**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**

**Факультет прикладної математики**

**Кафедра прогрмного забезпечення комп'ютерних систем**

«До захисту допущено»
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___» _____ 2023 р.

# Дипломний проєкт

**на здобуття ступеня бакалавра**

**за освітньо-професійною програмою**
**«Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»**

**спеціальності 121 Інженерія програмного забезпечення**

**на тему: «Вебзастосунок електронної бібліотеки»**

Виконав:
студент IV курсу, групи КП-91
Вербовий Дмитро Станіславович _____

Керівник:
Старший викладач кафедри ПЗКС, к.т.н.,
Шкурат Оксана Сергіївна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Асистент кафедри СПіСКС,
Радченко Костянтин Олександрович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2023 року

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп'ютерних систем**

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___»_____2022 р.

# ЗАВДАННЯ

**на дипломний проєкт студенту**

Вербовому Дмитру Станіславовичу

1. Тема проєкту «Вебзастосунок електронної бібліотеки», керівник проєкту Шкурат Оксана Сергіївна, к.т.н., старший викладач, затверджені наказом по університету від «31» травня 2023 р. № 2107-с

2. Термін подання студентом проєкту «16» червня 2023 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - аналіз існуючих рішень та обґрунтування теми дипломного проєкту;
   - обґрунтування вибору засобів реалізації;
   - розроблення вебзастосунку електронної бібліотеки;
   - аналіз розробленого додатка.

5. Перелік обов'язкового графічного матеріалу:
   - архітектура системи вебзастосунку (креслення);
   - функціональність застосунку (креслення);
   - структура бази даних (плакат);
   - алгоритм роботи програми (плакат).

6. Консультанти розділів проєкту

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Нормоконтроль | Онай М.В., доцент | | |

7. Дата видачі завдання «25» грудня 2022 р.

Календарний план

| № з/п | Назва етапів виконання дипломного проєкту | Термін виконання етапів проєкту | Примітка |
|---|---|---|---|
| 1. | Вивчення літератури за тематикою роботи | 17.11.2022 | |
| 2. | Розроблення та узгодження технічного завдання | 27.11.2022 | |
| 3. | Розроблення структури вебзастосунку | 15.12.2023 | |
| 4. | Підготовка матеріалів першого розділу дипломного проєкту | 05.01.2023 | |
| 5. | Розроблення дизайну сторінок та графічних елементів | 01.02.2023 | |
| 6. | Підготовка матеріалів другого розділу дипломного проєкту | 01.03.2023 | |
| 7. | Програмна реалізація вебзастосунку | 15.03.2023 | |
| 8. | Тестування вебзастосунку | 01.04.2023 | |
| 9. | Підготовка матеріалів третього розділу дипломного проєкту | 15.04.2023 | |
| 10. | Підготовка матеріалів четвертого розділу дипломного проєкту | 30.04.2023 | |
| 11. | Підготовка матеріалів графічної частини дипломного проєкту | 10.05.2023 | |
| 12. | Оформлення технічної документації дипломного проєкту | 25.05.2023 | |

Студент Дмитро ВЕРБОВИЙ

Керівник проєкту Оксана ШКУРАТ

# АНОТАЦІЯ

Дипломний проект присвячений розробці вебзастосунку електронної бібліотеки. Метою проекту є створення зручної та ефективної системи організації, зберігання та доступу до широкого кола літературних джерел.

Основною метою дипломного проекту є розробка вебзастосунку, який задовольняє потреби читачів, дослідників і студентів відповідних навчальних закладів. Вебзастосунок надає вичерпну інформацію та підтримку для їх навчання та академічних занять.

У процесі розробки проведено поглиблений порівняльний аналіз існуючих інформаційних рішень електронних бібліотек. Було оцінено сильні та слабкі сторони цих рішень, що призвело до визначення основних вимог до вебдодатку.

Отриманий програмний продукт є багатофункціональним вебзастосунком, який використовує ретельно підібрані бібліотеки, фреймворки та технології як для клієнтських, так і для серверних компонентів. Конструкція також включає надійні системи керування базами даних для забезпечення безпечного зберігання даних користувачів. Розроблено алгоритми реєстрації та авторизації користувачів, а також завантаження різних типів мультимедійного контенту, наприклад відео, аудіо, тексту та графіки.

# ABSTRACT

The diploma project focuses on the development of a web application electronic library platform. The aim of the project is to create a user-friendly and efficient system for organizing, storing, and accessing a wide range of literary sources.

The primary objective of the diploma project is to develop a web application that caters to the needs of readers, researchers, and students in relevant educational institutions. The web application will provide comprehensive information and support for their training and academic pursuits.

During the development process, an in-depth comparative analysis of existing information solutions of electronic libraries. The strengths and weaknesses of these solutions were evaluated, leading to the identification of essential requirements for the web application.

The resulting software product is a feature-rich web application that leverages carefully chosen libraries, frameworks, and technologies for both the client and server components. The design also incorporates robust database management systems to ensure secure storage of user data. Algorithms for user registration and authorization, as well as for uploading various types of multimedia content such as video, audio, text, and graphics, were developed.

ДП.045440-01-90 Вебзастосунок електронної бібліотеки. Відомість проєкту

| Позначення | Найменування | Кіл-ть | Примітка |
|---|---|---|---|
| | Документація проєкту | | |
| | | | |
| ДП.045440-02-91 | Вебзастосунок електронної | 6 | |
| | бібліотеки. Технічне завдання | | |
| | | | |
| ДП.045440-03-81 | Вебзастосунок електронної | 64 | |
| | бібліотеки. Пояснювальна | | |
| | записка | | |
| | | | |
| ДП.045440-04-51 | Вебзастосунок електронної | 4 | |
| | бібліотеки. Програма та | | |
| | методика тестування | | |
| | | | |
| ДП.045440-05-34 | Вебзастосунок електронної | 18 | |
| | бібліотеки. Керівництво | | |
| | користувача | | |
| | | | |
| ДП.045440-06-99 | Вебзастосунок електронної | 1 | |
| | бібліотеки. Архітектура | | |
| | вебзастосунку. Схема бази | | |
| | даних | | |
| | | | |
| ДП.045440-07-99 | Вебзастосунок електронної | 1 | |
| | бібліотеки. Функціональні | | |
| | можливості вебзастосунку. | | |
| | Діаграма використання | | |
| | | | |
| ДП.045440-08-98 | Вебзастосунок електронної | 1 | |
| | бібліотеки. Компакт-диск | | |
| | | | |

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп'ютерних систем**

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___» _____ 2022 р.

# ВЕРБЗАСТОСУНОК ЕЛЕКТРОННОЇ БІБЛІОТЕКИ

## Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Оксана ШКУРАТ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Дмитро ВЕРБОВИЙ

2022

**ЗМІСТ**

## 1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

**Назва розробки**: Вебзастосунок електронної бібліотеки

**Галузь застосування**: інформаційні технології.

## 2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

## 3. ПРИЗНАЧЕННЯ РОЗРОБКИ

У сучасному світі все більше людей віддають перевагу електронним джерелам інформації, зокрема підручникам, книгам, журналам та статтям, оскільки це більш зручно та економно в порівнянні з традиційними персональними сховищами або закладами зберігання друкованих матеріалів для колективного користування.

Сучасні електронні бібліотеки мають різні недоліки, наприклад є обмеженими у виборі матеріалів, незручними у використанні та мають недосконалі алгоритми пошуку та прогнозування. Призначенням розробки вебзастосунку електронної бібліотеки є створення електронного літературного фонду зі зручним мультимедійним інтерфейсом, який дозволить організовувати електронні ресурси, зокрема зберігати, здійснювати пошук та прогнозування, перетворювати та візуалізувати.

Розроблення «Вебзастосунок електронної бібліотеки» покращить доступ до літературного фонду, що сприятиме розвитку освіти та культури в цілому.

# 4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Вебзастосунок повинен задовольняти наступні вимоги:

1. Функціональність: програма повинна мати можливість пошуку та збереження документів в електронній бібліотеці, а також забезпечувати доступ до них користувачам. Для цього необхідно реалізувати такі функції, як додавання нових документів, редагування та видалення існуючих документів, пошук та сортування за різними параметрами, перегляд та завантаження документів.
2. Зручний та інтуїтивно зрозумілий інтерфейс: програма повинна мати зрозумілий інтерфейс для користувачів. Це означає, що дизайн програми повинен бути простим та лаконічним, а всі функції мають бути легко доступні та зрозумілі.
3. Безпека: програма повинна забезпечувати конфіденційність даних користувачів та документів. Для цього необхідно реалізувати механізми авторизації та аутентифікації користувачів, а також забезпечити захист даних.
4. Масштабованість: програма повинна бути придатною до масштабування. Програма повинна працювати з великою кількістю документів та користувачів, а також бути придатною до додавання нових функціональних можливостей в майбутньому.
5. Сумісність: програма повинна бути сумісна з різними браузерами та операційними системами, щоб забезпечити доступність для максимальної кількості користувачів. Для цього необхідно тестувати програму на різних платформах та браузерах, а також забезпечити її сумісність з популярними операційними системами та браузерами. Також важливо забезпечити адаптивність програми,

щоб вебзастосунок коректно відображався на екранах з різними розмірами.

6. Надійність: програма повинна бути стійкою до помилок та аварійних ситуацій. Для цього необхідно забезпечити резервне копіювання даних та забезпечити їх безперебійний доступ. Також важливо реалізувати механізми моніторингу та логування, які дозволять швидко виявляти та виправляти можливі проблеми.
7. Швидкодія: програма повинна працювати швидко та ефективно. Для цього необхідно забезпечити оптимізацію коду та запитів до бази даних, а також використовувати сучасні технології для покращення швидкодії програми.

## 5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

1) пояснювальна записка;
2) програма та методика тестування;
3) керівництво користувача;
4) креслення:
   - «Вебзастосунок електронної бібліотеки. Архітектура вебзастосунку. Схема бази даних»;
   - «Вебзастосунок електронної бібліотеки. Функціональні можливості вебзастосунку. Діаграма використання»;
5) плакати:
   - «Вебзастосунок електронної бібліотеки. Алгоритм роботи програми»;
   - «Вебзастосунок електронної бібліотеки. Архітектура вебзастосунку. Діаграма компонентів».

## 6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи ..... 17.11.2022

Розроблення та узгодження технічного завдання ..... 27.11.2022

Розроблення структури вебзастосунку ..... 15.12.2023

Розроблення дизайну сторінок та графічних елементів..... 01.02.2023

Програмна реалізація вебзастосунку..... 15.03.2023

Тестування вебзастосунку ..... 01.04.2023

Підготовка матеріалів текстової частини проєкту ..... 30.04.2023

Підготовка матеріалів графічної частини проєкту ..... 10.05.2023

Оформлення технічної документації проєкту ..... 25.05.2023

## 7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до «Програми та методики тестування».

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп'ютерних систем**

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___» _____ 2023 р.

# ВЕРБЗАСТОСУНОК ЕЛЕКТРОННОЇ БІБЛІОТЕКИ

**Пояснювальна записка**

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Оксана ШКУРАТ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Дмитро ВЕРБОВИЙ

2023

**ЗМІСТ**

# СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Вебзастосунок – це програмне забезпечення, яке працює на вебсервері та доступ до якого здійснюється через веббраузер. Він забезпечує інтерактивні функції та може використовуватися для виконання різноманітних завдань або надання певних послуг через Інтернет.

