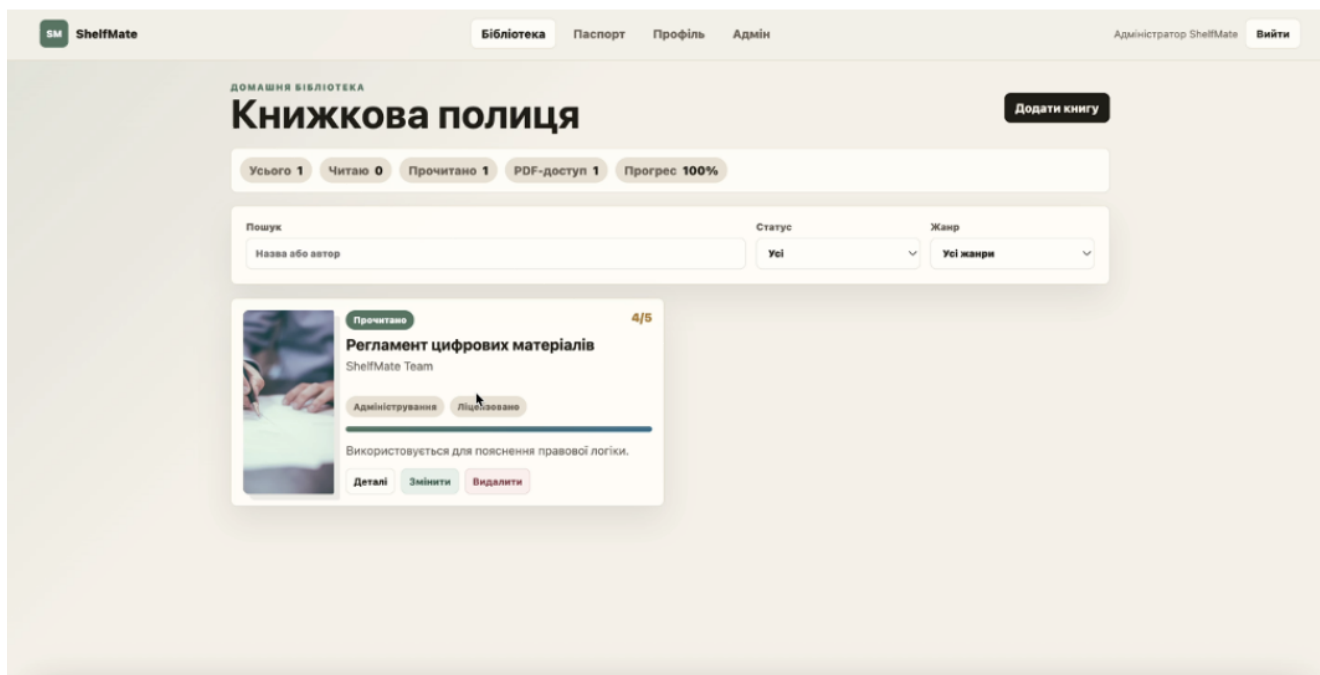
Детальна сторінка книги

ЕКРАННІ ФОРМИ



Сторінка читацького паспорта

ДЕМОНСТРАЦІЯ РОБОТИ ПРОГРАМИ



АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Тези доповідей:

1. Здоровенко Ю.С., Корецька В.О. Розробка web-застосунку для відстеження процесу тренувань. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2025, С.459-461.
2. Здоровенко Ю.С., Корецька В.О. Особливості ведення домашньої бібліотеки та трекінгу прочитаного матеріалу за допомогою веб-застосунку ShelfMate. Матеріали Всеукраїнської науково-технічної конференції «Технології цифрового розвитку: програмні системи та децентралізація», 1-3 травня 2026 року, Київ, ДУІКТ. Збірник тез. (подано до друку).