Діджиталізація – це процес перетворення аналогової інформації в цифровий формат. Він передбачає збір, зберігання та обробку даних у цифровій формі, що полегшує маніпуляції, аналіз і зберігання.

Авторизація – це процес надання або заборони доступу до ресурсів або функціональних можливостей на основі особи користувача, ролей або дозволів. Це гарантує, що користувачі мають доступ лише до тих ресурсів, які їм дозволено використовувати.

Аутентифікація – це процес перевірки особи користувача або сутності. Це передбачає перевірку облікових даних, наданих користувачем, наприклад імені користувача та пароля, щоб переконатися, що користувач є тим, за кого себе видає.

Шифрування – це процес перетворення даних у закодовану форму для запобігання несанкціонованому доступу. Він використовує алгоритми для шифрування даних, що робить їх нечитабельними без відповідного ключа дешифрування.

Онлайн – підключенний до мережі, як правило, до Інтернету. Це означає, що пристрій або система активно підключені та можуть отримувати доступ або надавати послуги через мережу.

Офлайн – відключення від мережі, наприклад Інтернету. Це означає, що пристрій або система не підключені активно та можуть мати обмежений або взагалі не мати доступу до онлайн-сервісів.

Локалізація – це процес адаптації програмного забезпечення, вебсайту або вмісту відповідно до мовних, культурних і регіональних уподобань певної цільової аудиторії. Це передбачає переклад вмісту, коригування форматів і врахування місцевих умов.

Функціональні можливості – можливості або функції, які надає програмне забезпечення або система. Вони описують, що може робити додаток і як він поводиться у відповідь на дії або введення користувача.

Валідація – це процес перевірки відповідності даних або вхідних даних певним вимогам або обмеженням. Це гарантує, що дані точні, узгоджені та відповідають певним правилам або стандартам.

Надмножина – у розробці програмного забезпечення наднабір відноситься до набору, який містить усі елементи іншого набору, а також додаткові елементи. Він розширює або включає функції, функціональність або можливості іншого набору чи компонента.

Фронтенд – відноситься до клієнтської частини вебдодатка або програмної системи. Він включає інтерфейс користувача, рівень презентації та взаємодію з користувачами. Інтерфейсні технології зазвичай включають HTML, CSS і JavaScript.

Бекенд – відноситься до серверної частини вебдодатка або програмної системи. Він виконує такі завдання, як обробка даних, зберігання, бізнес-логіка та зв’язок із базами даних або зовнішніми службами.

Фреймворк – це інструмент або платформа розробки програмного забезпечення, яка забезпечує структуру, набір бібліотек і багаторазово використовувані компоненти для полегшення розробки програм. Він пропонує попередньо визначені функції та допомагає оптимізувати процес розробки.

Реактивне програмування – це парадигма програмування, яка зосереджена на створенні програм, які реагують на зміни та події в реальному часі. У ньому наголошується на асинхронному та керованому подіями програмуванні, що забезпечує адаптивні та масштабовані системи.

Кешування – це процес зберігання даних, до яких часто звертаються, у кеш-пам’яті або сховищі для підвищення продуктивності та зменшення потреби в повторюваних обчисленнях або пошуку даних із вихідного джерела.

Ідентифікація – в контексті програмних систем ідентифікація відноситься до унікального представлення або інформації, пов’язаної з користувачем, сутністю або компонентом. Вона використовується для автентифікації, авторизації та розрізнення різних користувачів або об’єктів.

Інтерфейс – визначає контракт або набір методів, які має реалізувати клас або компонент. Він визначає доступні операції, входи та виходи, які можна використовувати для взаємодії з класом або компонентом.

Конфігурація – відноситься до налаштувань, параметрів або опцій, які визначають поведінку або властивості програми або системи. Це дозволяє налаштувати та адаптувати до конкретних вимог або середовища.

Репозиторій – це компонент зберігання або керування даними, який забезпечує централізоване розташування для зберігання, отримання та керування даними. Він абстрагує доступ до даних і забезпечує інтерфейс для взаємодії з джерелом даних.

Компіляція – це процес перетворення вихідного коду програми, написаного на високорівневій мові програмування, в машинний код, який може бути виконаний комп'ютером або іншим пристроєм. Під час компіляції, компілятор (спеціальний програмний інструмент) аналізує синтаксис і структуру вихідного коду, перетворює його на послідовність низькорівневих інструкцій, які можуть бути розуміні та виконані обчислювальним пристроєм.

# ВСТУП

З сучасним розвитком інформаційних технологій, кількість користувачів електронних бібліотек неперервно зростає. Все більше людей стають прихильниками читання електронних джерел, зокрема книг, статей, журналів, підручників тощо. Електронні бібліотеки дозволяють зберігати, організовувати джерела інформації та здійснювати пошук, а також здобувати знання у будь-який момент часу, що є зручним для користувачів з різних куточків світу. Проте існують проблеми з використанням існуючих електронних бібліотек, зокрема погано організований пошук, відсутність рекомендацій на основі областей інтересу та/або обраної літератури, нестабільна робота вебзастосунків, обмежений доступ до літератури та інше.

Метою дипломного проєкту є розроблення вебзастосунку електронної бібліотеки для забезпечення користувачам доступу до літературних джерел. Вебзастосунок електронної бібліотеки забезпечить:

- авторизацію та аутентифікацію користувачів;
- зручний та інтуїтивно зрозумілий інтерфейс користувачів;
- організацію літературних джерел, зокрема пошук, збереження та відтворення інформації у кабінетах користувачів;
- зручно організований пошук літературних джерел.

Вебзастосунок електронної бібліотеки є зручним інструментом для отримання необхідної інформації та розвитку культури читання серед широкого кола користувачів.

# 1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЄКТУ

## 1.1. Загальний опис предметної області

У сучасному світі, в якому інформація є ключовим фактором електронні бібліотеки набувають все більшої популярності, забезпечуючи зручний доступ до різних джерел інформації, зокрема книг, статей, журналів, підручників тощо.

Основними користувачами вебзастосунку електронної бібліотеки є студенти, викладачі, науковці та фахівці різних галузей. Вебзастосунок дозволить різним користувачам зручно та швидко знаходити потрібну інформацію, зберігати та організовувати матеріали для подальшого використання.

У зв’язку зі зростанням темпів диджиталізації суспільства вебзастосунки електронних бібліотек набули широку популярність. Однак, у зв'язку зі значним зростанням обсягу інформації та відповідно електронних ресурсів, їх розмаїтістю, виникає проблема організації цієї інформації, зокрема пошук та вибір потрібної інформації, оцінка її відповідності запиту користувачів тощо.

Таким чином, метою розроблення вебзастосунку електронної бібліотеки є забезпечення користувачів доступом до електронного сховища літературних джерел та інструментами організації цієї інформації. Також вебзастосунок зможе забезпечити підтримку навчального процесу та наукових досліджень, забезпечуючи доступ до актуальної та достовірної інформації.

Для досягнення мети розроблення вебзастосунку електронної бібліотеки необхідно провести аналіз предметної області, зокрема, дослідити ринок програмних застосунків електронних бібліотек, їх функціональні можливості та особливості, а також потреби та очікування потенційних користувачів.

У сучасному інформаційному суспільстві існує велика кількість різних додатків електронних бібліотек, які надають користувачам різноманітні сервіси та функції. Деякі вебзастосунки обмежені доступом до певного типу літературних джерел, зокрема наукові статті та матеріали конференцій. Інші вебзастосунки обмежуються певними галузями знань, зокрема медицина, право тощо. Також існують вебзастосунки, які надають доступ до бази різних джерел, але мають погано організований пошук, незручний інтерфейс, недостатню організацію кабінету користувачів тощо.

Для розроблення вебзастосунку електронної бібліотеки необхідно визначити особливості та недоліки наявних рішень, щоб забезпечити якість та конкурентоспроможність розробки. Також необхідно провести аналіз потреб та очікувань користувачів, зокрема визначити сценарії використання та функціональні вимоги вебзастосунку.

Отже, проведення аналізу предметної області є необхідним етапом у розробленні вебзастосунку електронної бібліотеки. Цей етап забезпечить високу якість та конкурентоспроможність розробки.

## 1.2. Аналіз існуючих застосунків електронної бібліотеки

При аналізі існуючих вебзастосунків електронної бібліотеки були розглянуті застосунки «Scribd», «Google Books», «Goodreads», «Kobo», «Apple Books».

*Scribd*

Scribd є вебзастосунком, який надає доступ до електронних книг, аудіокниг та журналів. Додаток надає користувачам можливість шукати та читати книги різних жанрів, зокрема художню літературу, бізнес-літературу, наукову літературу та інше. Застосунок є масштабованим для перегляду та читання літературних джерел на будь-яких пристроях. Крім того, Scribd надає можливість читати книги офлайн, що дозволяє користувачам мати доступ до своєї електронної бібліотеки у будь-який момент часу. У застосунок вбудовані інструменти додавання книг до

персоналізованих бібліотек користувачів, створення списків для подальшого перегляду та інше.

*Google Books*

Google Books є вебзастосунком, який надає доступ до електронних книг різних авторів та жанрів. У застосунку пошук літературних джерел організовано за назвою, автором або ключовими словами. Крім того, Google Books надає можливість читати книги онлайн або завантажувати їх на свій пристрій для подальшого читання офлайн. Користувачі застосунку мають можливість створювати власну електронну бібліотеку та додавати до неї книги для подальшого перегляду.

*Goodreads*

Goodreads є вебзастосунком, який надає можливість користувачам шукати книги, оцінювати їх та залишати відгуки. Крім того, за допомогою Goodreads користувачі можуть створювати списки книг, які вони хочуть прочитати та списки книг, які є прочитаними. Застосунок також надає рекомендації користувачам на основі їхніх попередніх читань та оцінок книг. Goodreads дозволяє користувачам ділитися своїми списками та відгуками з іншими користувачами, що дозволяє створювати спільноти читачів та обговорювати книги.

*Kobo*

Kobo є застосунком електронної бібліотеки, який дозволяє користувачам читати книги різних жанрів та авторів. Додаток має понад 6 мільйонів книг, доступних для придбання та завантаження. Крім того, застосунок Kobo працює у двох режимах, зокрема онлайн режим дозволяє користувачам читати книги безпосередньо в додатку, для читання в офлайн режимі необхідно завантажувати електронні книги на пристрої користувачів. Застосунок дозволяє створювати користувацькі електронні бібліотеки. Kobo має інструменти організовувати зберігання книг за категоріями, а також зберігати позначки та замітки на сторінках книг.

*Apple Books*

Apple Books є застосунком електронної бібліотеки, який надає доступ до широкого асортименту книг з різних жанрів та авторів. Додаток дозволяє користувачам шукати книги за назвою, автором або ключовими словами, а також завантажувати їх на свій пристрій для подальшого читання офлайн. Apple Books має вбудовані інструменти для позначок та заміток на сторінках книг, а також можливість підсвічувати текст та налаштовувати розмір шрифту для зручного читання. Крім того, додаток дозволяє користувачам створювати власні колекції книг та ділитися ними з іншими користувачами.

## 1.3. Актуальність розроблення вебзастосунку електронної бібліотеки

Результати аналізу існуючих вебзастосунків електронних бібліотек дозволили виділити основні недоліки. Уникнення цих недоліків допоможе розробити якісний вебзастосунок, який зацікавить багато споживачів. Наведемо недоліки існуючих вебзастосунків.

*Scribd*

Одним з основних недоліків застосунку Scribd є обмежена кількість книг, які можуть бути доступні для прочитання за місячну підписку. Це означає, що користувачі не завжди зможуть знайти потрібну книгу і будуть змушені шукати її в інших джерелах. Scribd також не дозволяє завантажувати книги у форматі PDF або EPUB, що може бути незручним для деяких користувачів. Також додаток має платну підписку розміром у 10 доларів за один місяць.

*Google Books*

Попри те, що застосунок Google Books містить велику базу літературних джерел, деякі книги не є повністю доступні для читання. Книги можуть мати обмеження на кількість сторінок, які можна прочитати або можуть бути доступні тільки для перегляду в деяких країнах. Крім того, інтерфейс Google Books може бути дещо заплутаним для новачків і

користувачам може знадобитися трохи часу, щоб зрозуміти, як організовується пошук та прочитання літератури в застосунку.

*Goodreads*

Основним недоліком застосунку Goodreads є те, що він непередбачений саме для читання, лише для залишення відгуків до літературних джерел. Також Goodreads не завжди надає достатньо повну інформацію про книги, і користувачі можуть бути змушені шукати додаткову інформацію в інших ресурсах.

*Kobo*

Застосунок Kobo має відносно повну базу літературних джерел, але для отримання до неї доступу, потрібно заплатити, інколи навіть трохи завищену ціну. Також існують обмеження у доступі до деяких книг через авторські права або територіальні обмеження. Інтерфейс застосунку може бути незручним для деяких користувачів, зокрема через неінтуїтивну навігацію та складність пошуку літературних джерел.

*Apple Books*

Головним недоліком додатку є обмежений доступ, оскільки Apple Books можна використовувати виключно на присторях виробництва Apple. Застосунок Apple Books вимагає використання формату файлів ePub для електронних книг, що може бути обмеженням для користувачів, які мають електронні книги в інших форматах, таких як MOBI або PDF.

Як можемо побачити жоден із вебзастосунків не є ідеальним та має свої недоліки та обмеження функціональних можливостей. Порівняння функціональних можливостей існуючих застосунків електронних бібліотек наведено в табл. 1.1. У більшості застосунків не передбачено введення архіву, а також статистичний аналіз прочитаних та рекомендованих літературних джерел, який може бути доступним у кабінетах користувачів.

Таблиця 1.1

Порівняння функціональних можливостей існуючих застосунків

| Критерії порівняння | Scribd | Google Books | Good reads | Kobo | Apple Books |
|---|---|---|---|---|---|
| Можливість читання літератури без завантаження | + | + | – | + | + |
| Можливість завантажувати літературу для читання | + | + | – | + | + |
| Аудіочитання | + | – | – | + | + |
| Можливість формування списків прочитаної/бажаної літератури | + | + | + | + | + |
| Можливість коментувати літературу користувачами | – | – | + | + | + |
| Можливість рекомендувати літературу на основі прочитаної | – | – | + | – | – |
| Можливість статистичного аналізу прочитаної літератури для користувачів | – | – | + | – | – |
| Наявність безкоштовних літературних джерел | – | + | + | + | – |

Відповідно до проведеного аналізу певних програмних рішень, розроблений вебзастосунок електронної бібліотеки повинен мати наступні переваги:

1. Швидкий та зручний доступ до відносно повної бази літературних джерел в режимах онлайн та офлайн.
2. Персоналізовану систему рекомендацій літератури на основі попередніх виборів та оцінок користувачів.

3. Інструменти організації бібліотек користувачів на основі прочитаної та/або бажаної для читання літератури.
4. Інструменти для аудіочитання.
5. Інструменти для коментування та оцінювання різних джерел зареєстрованими користувачами.
6. Інструменти для ведення статистичного аналізу прочитаної літератури користувачами, яка є доступною в особистих кабінетах користувачів.
7. Розроблений застосунок є безкоштовним.

Функціональними можливостями розробленого вебзастосунку електронної бібліотеки є:

1. Пошук літератури за категоріями, назвою, авторами, ключовими словами тощо.
2. Візуалізація інформації про джерела літератури, зокрема автори, категорії, кількість сторінок, відгуки, оцінки, рейтинг та інше.
3. Наявність персональних кабінетів користувачів, який містить інформацію щодо обраних книжок, оцінки, інформацію про авторизацію та інше.
4. Можливість створення користувацьких списків літератури, наприклад, список "Те, що я хочу прочитати", "Те, що я вже прочитав", "Мої улюблені книжки" та інше.
5. Наявність інструментів для статистичного аналізу та візуалізації результатів у кабінетах користувачів з можливістю поширення цієї інформації.

## 1.4. Загальні вимоги до вебзастосунку

У результаті аналізу ринку вебзастосунків електронних бібліотек були сформовані функціональні вимоги, нефункціональні вимоги, а також вимоги до безпеки використання застосунку та захисту даних.

Розроблений вебзастосунок електронної бібліотеки має два рівні користувачів, зокрема саме користувацький та адміністративний. Користувацький рівень призначений для реєстрації, вибору та читання літературних джерел, коментування тощо. Адміністративний рівень призначений для системи керування користувачами та літературними джерелами, моніторингу чистоти коментування книг та інше.

Функціональні вимоги для рівня користувача:

– користувач може зареєструватись та авторизуватись в системі за допомогою електронної пошти;
– користувач може переглядати базу літературних джерел;
– користувач може виконувати пошук у базі літературних джерел за категоріями (назва, автори, жанри тощо);
– користувач може читати літературні джерела в онлайн та офлайн режимах;
– користувач може читати книги в аудіоформаті;
– користувач може здійснювати пошук літературних джерел за назвою, авторами, ключовими словами або фразами;
– користувач може організовувати літературні джерела у кабінеті за наступними категоріями «Улюблені», «Хочу прочитати», «Читаю», «Прочитав»;
– користувач може оцінювати літературні джерела та залишати коментарі до них;
– користувач може отримувати рекомендації на основі власних вподобань;
– користувач може отримувати власну статистику щодо прочитаних матеріалів.

Функціональні вимоги для рівня адміністратор:

– адміністратор може додавати нові матеріали до електронної бібліотеки, редагувати та видаляти матеріали;

– адміністратор може керувати коментарями, які залишають користувачі, зокрема видаляти повідомлення з нецензурною лексикою;
– адміністратор може керувати користувачами системи, зокрема, блокувати користувачів, які порушують правила використання застосунку.

Нефункціональні вимоги:

– застосунок повинен бути доступним на різних пристроях з доступом до Інтернету та працювати для різних веббраузерах (Chrome, Firefox, Safari тощо);
– застосунок повинен забезпечувати безпеку даних користувачів, зокрема, захист від несанкціонованого доступу, зламу, викрадення та втрати даних;
– застосунок повинен бути інтуїтивно зрозумілим та зручним у використанні для користувачів різного рівня комп'ютерної грамотності;
– застосунок повинен мати зручну та легку в обслуговуванні інтерфейсну частину, яка дозволить адміністратору з легкістю додавати, редагувати та видаляти матеріали.

## 1.5. Вимоги до безпеки застосунку

Щоб захистити дані користувачів та запобігти будь-яким зловмисним діям, спрямованим на компрометацію доступності програми, дуже важливо виконати ретельну оцінку можливих ризиків, які можуть виникнути під час використання програмного забезпечення.

Аналіз ризиків зауважив, що потенційні наслідки неправильного використання застосунку або виникнення неочікуваних збоїв сервера є досить значними умовами, що спричиняють виникненню загроз безпеки використання вебзастосунку. З усіх потенційних небезпек і недоліків було

виділено декілька умов, які могли поставити під загрозу безпеку та безперебійну роботу вебзастосунку. Серед цих умов можна виділити:

– неправильна або відсутня валідація даних користувачів;

– відсутність попередньої обробки запиту до бази даних;

– збереження даних без попереднього шифрування;

– відсутність резервного копіювання бази даних.

У табл. 1.2 представлений детальний опис наслідків загроз безпеки даних, які можуть впливати на роботу вебзастосунку.

Таблиця 1.2

Наслідки безпекових загроз вебзастосунку електронної бібліотека

| № | Уразливість | Загроза | Наслідки |
|---|---|---|---|
| 1 | Неправильна або відсутня валідація даних користувачів | Вхід з невалідними даними в систему | Перевищення повноважень. Витік конфіденційної інформації. Порушення функціональності. Порушення роботи БД |
| 2 | Відсутність попередньої обробки запиту до бази даних | SQL-ін'єкція в базу даних | Втрата конфіденційної інформації. Внесення змін до бази даних. Знищення даних |
| | | Досягнення відмови в обслуговуванні | Перевантаження системи і унеможливлення її роботи |
| 3 | Збереження даних без попереднього шифрування | Доступ до конфіденційної інформації | Витік конфіденційної інформації |
| 4 | Відсутність резервного копіювання бази даних | Відсутність можливості відновлення даних після випадкового чи зловмисного видалення або пошкодження | Повна втрата даних системи |

Наведені у табл. 1.2 наслідки мають високий рівень впливу на систему. Для унеможливлення настання будь-яких наслідків проведено аналіз та виявлено необхідні заходи безпеки, що описані в табл. 1.3.

Таблиця 1.3

Заходи безпеки вебзастосунку електронної бібліотеки

| № | Уразливість | Загроза | Заходи безпеки |
|---|---|---|---|
| 1 | Неправильна або відсутня валідація користувацьких даних | Вхід з невалідними даними в систему | Використання механізмів валідації даних |
| 2 | Відсутність попереднього оброблення запиту до бази даних | SQL-ін'єкція у базу даних | Використання спеціальних програмних та апаратних засобів, що фільтрують шкідливий трафік.<br>Використання вбудованих функцій та бібліотек для обробки та валідації введення.<br>Встановлення належних прав доступу до бази даних. |
| | | Досягнення відмови в обслуговуванні | |
| 3 | Збереження даних без попереднього шифрування | Доступ до конфіденційної інформації | Шифрування даних в системі |
| 4 | Відсутність резервного копіювання бази даних | Відсутність можливості відновлення даних після випадкового чи зловмисного видалення або пошкодження | Резервне копіювання бази даних |

Таким чином, з метою забезпечення безпечного та безперебійного використання вебзастосунку електронної бібліотеки було визначено безпекові загрози, їх наслідки для роботи системи та необхідні заходи безпеки, реалізація яких сприятиме запобіганню цим проблемам.

## 1.6. Висновки до розділу

Сучасний інформаційний ринок є достатньо широким. Зокрема використання застосунків електронних бібліотек стає популярним та стрімко зростає серед користувачів всього світу. Звичайно, випущені вебзастосунки мають свої особливості, а також обмеження та недоліки.

Метою розроблення вебзастосунку електронної бібліотеки є забезпечення користувачів доступом до електронного сховища літературних джерел та інструментами організації цієї інформації.