ВИСНОВКИ

1. Проаналізовано предметну галузь ведення домашньої бібліотеки та визначено основні проблеми: складність пошуку книг, контролю статусу читання, збереження нотаток, оцінок і читацького прогресу.
2. Досліджено існуючі рішення: Goodreads, The StoryGraph, LibraryThing, Libib, Excel і Google Sheets. Визначено, що вони не поєднують повноцінний каталог, читацький паспорт, рекомендації, адміністрування та контроль цифрових матеріалів.
3. Сформовано функціональні й нефункціональні вимоги до ShelfMate, зокрема авторизацію, роботу з книгами, пошук, фільтрацію, читацький паспорт, рекомендації та адміністративну панель.
4. Обґрунтовано вибір технологій: Java, Spring Boot, Spring Security, JWT, PostgreSQL, Liquibase, React, TypeScript та Vite.
5. Спроектовано клієнт-серверну архітектуру, базу даних, UML-діаграми та основні алгоритми роботи системи.
6. Реалізовано основні модулі ShelfMate: авторизацію, профіль, керування книгами, статуси читання, нотатки, оцінювання, читацький паспорт, рекомендації та адмінпанель.
7. Розроблено механізм контролю правового статусу цифрових матеріалів із типами доступу: тільки опис, особиста копія, суспільне надбання, ліцензований матеріал та зовнішнє джерело.
8. Проведено тестування системи та підтверджено коректну роботу основних функцій, розмежування ролей і відповідність вебплатформи поставленим вимогам.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

ShelfMateApplication.java

```
@SpringBootApplication
public class ShelfMateApplication {

    public static void main(String[]
args) {

SpringApplication.run(ShelfMateApplica
tion.class, args);
}
}
User.java
```

```
@Entity
@Table(name = "users")
@Getter
@Setter
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class User {

    @Id
    @GeneratedValue(strategy =
GenerationType.IDENTITY)
    private Long id;

    private String email;

    @Column(name = "password_hash")
    private String passwordHash;

    @Column(name = "full_name")
    private String fullName;

    @Enumerated(EnumType.STRING)
    private UserRole role;

    private String city;

    @Column(name = "avatar_url")
    private String avatarUrl;

    private String bio;

    @Column(name = "favorite_genre")
    private String favoriteGenre;

    @Column(name =
"reading_goal_per_month")
    private Integer
readingGoalPerMonth;

    private LocalDateTime createdAt;
```

```
private LocalDateTime updatedAt;
}
Book.java
```

```
@Entity
@Table(name = "books")
@Getter
@Setter
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class Book {

    @Id
    @GeneratedValue(strategy =
GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    private User user;

    private String title;
    private String author;
    private String genre;

    @Column(columnDefinition = "TEXT")
    private String description;

    private String isbn;
    private String publisher;

    @Column(name = "publication_year")
    private Integer publicationYear;

    private String language;

    @Enumerated(EnumType.STRING)
    private ReadingStatus status;

    private Integer rating;

    @Column(columnDefinition = "TEXT")
    private String notes;

    @Column(name = "total_pages")
    private Integer totalPages;

    @Column(name = "current_page")
    private Integer currentPage;

    @Column(name = "cover_url")
    private String coverUrl;

    @Enumerated(EnumType.STRING)
    private BookAccessType accessType;
```

```

    @Enumerated(EnumType.STRING)
    private BookVisibility visibility;

    @Column(name = "source_url")
    private String sourceUrl;

    @Column(name = "file_name")
    private String fileName;

    @Column(name = "access_note")
    private String accessNote;

    private LocalDateTime createdAt;
    private LocalDateTime updatedAt;
}
UserRole.java

public enum UserRole {
    USER,
    ADMIN
}
ReadingStatus.java

public enum ReadingStatus {
    PLANNED,
    READING,
    READ
}
BookAccessType.java

public enum BookAccessType {
    METADATA_ONLY,
    PRIVATE_COPY,
    PUBLIC_DOMAIN,
    LICENSED,
    EXTERNAL_LINK,
    BLOCKED
}
BookVisibility.java

public enum BookVisibility {
    PRIVATE,
    PUBLIC
}
AuthController.java

@RestController
@RequestMapping("/api/auth")
@RequiredArgsConstructor
public class AuthController {

    private final AuthService
    authService;

    @PostMapping("/register")
    public AuthResponse
    register(@Valid @RequestBody
    RegisterRequest request) {
        return
    authService.register(request);
    }

    @PostMapping("/login")