Вебзастосунок електронної бібліотеки надає онлайн та офлайн (PDF та MP3 формати) доступ до багатьох електронних книг. Розроблений вебзастосунок надає можливість здійснювати пошук джерел за різними категоріями, зокрема жанри, автори та ключові слова. Крім того, користувачі можуть складати власні списки книг та додавати до них книги з бібліотеки для подальшого читання. Нарешті, застосунок забезпечує статистикою читання, що дозволяє відстежувати користувачам власні досягнення та прогрес.

Розроблений вебзастосунок має два рівні користувачів та є доступним для різних пристроїв з доступом до Інтернету.

У розділі були визначені функціональні вимоги для рівнів користувачів та адміністраторів, а також нефункціональні вимоги. Реалізація цих вимог забезпечить якісний та зручний досвід використання вебзастосунку.

У розділі описані результати аналізу безпекових загроз застосунку та описані розроблені відповідні заходи безпеки, що забезпечить безперебійну та безпечну роботу програмного забезпечення.

# 2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

## 2.1. Обґрунтування вибору мови програмування

Щоб обрати мову програмування серверної частини, було вирішено зробити аналіз існуючих мов програмування, розглянути їх переваги та недоліки.

*Python*

Переваги та недоліки мови програмування Python наведено у табл. 2.1 [1]. Найбільше Python використовують для роботи з data science, штучним інтелектом, аналізом даних, візуалізацією даних тощо. Також його фреймворки широко використовуються у веброзробленні [2].

Таблиця 2.1

Переваги та недоліки мови програмування Python

| Переваги | Недоліки |
|---|---|
| Легкий у вивченні та використанні: Python є одним з найпростіших мов програмування для вивчення та використання через його максимально зрозумілий синтаксис та простоту написання. | Продуктивність: Python менш продуктивний у порівнянні з C++ та Java. Це може знизити показники швидкодії для великих програм. |
| Універсальність: Python можна використовувати для широкого спектра програм, зокрема веброзроблення, аналізу даних, технологій штучного інтелекту та інше. | Global Interpreter Lock: Python може обмежити здатність масштабувати багатопотокові програми. Тобто може бути важче скористатись перевагами багатоядерних процесорів. |

Продовження табл. 2.1

| | |
|---|---|
| Велика та активна спільнота: Python має велику та активну спільноту розробників. Отже, існує багато відкритих бібліотек та фреймворків для розроблення різних застосунків. Ця спільнота забезпечує підтримку іншим розробникам. | Споживання пам’яті: Python може споживати великий обсяг пам’яті, що є проблемою в програмах з обробленням великих обсягів даних. |
| Кросплатформенність: Python може бути запущений на багатьох операційних системах, зокрема Windows, macOS та Linux. | Недостатньо потужний для мобільних обчислень: Python не широко використовується в обчисленнях мобільних додатків. |
| Швидкість та продуктивність: простота та легкість використання Python обумовлює невеликий термін розроблення та підвищення продуктивності. | Помилки під час виконання: Python є інтерпретованою мовою, тобто деякі помилки можуть бути виявлені тільки під час виконання. Це може ускладнити налагодження. |

Найбільше Python використовують у роботі з data science, штучним інтелектом, аналізом даних, візуалізацією даних тощо. Також його фреймворки широко використовуються у веброзробленні [2].

*Java*

Переваги та недоліки мови програмування Java наведено у табл. 2.2 [3]. Java широко використовується у розробленні відеоігор, мобільних застосунків, десктопних застосунків та для роботи з великими даними. Також її часто використовують для розроблення вебзастосунків

корпоративного рівня, оскільки Java має відповідні потужні фреймворки з широкими можливостями [4].

Таблиця 2.2

Переваги та недоліки мови програмування Java

| Переваги | Недоліки |
|---|---|
| Незалежність від платформи: Java є незалежною від платформи мовою, тобто код мовою Java може працювати на будь-якій платформі без необхідності повторної компіляції. Це робить Java чудовим вибором для створення кросплатформних програм. | Продуктивність: Java може працювати повільніше в порівнянні з C++ та C#. Це може бути проблемою в програмах, які потребують високошвидкісного оброблення або продуктивності в реальному часі. |
| Об’єктно-орієнтована мова: Java є об’єктно-орієнтованою мовою, що означає, що вона заохочує модульне програмування та полегшує повторне використання коду. Це може призвести до більш зручного обслуговування та масштабування коду. | Споживання пам’яті: Java може споживати багато пам’яті, що може бути проблемою в програмах, яким потрібно обробляти великі обсяги даних. |
| Велика та активна спільнота: Java має велику та активну спільноту розробників. Тобто існує багато бібліотек та фреймворків, доступних для допомоги в розробленні. | Багатослівність: програмний код Java може бути багатослівним та вимагати більше рядків коду в порівнянні з іншими мовами програмування для досягнення однакової функціональності. |

Продовження табл. 2.2

| | |
|---|---|
| | Це може зробити написання та підтримку коду більш трудомістким. |
| Безпека: Java має вбудовані функції безпеки, які можуть допомогти запобігти поширеним уразливостям безпеки, зокрема переповнення буфера та неавторизований доступ до системних ресурсів. | Крута крива навчання: Java може бути важко вивчити, особливо для початківців через її складність та кількість концепцій, які потрібно зрозуміти. |
| Масштабованість: Java – це мова з високою масштабованістю, яку можна використовувати для створення великих програм корпоративного рівня. | Застарілий код: існує багато застарілого програмного коду на Java, який нелегко підтримувати чи оновлювати. |

Переваги та недоліки мови програмування JavaScript наведено у табл. 2.3 [5]. JavaScript є високорівневою, інтерпретованою мовою програмування, яка використовується для створення динамічного вебконтенту. Вона є однією з найпопулярніших мов програмування у світі і широко застосовується для розробки вебдодатків, включаючи взаємодію з користувачем, маніпуляцію даними та створення анімації. Проте JavaScript вважається мовою для клієнта, тому її частіше використовують у фронтенді, але вона має фреймворки, які надають змогу використовувати її для розроблення серверної частини. Також її використовують для розроблення мобільних та декстопних додатків, ігор [6].

Таблиця 2.3

Переваги та недоліки мови програмування JavaScript

| Переваги | Недоліки |
|---|---|
| Інтерактивність на стороні клієнта: JavaScript використовується для створення сценаріїв на стороні клієнта, що дозволяє створювати інтерактивні та динамічні вебсторінки. JavaScript можна використовувати для створення анімації, перевірки введених користувачем даних та зміни вмісту вебсторінки без необхідності її перезавантаження. | Ризики безпеки: JavaScript можна використовувати для виконання шкідливого коду, який може становити загрозу безпеці вебдодатків. |
| Популярність: JavaScript є однією з найпопулярніших мов програмування, яка підтримується всіма сучасними веббраузерами. | Проблеми сумісності браузера: різні браузери можуть по-різному інтерпретувати код JavaScript, що може призвести до проблем сумісності. |
| Велика спільнота розробників: існує велика спільнота розробників, яка надає підтримку та ресурси для JavaScript, включаючи бібліотеки, фреймворки та інструменти. | Продуктивність: JavaScript є інтерпретованою мовою, тобто є не швидкою в порівнянні з C++ або Java. |
| Гнучкість: JavaScript – це гнучка мова, яку можна використовувати для зовнішнього та внутрішнього розроблення. За допомогою Node.js | Налагодження: налагодження JavaScript може бути складним через його динамічну природу, |

Продовження табл. 2.3

| | |
|---|---|
| JavaScript також можна використовувати для розроблення серверного модуля. | що може призвести до помилок виконання, які важко визначити. |
| Сумісність: JavaScript можна легко інтегрувати з іншими мовами програмування. Також існують API для інтеграції коду з іншим програмним забезпеченням. | Відсутність перевірки типів: JavaScript є мовою з динамічним типом, що означає, що він може бути схильний до помилок, спричинених типами даних. |

Для розроблення клієнтської частини, було вирішено зробити аналіз способів реалізації клієнтської частини та розглянути відповідні мови програмування, а також їх переваги та недоліки.

*HTML*

HTML (Hypertext Markup Language) – це мова розмітки, мова програмування, яка описує структуру та вміст вебсторінки. HTML надає набір тегів розмітки, які можна використовувати для визначення структури, макета та вмісту вебсторінки [7].

HTML можна використовувати для створення різноманітного вмісту, зокрема текст, зображення, відео та гіперпосилання. Він також надає можливість структурувати вміст за допомогою заголовків, абзаців, списків, таблиць та інших елементів.

*CSS*

CSS – мова програмування для оформлення будь яких вебсторінок. Вона використовується для опису презентації або візуального вигляду вебсторінок або вебдодатків, написаних на мовах розмітки HTML або XML [8]. CSS дозволяє розробникам визначати, як має відображатися вміст

вебсторінки, зокрема макети, кольори, шрифти та інші стилістичні елементи. За допомогою CSS розробники можуть відокремити рівень презентації від рівня вмісту, полегшуючи підтримку та оновлення вебсторінок та вебдодатків.

*TypeScript*

TypeScript – це мова програмування з відкритим кодом, яка є надмножиною JavaScript. TypeScript був розроблений Microsoft та випущений у 2012 році. TypeScript надає додаткову статичну типізацію, класи, інтерфейси та інші функції, яких немає в JavaScript [9]. Код TypeScript транспілюється в JavaScript, який потім можна виконувати у веббраузерах або середовищі на стороні сервера. TypeScript можна використовувати для клієнтського та серверного розроблення. TypeScript стає все більш популярним для веброзроблення завдяки численним перевагам. Розглянемо ці переваги нижче.

Безпека типів: TypeScript надає необов'язкову статичну типізацію, яка дозволяє розробникам виявляти пов'язані з типом помилки під час компіляції, а не під час виконання. Це полегшує виявлення помилок на ранніх стадіях процесу розроблення, що може заощадити час та зменшити кількість помилок.

Підтримка коду: TypeScript полегшує написання підтримуваного коду, надаючи такі функції, як класи, інтерфейси та модулі. Це може допомогти зробити код більш упорядкованим та зрозумілим.

Інструментарій: TypeScript має чудову підтримку інструментів, включаючи IDE, зокрема Visual Studio Code та WebStorm, які надають ряд функцій, наприклад автозаповнення, перевірка помилок та налагодження.

Підтримка спільноти: TypeScript має велику та активну спільноту розробників, а це означає, що розробникам доступно багато ресурсів і бібліотек.

Нижче розглянемо недоліки TypeScript.

Крива навчання: TypeScript має крутішу криву навчання, ніж JavaScript, особливо для розробників, які не знайомі зі статичним типом.

Додаткова складність: TypeScript додає додатковий рівень складності до розроблення, що може ускладнити написання та підтримку коду.

Час компіляції: код TypeScript потрібно скомпілювати в JavaScript, перш ніж його можна буде виконати, що може сповільнити термін розроблення [10].

Отже, для створення вебзастосунку електронної бібліотеки було обрано програмні засоби Spring Boot та Angular. Нижче наведемо обґрунтування вибору.

*Spring Boot*

Spring Boot – це популярний фреймворк на основі Java, який забезпечує спрощене та швидке середовище розроблення для створення вебдодатків. Переваги Spring Boot описані нижче.

Потужна підтримка спільноти: у Spring Boot є велика спільнота підтримки, яка надає корисні ресурси, навчальні посібники та форуми, щоб допомогти розробникам швидко вирішити проблеми. Ця підтримка спільноти гарантує, що розробники мають доступ до найновіших інструментів, методів і найкращих практик для створення високоякісних вебдодатків.

Швидке розроблення: Spring Boot забезпечує спрощене та швидке середовище розроблення для створення вебдодатків. Автоматична конфігурація Spring Boot та початкові залежності спрощують швидке налаштування нового проєкту та одразу починають роботу зі створення функціональних можливостей.

Потужні функції безпеки: безпека має вирішальне значення для будь-якого вебзастосунку, особливо для електронної бібліотеки, де дані користувача потребують захисту. Spring Boot надає надійний набір функцій

безпеки, які можуть допомогти забезпечити безпеку даних користувача, наприклад шифрування, контроль доступу та автентифікацію.

Архітектура мікросервісів: Spring Boot – це ідеальна структура для створення програм із архітектурою мікросервісів. Ця архітектура дозволяє розробникам розбивати програму на менші, більш керовані компоненти, що полегшує масштабування та підтримку програми з часом.

Легка інтеграція з іншими технологіями: Spring Boot можна легко інтегрувати з іншими технологіями, зокрема базами даних, системи обміну повідомленнями та API для створення надійного інтегрованого вебзастосунку.

Добре задокументований та зрілий фреймворк: Spring Boot існує вже понад десяти років і широко використовується розробниками в усьому світі. Це означає, що фреймворк добре задокументований, зрілий та має перевірений досвід створення високоякісних вебдодатків [11].

Загалом, Spring Boot надає надійний набір інструментів і функцій, які роблять його чудовим вибором для створення сучасної, масштабованої та безпечної електронної бібліотеки. Його сильна підтримка спільноти, швидке середовище розроблення, архітектура мікросервісів та потужні функції безпеки роблять його ідеальною платформою для створення високоякісних вебдодатків [12].

*Angular*

Angular – це фреймворк з відкритим вихідним кодом на основі TypeScript, який використовується для створення односторінкових програм і надає надійний набір інструментів і бібліотек для створення складних вебзастосунків. Переваги Angular описані нижче.

Розширений інтерфейс користувача: Angular – це потужна інтерфейсна платформа JavaScript, яка забезпечує розширений інтерфейс користувача та покращений досвід роботи з користувачем. Angular надає такі функції, як зв’язування даних, компоненти, директиви та служби, які дозволяють розробникам створювати динамічні та інтерактивні вебдодатки.

Кросплатформна розробка: Angular – це кросплатформна структура, яку можна використовувати для розроблення вебдодатків для настільних, мобільних та інших платформ. Angular забезпечує вбудовану підтримку адаптивного вебдизайну, яка гарантує, що програма добре працює на різних пристроях та розмірах екрана.

Простий у використанні: Angular розроблено таким чином, щоб його було легко освоїти та використовувати навіть для розробників, які починають займатись веброзробленням. Angular забезпечує чіткий і стислий синтаксис, який полегшує читання та розуміння коду.

Велика спільнота та активна розробка: Angular має велике та активне співтовариство розробників, які роблять внесок у структуру та надають корисні ресурси та інструменти. Спільнота надає велику кількість навчальних посібників, документації та форумів, які допомагають розробникам вирішувати проблеми та бути в курсі останніх тенденцій і найкращих практик.

Відмінна підтримка для тестування: тестування є важливою частиною створення високоякісних вебдодатків, і Angular забезпечує чудову підтримку для тестування. Angular надає набір інструментів і фреймворків для тестування, включаючи модульне тестування, наскрізне тестування та інтеграційне тестування.

Покращена продуктивність: Angular надає кілька функцій, які можуть покращити продуктивність вебдодатків, наприклад відкладене завантаження, струшування дерева та випереджаюча компіляція. Ці функції можуть допомогти скоротити час завантаження та покращити загальну продуктивність програми.

Проста інтеграція з backend технологіями: Angular можна легко інтегрувати з різними backend технологіями, зокрема Spring Boot, що може спростити процес розроблення та покращити загальну якість програми [13].

Загалом, Angular надає потужний набір функцій, які роблять його чудовим вибором для створення сучасного, адаптивного та інтерактивного

вебзастосунку електронної бібліотеки. Його багатий інтерфейс користувача, кросплатформне розроблення, простота використання, велика спільнота та активне розроблення, відмінна підтримка тестування, покращена продуктивність і легка інтеграція з серверними технологіями роблять його ідеальною платформою для створення високоякісних вебдодатків [14].

## 2.2. Обґрунтування вибору бази даних

Для реалізації бази даних для вебзастосунку електронної бібліотеки було обрано PostgreSQL. PostgreSQL – це потужна система керування реляційними базами даних (RDBMS), яка широко використовується для зберігання та керування структурованими даними. PostgreSQL підтримує широкий спектр мов програмування та надає драйвери для багатьох популярних мов програмування, що полегшує інтеграцію з різними стеками програми [15]. Нижче описані особливості бази даних PostgreSQL.

Відкритий вихідний код: PostgreSQL – це система керування базами даних із відкритим кодом, що означає, що її можна вільно використовувати, змінювати та поширювати.

Надійність та стабільність: PostgreSQL відомий своєю надійністю і стабільністю. Він має багаторічний досвід роботи в критично важливих програмах із високим трафіком. PostgreSQL робить сильний акцент на цілісність даних, що означає меншу ймовірність втрати або пошкодження даних.

Відповідність ACID: PostgreSQL повністю сумісний з ACID, що означає, що він надає гарантії атомарності, узгодженості, ізоляції та довговічності. Це робить його придатним вибором для програм, які вимагають узгодженості транзакцій і цілісності даних.

Масштабованість: PostgreSQL має високу масштабованість та може обробляти великі обсяги даних, іншими словами великий трафік. Він надає такі функції, як розділення таблиць та багатоверсійний контроль

паралельності (MVCC), які дозволяють обробляти великі набори даних при високому рівні паралельності.

Розширюваність: PostgreSQL дуже розширюваний і може бути налаштований відповідно до конкретних вимог. Він забезпечує підтримку процедурних мов, зокрема PL/pgSQL та PL/Python, які дозволяють розробникам писати складні функції та процедури.

Гнучкість: PostgreSQL є гнучким і підтримує широкий діапазон типів даних, включаючи текстові, числові, дата, час та JSON. Він також забезпечує підтримку складних структур даних, таких як масиви, hstore та XML.

Підтримка спільноти: PostgreSQL має велику та активну спільноту розробників і користувачів, які надають підтримку, документацію та інструменти. Це означає, що розробники можуть легко знайти допомогу та ресурси, коли вони стикаються з проблемами або потребують навчання новим навичкам [16].

Загалом, PostgreSQL – це надійна, масштабована та гнучка система керування базами даних із відкритим кодом, яка забезпечує відповідність ACID, розширюваність та потужну підтримку спільноти. Ці функції роблять його гарним вибором для вебзастосунку електронної бібліотеки, де цілісність даних, масштабованість і гнучкість є вирішальними.

## 2.3. Висновки до розділу

Після аналізу різних технологій для розробки масштабованого та надійного вебзастосунку електронної бібліотеки було вирішено використовувати фреймворки Spring Boot для backend розроблення та Angular для frontend розроблення. Spring Boot пропонує ряд функцій, зокрема проста конфігурація, ін’єкція залежностей та підтримка вебсервісів RESTful, тоді як Angular надає такі функції, як двостороннє зв’язування даних, ін’єкція залежностей та підтримка реактивного програмування.

Для системи керування базами даних було вирішено використовувати PostgreSQL завдяки підтримці SQL, керування транзакціями та цілісності даних. Він також забезпечує високу продуктивність, масштабованість і надійність, які є важливими для програми електронної бібліотеки, яка потребує великого обсягу зберігання та пошуку даних.

Отже, інтегрування Spring Boot, Angular та PostgreSQL забезпечує потужну та масштабовану архітектуру для створення вебзастосунку електронної бібліотеки, яка може обробляти велику кількість даних, а також користувачів, забезпечуючи безпеку, надійність і простоту використання.

# 3. РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ ЕЛЕКТРОННОЇ БІБЛІОТЕКИ

## 3.1. Загальний опис системи вебзастосунку

Користувачами вебзастосунку електронної бібліотеки можуть бути дорослі читачі, діти, науковці, фахівці будь-який галузей тощо. Також додатком повинні оперувати адміністратори, які виконують необхідні обов'язки перед джерелами літератури, користувачами, їх коментарями тощо.

Вебзастосунок має ієрархічну структуру, в якому виділено такі ролі:

– користувач;

– адміністратор.

Адміністратор виконує відповідні задачі на своєму спеціальному обліковому записі, тому для нього не розглядаються функціональні можливості користувача. Обліковий запис адміністратора може бути зареєстровано вручну через базу даних.

Адміністратор застосунку може виконувати наступні дії:

1. Створення, редагування, видалення літературних джерел.
2. Керування посиланнями на ресурси літературних джерел.
3. Керування користувачами (блокування/розблокування).
4. Перегляд та видалення коментарів, якщо вони порушують правила додатку.

Функціональні можливості адміністратора вебзастосунку описано на рис. 3.1.

Користувач вебзастосунку може бути авторизованим або незареєстрованим. У незареєстрованого користувача є наступні функціональні можливості:

1. Перегляд головної сторінки вебзастосунку.
2. Пошук літературних джерел, зокрема за допомогою інструментів пошуку за категоріями, фільтрів пошуку тощо.

3. Перегляд сторінок з інформацією про конкретні літературні джерела.
4. Авторизація та реєстрація.

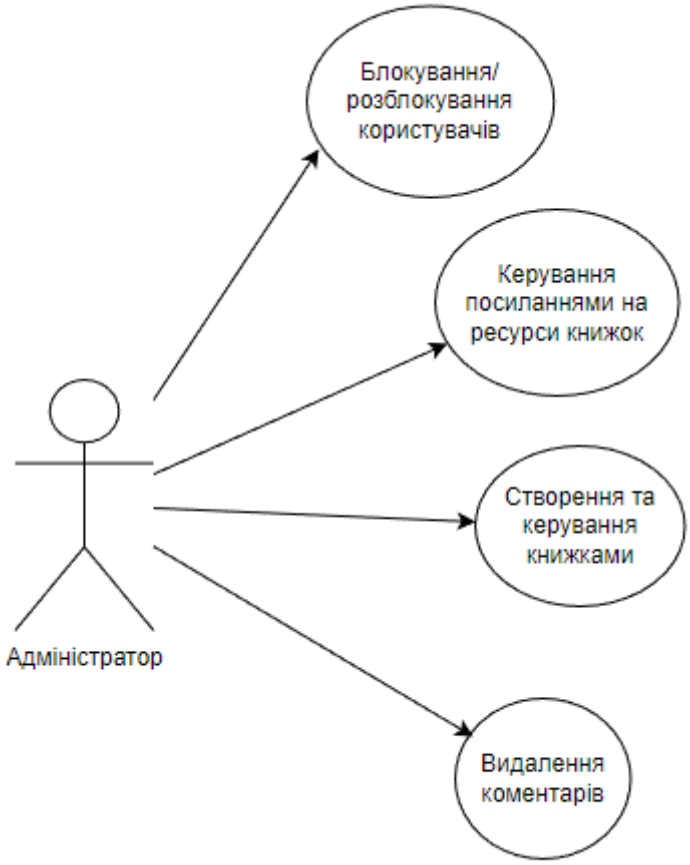



Рис. 3.1. Діаграма функціональних можливостей адміністратора системи

Авторизований користувач має можливість використовувати всі функціональні можливості застосунку, окрім адміністративного, а саме:

1. Перегляд головної сторінки застосунку.
2. Пошук літературних джерел за жанрами, авторами або назві.
3. Перегляд сторінки з інформацією та відгуками про літературні джерела.
4. Додавання літературних джерел до своєї бібліотеки.
5. Перегляд своєї бібліотеки та списків.