```

```

    public AuthResponse login(@Valid
    @RequestBody LoginRequest request) {
        return authService.login(request);
    }
}
AuthService.java

@Service
@RequiredArgsConstructor
public class AuthService {

    private final UserRepository
    userRepository;
    private final PasswordEncoder
    passwordEncoder;
    private final
    AuthenticationManager
    authenticationManager;
    private final JwtTokenProvider
    jwtTokenProvider;

    public AuthResponse
    register(RegisterRequest request) {
        if
        (userRepository.existsByEmail(request.
        getEmail())) {
            throw new
            ConflictException("Користувач з таким
            email вже існує");
        }

        User user = User.builder()
            .email(request.getEmail())

            .passwordHash(passwordEncoder.encode(r
            equest.getPassword()))
            .fullName(request.getFullName())
            .role(UserRole.USER)
            .build();

        userRepository.save(user);

        String token =
        jwtTokenProvider.generateToken(user.ge
        tEmail());

        return new AuthResponse(token,
        user.getEmail(), user.getFullName(),
        user.getRole());
    }

    public AuthResponse login(LoginRequest
    request) {

        authenticationManager.authenticate(
            new
            UsernamePasswordAuthenticationToken(
                request.getEmail(),
                request.getPassword()
            )
        );

        User user =
        userRepository.findByEmail(request.get

```

```

Email())
    .orElseThrow();

    String token =
jwtTokenProvider.generateToken(user.ge
tEmail());

    return new AuthResponse(token,
user.getEmail(), user.getFullName(),
user.getRole());
}
}
JwtTokenProvider.java

@Component
public class JwtTokenProvider {

    public String generateToken(String
email) {
        Date now = new Date();
        Date expiryDate = new
Date(now.getTime() + tokenExpiration);

        return Jwts.builder()
.subject(email)
.issuedAt(now)
.expiration(expiryDate)
.signWith(getSigningKey())
.compact();
}

    public String getEmailFromToken(String
token) {
        return Jwts.parser()
            .verifyWith(getSigningKey())
            .build()
            .parseSignedClaims(token)
            .getPayload()
            .getSubject();
    }

    public boolean validateToken(String
token) {
        try {
            Jwts.parser()

.verifyWith(getSigningKey())
            .build()
            .parseSignedClaims(token);
            return true;
        } catch (JwtException ex) {
            return false;
        }
    }
}
BookController.java

@RestController
@RequestMapping("/api/books")
@RequiredArgsConstructor
public class BookController {

    private final BookService
bookService;

```

```

    private final PdfMaterialService
pdfMaterialService;

    @GetMapping
    public List<BookResponse>
findAll(@AuthenticationPrincipal
UserDetails userDetails) {
        return
bookService.findAll(userDetails);
    }

    @GetMapping("/{id}")
    public BookResponse
findById(@PathVariable Long id,
@AuthenticationPrincipal UserDetails
userDetails) {
        return bookService.findById(id,
userDetails);
    }

    @PostMapping
    public BookResponse create(@Valid
@RequestBody BookRequest request,
@AuthenticationPrincipal UserDetails
userDetails) {
        return bookService.create(request,
userDetails);
    }

    @PutMapping("/{id}")
    public BookResponse
update(@PathVariable Long id,
@Valid @RequestBody BookRequest
request,
@AuthenticationPrincipal UserDetails
userDetails) {
        return bookService.update(id,
request, userDetails);
    }

    @DeleteMapping("/{id}")
    public void delete(@PathVariable Long
id,
@AuthenticationPrincipal UserDetails
userDetails) {
        bookService.delete(id,
userDetails);
    }

    @GetMapping("/{id}/pdf/download")
    public ResponseEntity<byte[]>
downloadPdf(@PathVariable Long id,
@AuthenticationPrincipal UserDetails
userDetails) {
        PdfMaterial material =
pdfMaterialService.prepareDownload(id,
userDetails);

        return ResponseEntity.ok()

.contentType(MediaType.APPLICATION_PDF
)

.header(HttpHeaders.CONTENT_DISPOSITIO