6. Організація своїх списків за категоріями «Улюблені», «Хочу прочитати», «Читаю», «Прочитав».
7. Налаштування власного облікового запису.
8. Перегляд власної статистики прочитаних книг.
9. Оцінка літературних джерел.
10. Коментування літературних джерел
11. Читання літературних джерел.
12. Прослуховування літературних джерел
13. Отримання та перегляд рекомендацій на основі власних вподобань.
14. Вихід з облікового запису.

Функціональні можливості авторизованого користувача вебзастосунку наведено у діаграмі функціональних можливостей на рис. 3.2.

Неавторизований користувач має можливість використовувати обмежені функціональні можливості вебзастосунку, а саме:

1. Перегляд головної сторінки застосунку.
2. Пошук літературних джерел по жанрам, авторам або назві.
3. Перегляд сторінки з інформацією та відгуками про літературні джерела.
4. Авторизація
5. Реєстрація.

Функціональні можливості неавторизованого користувача вебзастосунку наведено у діаграмі функціональних можливостей на рис. 3.3.

Зазначені функціональні можливості забезпечать користувачів надійним, простим та зручним досвідом використання вебзастосунку електронної бібліотеки.

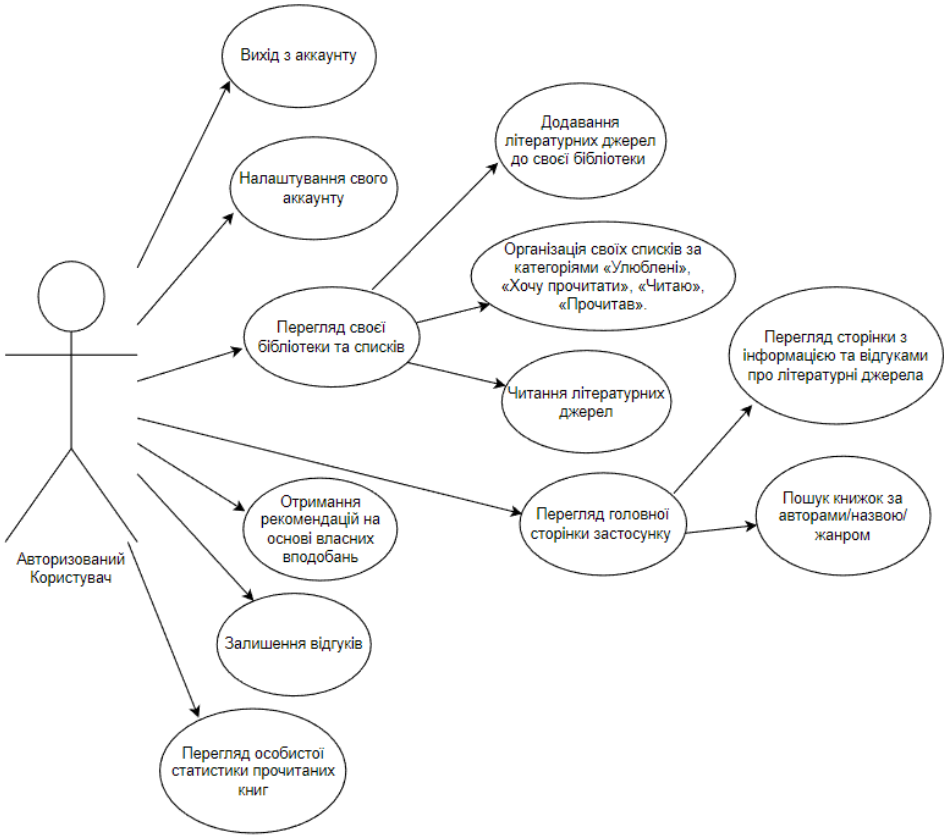



Рис. 3.2. Діаграма функціональних можливостей авторизованого користувача системи

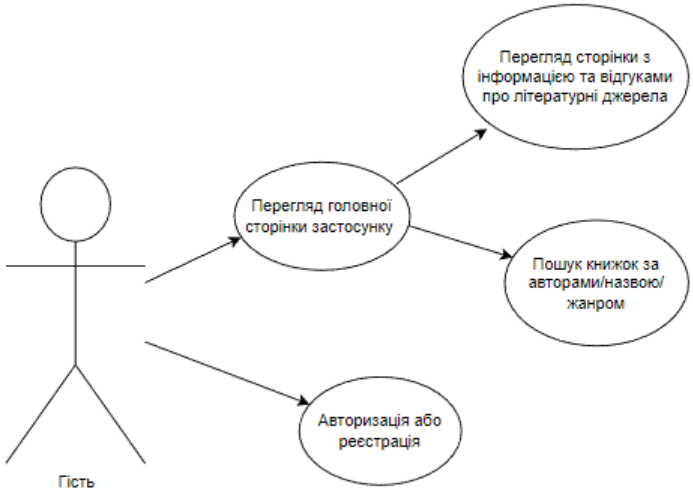



Рис. 3.3. Діаграма функціональних можливостей неавторизованого користувача системи

## 3.2. Структура бази даних

Для роботи з даними при розробленні вебзастосунку були розроблені наступні сутності.

Сутність "account":

1) "id": унікальний ідентифікатор для кожного облікового запису;
2) "username": ім'я користувача для кожного облікового запису;
3) "email": адресу електронної пошти для кожного облікового запису;
4) "password": пароль для кожного облікового запису;
5) "blocked": вказує, чи обліковий запис заблоковано чи ні;
6) "role": роль облікового запису (ADMIN або USER);
7) "firstname": ім'я власника облікового запису;
8) "lastname": прізвище власника облікового запису.

Сутність "account" забезпечує організування даних про обліковий запис користувача.

Сутність "book":

1) "id": унікальний ідентифікатор для кожної книги;
2) "title": назва книги;
3) "authors": автори книги;
4) "genre": жанр книги;
5) "subtitle": підзаголовок книги;
6) "description": опис книги;
7) "published": рік видання книги;
8) "pages": кількість сторінок у книзі.

Сутність "book" забезпечує організування даних про книги.

Сутність "accountBook":

1) "id": унікальний ідентифікатор для кожного запису в таблиці;
2) "FK book_id": зовнішній ключ книги, пов’язаної з обліковим записом;
3) "FK account_id": зовнішній ключ облікового запису, пов’язаного з книгою;

4) "bought": вказує, чи була книга додана до власної бібліотеки власником облікового запису чи ні;
5) "favourite": вказує, чи позначено книгу як вибране власником облікового запису чи ні;
6) "wanted": вказує, чи хоче власник облікового запису купити книгу чи ні;
7) "reading": вказує, чи читає книгу власник облікового запису в даний момент чи ні;
8) "already_read": вказує, чи власник облікового запису вже прочитав книгу повністю чи ні.

Сутність "accountBook" забезпечує організування категорій книг користувачів.

Сутність "link":

1) "id": унікальний ідентифікатор для кожного запису в таблиці;
2) "FK book_id": зовнішній ключ книги, пов’язаної з посиланням;
3) "audio_link": посилання на аудіофайл книги;
4) "pdf_link": посилання на PDF-версію книги;
5) "thumbnail_link": посилання на мініатюру книги.

Сутність "link" забезпечує організування даних посилань на файли.

Сутність "review":

1) "id": унікальний ідентифікатор для кожного відгуку;
2) "FK book_id": зовнішній ключ книги, пов’язаної з відгуком;
3) "FK account_id": зовнішній ключ облікового запису, пов’язаного з відгуком;
4) "review": текст відгуку;
5) "rating": рейтинг, наданий власником облікового запису книзі.

Сутність "review" забезпечує організування даних відгуків користувачів.

Кожна сутність має певні зв’язки з іншими таблицями. Ці зв’язки представлені у табл. 3.1.

Таблиця 3.1

Таблиця зв’язків між сутностями

| № | Сутність | Сутність | Відношення |
|---|---|---|---|
| 1 | account | book | Багато до багатьох |
| 2 | account | accountBook | Один до багатьох |
| 3 | book | accountBook | Один до багатьох |
| 4 | book | link | Один до одного |
| 5 | book | review | Один до багатьох |
| 6 | account | review | Один до багатьох |

Описані сутності та їх зв’язки відповідають структурі бази даних. У табл. 3.2-3.6 наведено детальний опис таблиць кожної сутності та особливості їх атрибутів.

Таблиця 3.2

Таблиця сутності "account"

| accounts | | | | |
|---|---|---|---|---|
| Назва поля | Тип даних | Не може бути NULL | Первинний ключ | Зовнішній ключ |
| id | serial int | так | так | ні |
| username | string | так | ні | ні |
| firstname | string | так | ні | ні |
| lastname | string | так | ні | ні |
| email | string | так | ні | ні |

Продовження табл. 3.2

| | | | | |
|---|---|---|---|---|
| password | string | так | ні | ні |
| blocked | bool | так | ні | ні |
| role | text | так | ні | ні |

Таблиця 3.3

Таблиця сутності "accountsBook"

| accountsBooks | | | | |
|---|---|---|---|---|
| Назва поля | Тип даних | Не може бути NULL | Первинний ключ | Зовнішній ключ |
| id | serial int | так | так | ні |
| book_id | integer | так | ні | так |
| account_id | integer | так | ні | так |
| bought | bool | так | ні | ні |
| favourite | bool | так | ні | ні |
| wanted | bool | так | ні | ні |
| reading | bool | так | ні | ні |
| already_read | bool | так | ні | ні |

Таблиця 3.4

Таблиця сутності "books"

| books | | | | |
|---|---|---|---|---|
| Назва поля | Тип даних | Не може бути NULL | Первинний ключ | Зовнішній ключ |
| id | serial int | так | так | ні |
| title | string | так | ні | ні |
| authors | string | так | ні | ні |
| genre | string | так | ні | ні |
| subtitle | string | ні | ні | ні |
| description | string | так | ні | ні |
| published | integer | так | ні | ні |
| pages | integer | так | ні | ні |

Таблиця 3.5

Таблиця сутності "link"

| links | | | | |
|---|---|---|---|---|
| Назва поля | Тип даних | Не може бути NULL | Первинний ключ | Зовнішній ключ |
| id | serial int | так | так | ні |
| book_id | integer | так | ні | так |

Продовження табл. 3.5

| | | | | |
|---|---|---|---|---|
| audio_link | string | ні | ні | ні |
| pdf_link | string | так | ні | ні |
| thumbnail_link | string | ні | ні | ні |

Таблиця 3.6

Таблиця сутності "review"

| reviews | | | | |
|---|---|---|---|---|
| Назва поля | Тип даних | Не може бути NULL | Первинний ключ | Зовнішній ключ |
| id | serial int | так | так | ні |
| book_id | integer | так | ні | так |
| account_id | integer | так | ні | так |
| review | string | ні | ні | ні |
| rating | integer | ні | ні | ні |

Можемо побачити, що база даних містить 5 таблиць, кожна з яких відповідає розробленій сутності. Як первинний та зовнішній ключ використовувався автоматично-генерований integer. Схематично структура бази даних зображена у вигляді UML-діаграми на рис. 3.3.