```

```

N,
        "attachment; filename=\""
+ material.fileName() + "\"")
        .body(material.content());
}
}
BookService.java

@Service
@RequiredArgsConstructor
public class BookService {

    private final BookRepository
bookRepository;
    private final UserProfileService
userProfileService;

    public List<BookResponse>
findAll(UserDetails userDetails) {
        User user =
userProfileService.currentUser(userDet
ails);

        return
bookRepository.findByUserOrderByUpdate
dAtDesc(user)
        .stream()
        .map(this::toResponse)
        .toList();
    }

    public BookResponse findById(Long id,
UserDetails userDetails) {
        User user =
userProfileService.currentUser(userDet
ails);

        Book book =
bookRepository.findByIdAndUser(id,
user)
            .orElseThrow(() -> new
ResourceNotFoundException("Книгу не
найджено"));

        return toResponse(book);
    }

    public BookResponse create(BookRequest
request, UserDetails userDetails) {
        User user =
userProfileService.currentUser(userDet
ails);

        Book book = Book.builder()
            .user(user)
            .build();

        applyRequest(book, request);

        return
toResponse(bookRepository.save(book));
    }

    public BookResponse update(Long id,

```

```

BookRequest request, UserDetails
userDetails) {
        User user =
userProfileService.currentUser(userDet
ails);

        Book book =
bookRepository.findByIdAndUser(id,
user)
            .orElseThrow(() -> new
ResourceNotFoundException("Книгу не
найджено"));

        applyRequest(book, request);

        return toResponse(book);
    }

    public void delete(Long id,
UserDetails userDetails) {
        User user =
userProfileService.currentUser(userDet
ails);

        Book book =
bookRepository.findByIdAndUser(id,
user)
            .orElseThrow(() -> new
ResourceNotFoundException("Книгу не
найджено"));

        bookRepository.delete(book);
    }
}
PdfMaterialService.java

@Service
@RequiredArgsConstructor
public class PdfMaterialService {

    private final BookRepository
bookRepository;
    private final UserProfileService
userProfileService;

    public PdfMaterial
preparePreview(Long bookId,
UserDetails userDetails) {
        Book book = findOwnedBook(bookId,
userDetails);
        validatePdfAccess(book, false);

        return new PdfMaterial(
            buildFileName(book,
"preview"),
            generatePlaceholderPdf(book,
false)
        );
    }

    public PdfMaterial
prepareDownload(Long bookId,
UserDetails userDetails) {
        Book book = findOwnedBook(bookId,

```

```

userDetails);
    validatePdfAccess(book, true);

    return new PdfMaterial(
        buildFileName(book, "legal-
placeholder"),
        generatePlaceholderPdf(book,
true)
    );
}

private void validatePdfAccess(Book
book, boolean download) {
    if (book.getAccessType() ==
BookAccessType.PRIVATE_COPY
        || book.getAccessType() ==
BookAccessType.PUBLIC_DOMAIN
        || book.getAccessType() ==
BookAccessType.LICENSED) {
        return;
    }

    if (book.getAccessType() ==
BookAccessType.EXTERNAL_LINK) {
        throw new
IllegalArgumentException(
            "Для цієї книги доступне
лише зовнішнє легальне джерело"
        );
    }

    if (book.getAccessType() ==
BookAccessType.BLOCKED) {
        throw new
AccessDeniedException(
            "Адміністратор обмежив
доступ до цифрового матеріалу"
        );
    }

    throw new
IllegalArgumentException(
        "Для цієї книги доступні
тільки бібліографічні дані"
    );
}

private byte[]
generatePlaceholderPdf(Book book,
boolean download) {
    // Генерує PDF-заглушку без тексту
    книги
    // Файл містить тільки назву,
    автора, тип доступу та юридичне
    пояснення
}

public record PdfMaterial(String
fileName, byte[] content) {
}
}
UserController.java

@RestController