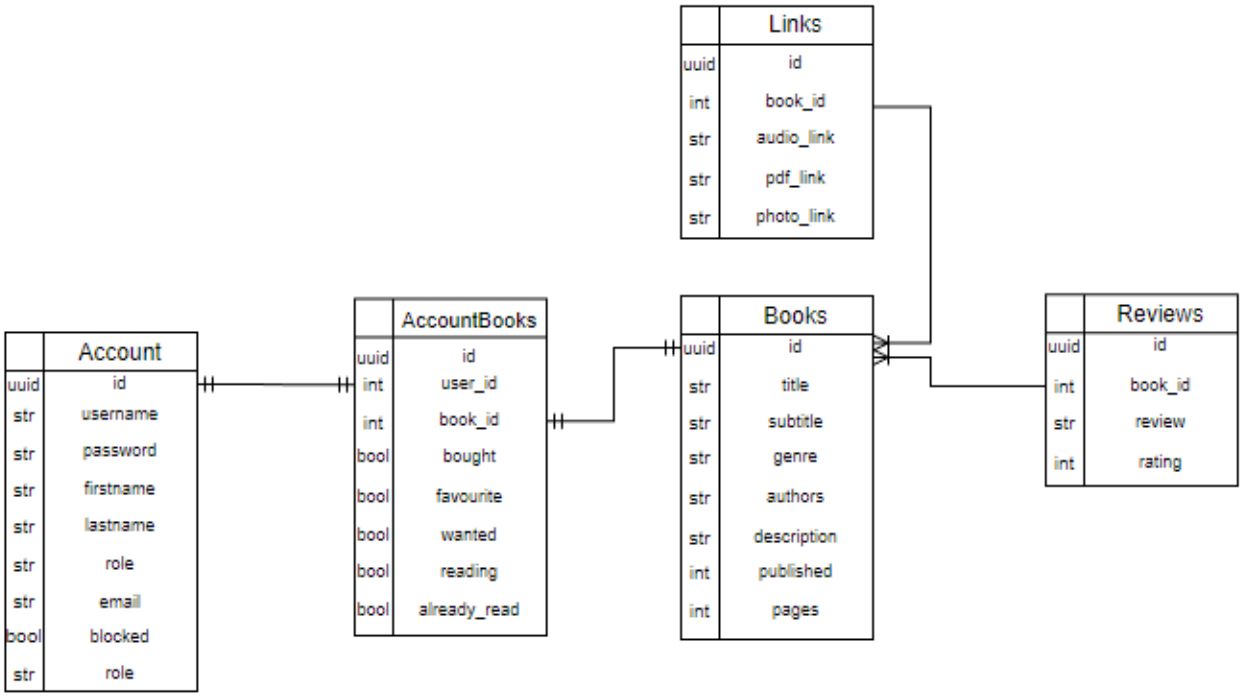



Рис. 3.3. Структура бази даних вебзастосунку

## 3.3. Архітектура системи

Розроблений вебзастосунок електронної бібліотеки має клієнт-серверну архітектуру. У цій архітектурі серверна частина відповідає за надання запитуваних ресурсів або послуг клієнтській частини, а клієнт відповідає за надсилання запитів на сервер. Сервер обробляє запит і повертає відповідь клієнту. Ця модель має кілька переваг перед іншими архітектурами, зокрема масштабованість, надійність і гнучкість.

Однією з головних переваг клієнт-серверної архітектури є те, що вона дозволяє централізоване керування ресурсами та послугами. Це означає, що всі клієнти можуть отримати доступ до тих самих ресурсів і послуг з одного місця, а не дублювати їх на кількох пристроях. Ця централізація також дозволяє спростити обслуговування та оновлення, оскільки зміни можна вносити на сервер, не впливаючи на клієнтів.

Ще одна перевага цієї архітектури полягає в тому, що вона забезпечує кращий захист і контроль над даними. Сервер забезпечує контроль доступу

та механізми автентифікації, гарантуючи, що лише авторизовані користувачі можуть отримати доступ до певних ресурсів або послуг. Це особливо важливо для безпечної роботи з конфіденційними даними [17].

### 3.3.1. *Серверна частина*

Серверна частина має наступні модулі для роботи з даними:

- Controller. Контролер відповідає за оброблення вхідних HTTP-запитів і надання належної відповіді. Він служить точкою входу для клієнтських запитів і містить методи, які відображають вхідні запити на певні дії або служби в програмі.
- Service. Рівень служби містить бізнес-логіку та обробляє дані, отримані від рівня контролера. Він діє як посередник між рівнем контролера та рівнем сховища та відповідає за керування специфічними операціями програми.
- Repository. Репозиторій відповідає за керуванням і збереженням даних. Він забезпечує інтерфейс рівня обслуговування для взаємодії з базою даних PostgreSQL.
- Model. Модель визначає структуру та атрибути об’єктів, що використовуються в додатку. Він представляє об’єкти домену та використовується рівнями служби та сховища для зберігання та отримання даних.
- Exception. Механізм оброблення винятків, що використовується для перехоплення та оброблення винятків під час виконання програми. Він використовується для надання значущих повідомлень про помилки клієнту та допомоги розробникам у виявленні та вирішенні проблем.
- Security. Компонент безпеки використовується для захисту програми та захисту від несанкціонованого доступу. Він включає механізми автентифікації та авторизації, такі як перевірка імені користувача, пароль та контроль доступу.

– Config. Компонент конфігурації містить параметри та конфігурації програми. Він включає загальну конфігурацію для вебзастосунку та конфігурацію для засобів безпеки.

Для реалізації роботи з базою даних використовувався JPA репозиторій. Репозиторій JPA – це компонент у Spring Boot, який забезпечує простий і елегантний спосіб виконання операцій CRUD (створення, читання, оновлення, видалення) у базі даних PostgreSQL за допомогою JPA. Він діє як посередник між програмою та базою даних, надаючи простий у використанні інтерфейс для виконання операцій з базою даних [18]. JPA (Java Persistence API) – це стандартна специфікація об'єктно-реляційного відображення в програмах на основі Java. Він надає набір інтерфейсів й анотацій для відображення об'єктів Java у таблиці реляційної бази даних та навпаки [19].

### 3.3.2. *Клієнтська частина*

Клієнтська частина має модулі головної сторінки, книга, моя бібліотека, авторизація та реєстрація, налаштування, статистика, головна сторінка адміністратора, сторінка створення книг та відповідних посилань, сторінка оновлення/видалення книг, сторінка блокування/розблокування користувачів та сторінка видалення відгуків.

Головна сторінка – це цільова сторінка програми, яка містить інструмент для пошуку книг; книги, які відображаються на сторінці, наприклад, популярні книги або рекомендовані книги; жанрові фільтри, які допомагають звузити результати пошуку; поле авторизації, щоб увійти або зареєструватися як новий користувач; логотипи та нижній колонтитул.

Компонент книга використовується для відображення інформації про окремі книги та містить додаткову інформацію, зокрема назва книги, автор, зображення обкладинки, опис, оцінки та відгуки. Цей компонент також може надавати функцію додавання книги до бібліотек користувачів або списку бажань.

Моя бібліотека – цей компонент відображає власні бібліотеки користувачів, включаючи книги, які додав користувач, які сподобались, які хоче прочитати користувач та іншу подібну інформацію.

Авторизація та реєстрація – цей компонент обробляє автентифікацію та реєстрацію користувачів. Він включає форму входу, де користувачі можуть ввести свої облікові дані та отримати доступ до програми. Він також містить функцію для скидання паролів користувачів, якщо вони його забули.

Налаштування – цей компонент надає користувачам можливість налаштувати параметри програми, наприклад вибір мови, налаштування сповіщень та інші подібні параметри.

Статистика – цей компонент надає користувачам можливість переглядати статистику, пов’язану з їх читацькою діяльністю, наприклад кількість прочитаних книг, кількість прочитаних сторінок та інші подібні показники.

Головна сторінка адміністратора – компонент, на якому розташовані всі функціональні можливості адміністратора, який можна обирати.

Сторінка створення книг та відповідних посилань – цей компонент дозволяє адміністратору додавати нові книги до системи, заповнюючи форму, яка містить деталі книги та посилання на пов’язані файли. Після надсилання форми книга додається до системи та стає доступною для пошуку, перегляду та читання користувачами.

Сторінка оновлення/видалення книг – цей компонент дозволяє адміністратору редагувати або видаляти існуючі книги в системі. Адміністратор може вибрати книгу зі списку існуючих книг й оновити будь-яке поле у записі книги. Крім того, адміністратор може повністю видалити книгу з системи.

Сторінка блокування/розблокування користувачів – цей компонент дозволяє адміністратору керувати обліковими записами користувачів,

блокуючи або розблоковуючи їх. Коли користувача заблоковано, він не може отримати доступ до програми, а його обліковий запис призупиняється.

Сторінка видалення відгуків – цей компонент дозволяє адміністратору видаляти відгуки користувачів, які порушують політику системи або є неприйнятними. Адміністратор може вибрати відгук зі списку наявних відгуків та видалити з системи.

## 3.4. Висновки по розділу

Для надання доступів до різних функціональних можливостей вебзастосунку було розроблено ієрархічну структуру, де виділено два рівні користувачів та розподілені їх ролі. Завдяки цьому читачі не зможуть виконувати дії, які не дозволені відповідно до їхньої ролі.

Вебзастосунок має сучасну та масштабовану архітектуру, яка базується на моделі клієнт-сервер. Сервер використовує базу даних PostgreSQL за допомогою репозиторіїв JPA. Сервер організовано в різні модулі, включаючи контролери, служби, сховища, моделі, винятки, безпеку та конфігурації, що допомагає підтримувати код упорядкованим і простим у обслуговуванні.

На стороні клієнта вебзастосунку були створені компоненти, які забезпечують роботу системи, включаючи головну сторінку, книги, бібліотеку користувача, авторизацію, реєстрацію та вихід з профілю, налаштування, профіль і статистику. Крім того, вебзастосунок має компоненти для керування системою як адміністратор, зокрема створення книг із посиланнями, оновлення/видалення книг, блокування/розблокування користувачів та видалення оглядів. Це допомагає гарантувати, що вебзастосунок простий у використанні та забезпечує чудовий досвід користувача для читачів. Також вебзастосунок забезпечує необхідні інструменти керування для адміністратора.

# 4. АНАЛІЗ РОЗРОБЛЕНОЇ СИСТЕМИ

## 4.1. Аналіз розробленого програмного застосунку

Вебзастосунок електронної бібліотеки був розроблений з метою створення зручного та інтуїтивного простору для організації, зберігання та роботи з літературними джерелами.