```

```

@RequestMapping("/api/users")
@RequiredArgsConstructor
public class UserController {

    private final UserProfileService
userService;

    @GetMapping("/profile")
    public UserResponse
getProfile(@AuthenticationPrincipal
UserDetails userDetails) {
        return
userService.getProfile(userDeta
ils);
    }

    @PutMapping("/profile")
    public UserResponse
updateProfile(@Valid @RequestBody
ProfileUpdateRequest request,
@AuthenticationPrincipal UserDetails
userDetails) {
        return
userService.updateProfile(reque
st, userDetails);
    }

    @PutMapping("/password")
    public void changePassword(@Valid
@RequestBody PasswordUpdateRequest
request,
@AuthenticationPrincipal UserDetails
userDetails) {

        userService.changePassword(requ
est, userDetails);
    }
}
UserProfileService.java

@Service
@RequiredArgsConstructor
public class UserProfileService {

    private final UserRepository
userRepository;
    private final PasswordEncoder
passwordEncoder;

    public User
currentUser(UserDetails userDetails) {
        return
userRepository.findByEmail(userDetails
.getUsername())
        .orElseThrow(() -> new
ResourceNotFoundException("Користувача
не знайдено"));
    }

    public UserResponse
getProfile(UserDetails userDetails) {
        User user =
currentUser(userDetails);
        return

```

```

    UserMapper.toResponse(user);
}

public UserResponse
updateProfile(ProfileUpdateRequest
request,
    UserDetails userDetails) {
    User user =
currentUser(userDetails);

    user.setFullName(request.getFullName());
    user.setCity(request.getCity());
    user.setBio(request.getBio());

    user.setAvatarUrl(request.getAvatarUrl());

    user.setFavoriteGenre(request.getFavoriteGenre());

    user.setReadingGoalPerMonth(request.getReadingGoalPerMonth());

    return
    UserMapper.toResponse(userRepository.save(user));
}

public void
changePassword(PasswordUpdateRequest
request,
    UserDetails userDetails) {
    User user =
currentUser(userDetails);

    if
    (!passwordEncoder.matches(request.getCurrentPassword(),
    user.getPasswordHash())) {
        throw new
        IllegalArgumentException("Поточный
        пароль неправильный");
    }

    user.setPasswordHash(passwordEncoder.encode(request.getNewPassword()));
    userRepository.save(user);
}
}
ReadingPassportController.java

@RestController
@RequestMapping("/api/reading-passport")
@RequiredArgsConstructor
public class ReadingPassportController
{

    private final
    ReadingPassportService
    readingPassportService;

```

```

    @GetMapping
    public ReadingPassportResponse
    getReadingPassport(
        @AuthenticationPrincipal
        UserDetails userDetails
    ) {
        return
        readingPassportService.buildPassport(userDetails);
    }
}
ReadingPassportService.java

@Service
@RequiredArgsConstructor
public class ReadingPassportService {

    private final BookRepository
    bookRepository;
    private final UserProfileService
    userProfileService;
    private final BookService
    bookService;

    public ReadingPassportResponse
    buildPassport(UserDetails userDetails)
    {
        User user =
        userProfileService.currentUser(userDetails);

        List<Book> books =
        bookRepository.findByUserOrderByUpdatedAtDesc(user);

        LibraryStatsResponse stats =
        bookService.calculateStats(books);

        Map<String, Long> genreStats =
        books.stream()
            .filter(book ->
            book.getGenre() != null)

        .collect(Collectors.groupingBy(Book::getGenre, Collectors.counting()));

        List<BookResponse> recommendations
        =
        bookService.recommendedBooks(books,
        user.getFavoriteGenre());

        return new
        ReadingPassportResponse(
            stats,
            user.getFavoriteGenre(),
            user.getReadingGoalPerMonth(),
            genreStats,
            recommendations
        );
    }
}
AdminController.java

```

```

@RestController
@RequestMapping("/api/admin")
@RequiredArgsConstructor
public class AdminController {

    private final AdminService
    adminService;

    @GetMapping("/users")
    public List<UserResponse>
    getUsers() {
        return
        adminService.getUsers();
    }

    @GetMapping("/books")
    public List<BookResponse>
    getBooks() {
        return
        adminService.getBooks();
    }

    @GetMapping("/stats")
    public AdminStatsResponse
    getStats() {
        return
        adminService.getStats();
    }

    @PutMapping("/users/{id}/role")
    public UserResponse
    updateUserRole(@PathVariable Long id,
        @RequestBody UserRoleUpdateRequest
        request) {
        return
        adminService.updateUserRole(id,
        request);
    }

    @PutMapping("/books/{id}/access")
    public BookResponse
    updateBookAccess(@PathVariable Long
    id,
    @RequestBody BookAccessUpdateRequest
    request) {
        return
        adminService.updateBookAccess(id,
        request);
    }
}
AdminService.java

@Service
@RequiredArgsConstructor
public class AdminService {

    private final UserRepository
    userRepository;
    private final BookRepository
    bookRepository;
    private final BookService
    bookService;

    public List<UserResponse>

```

```

    getUsers() {
        return
        userRepository.findAll()
        .stream()

        .map(UserMapper::toResponse)
        .toList();
    }

    public List<BookResponse>
    getBooks() {
        return
        bookRepository.findAllByOrderByUpdated
        AtDesc()
        .stream()

        .map(bookService::toResponse)
        .toList();
    }

    public AdminStatsResponse
    getStats() {
        return new AdminStatsResponse(
            userRepository.count(),
            bookRepository.count(),

            bookRepository.countByAccessType(BookA
            ccessType.METADATA_ONLY),

            bookRepository.countByAccessType(BookA
            ccessType.LICENSED),

            bookRepository.countByAccessType(BookA
            ccessType.BLOCKED)
        );
    }

    public UserResponse
    updateUserRole(Long id,
        UserRoleUpdateRequest request) {
        User user =
        userRepository.findById(id)
        .orElseThrow(() -> new
        ResourceNotFoundException("Користувача
        не знайдено"));

        user.setRole(request.getRole());

        return
        UserMapper.toResponse(userRepository.s
        ave(user));
    }

    public BookResponse
    updateBookAccess(Long id,
        BookAccessUpdateRequest request) {
        Book book =
        bookRepository.findById(id)
        .orElseThrow(() -> new
        ResourceNotFoundException("Книгу не
        знайдено"));

        book.setAccessType(request.getAccessTy

```

```

    pe());

    book.setAccessNote(request.getAccessNote());

    return
    bookService.toResponse(bookRepository.
    save(book));
}
}
BookRepository.java

public interface BookRepository
extends JpaRepository<Book, Long> {

    List<Book>
    findByUserOrderByUpdatedAtDesc(User
    user);

    List<Book>
    findAllByOrderByUpdatedAtDesc();

    Optional<Book> findByIdAndUser(Long
    id, User user);

    long countByUserAndStatus(User user,
    ReadingStatus status);

    long countByAccessType(BookAccessType
    accessType);
}
UserRepository.java

public interface UserRepository
extends JpaRepository<User, Long> {

    Optional<User> findByEmail(String
    email);

    boolean existsByEmail(String email);

    long countByRole(UserRole role);
}

```