Відповідно до функціональних вимог вебзастосунку було реалізовано наступне:

- Ієрархія системи, а саме рівні доступу адміністратора та користувача.
- Сторінка реєстрації/авторизації користувача або адміністратора.
- Функціональні сторінки адміністратора, за допомогою яких він виконує модерацію системи. Функціональні сторінки адміністратора були розроблені у вигляді спливаючих діалогових вікон. Всього адміністратор має 6 функціональних сторінок. Вони мають схожий вигляд, тому на рис. 4.3-4.4 буде представлено вигляд деяких з них.
- Головна сторінка застосунку, на якій звичайні користувачі отримують доступ до функціональних можливостей вебзастосунку.
- Сторінка книги, на якій користувачі отримують інформацію про відповідну книгу, її рейтинг та відгуки.
- Сторінка налаштувань, де користувач може задати власті потрібні налаштування.
- Сторінка перегляду власної бібліотеки користувача, в якій можна переглянути додані за певними критеріями книги.
- Сторінка статистики, де користувач може переглянути статистику щодо прочитаних книг.
- Можливість додавання користувачами коментарів до книжок.

Відповідно до визначених нефункціональних вимог було реалізовано наступне:

- Доступність вебзастосунку для використання на різних пристроях та браузерах.
- Безпека даних користувача за допомогою ієрархічної системи доступу до даних та шифрування даних.
- Зрозумілий інтерфейс вебзастосунку для користувачів будь-якого рівня. Це було забезпечено за допомогою візуальної бібліотеки Angular material.
- Зручний та зрозумілий інтерфейс вебзастосунку для адміністраторів.

## 4.2. Опис інтерфейсу вебзастосунку

Розглянемо розроблені сторінки вебзастосунку. Сторінки авторизації та реєстрації містять поля з валідацією. Вони майже ідентичні, тому представлено вигляд сторінки реєстрації на рис. 4.1.

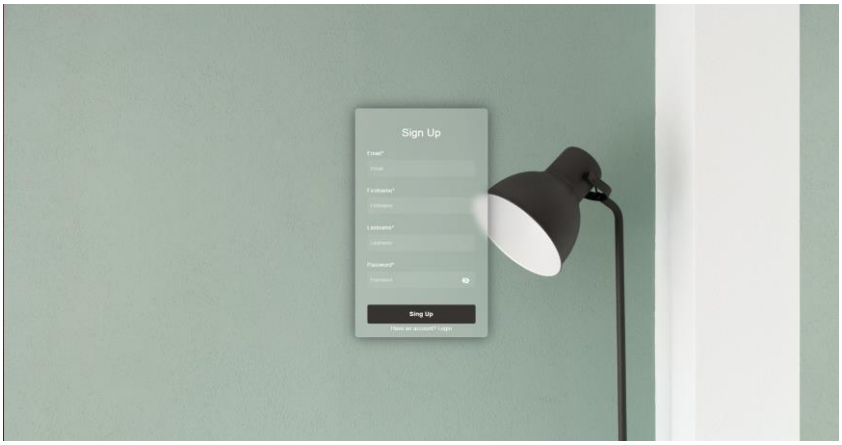



Рис. 4.1. Сторінка реєстрації користувача

Сторінка адміністратора містить опції для переходу до сторінок функціональних можливостей та повне ім’я адміністратора. Вигляд головної сторінки адміністратора представлено на рис. 4.2.

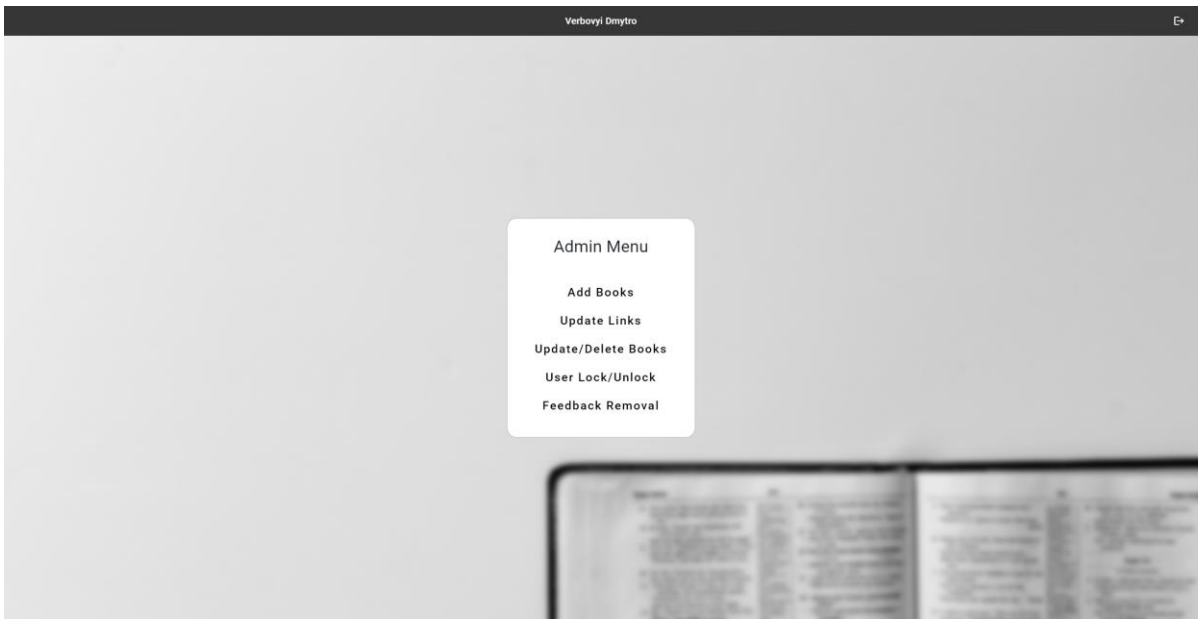



Рис. 4.2. Головна сторінка адміністратора системи

Функціональні сторінки адміністратора розроблені у вигляді спливаючих вікон. Частина вікон сприймає вхідні дані, частина відображає список сутностей для вибору. Вигляд цих вікон відповідно представлено на рис. 4.3-4.4.

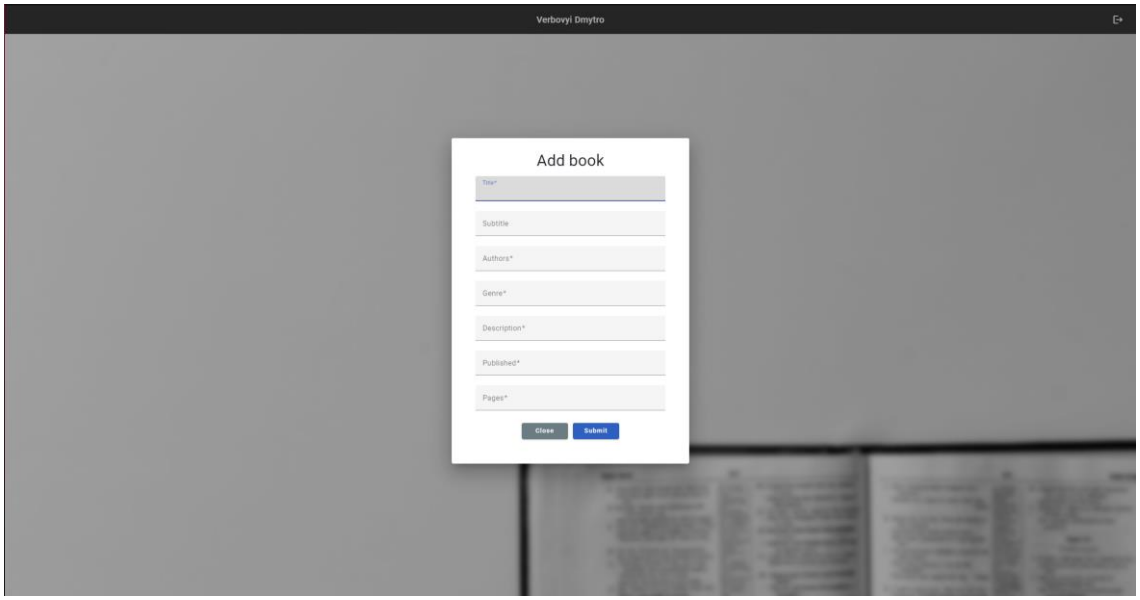



Рис. 4.3. Функціональна сторінка додавання книги

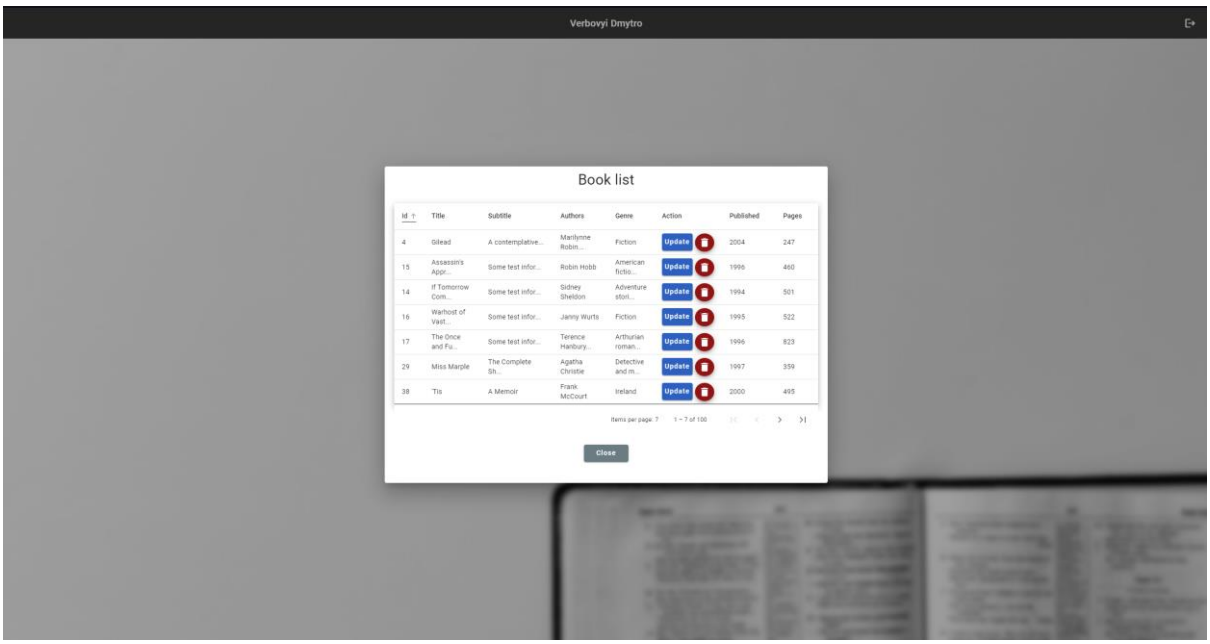



Рис. 4.4. Функціональна сторінка перегляду та видалення книги

На головній сторінці вебзастосунку відображаються книги з набільшим рейтингом, рекомендовані книги для користувача та рядок пошуку. Вигляд головної сторінки представлено на рис. 4.5.

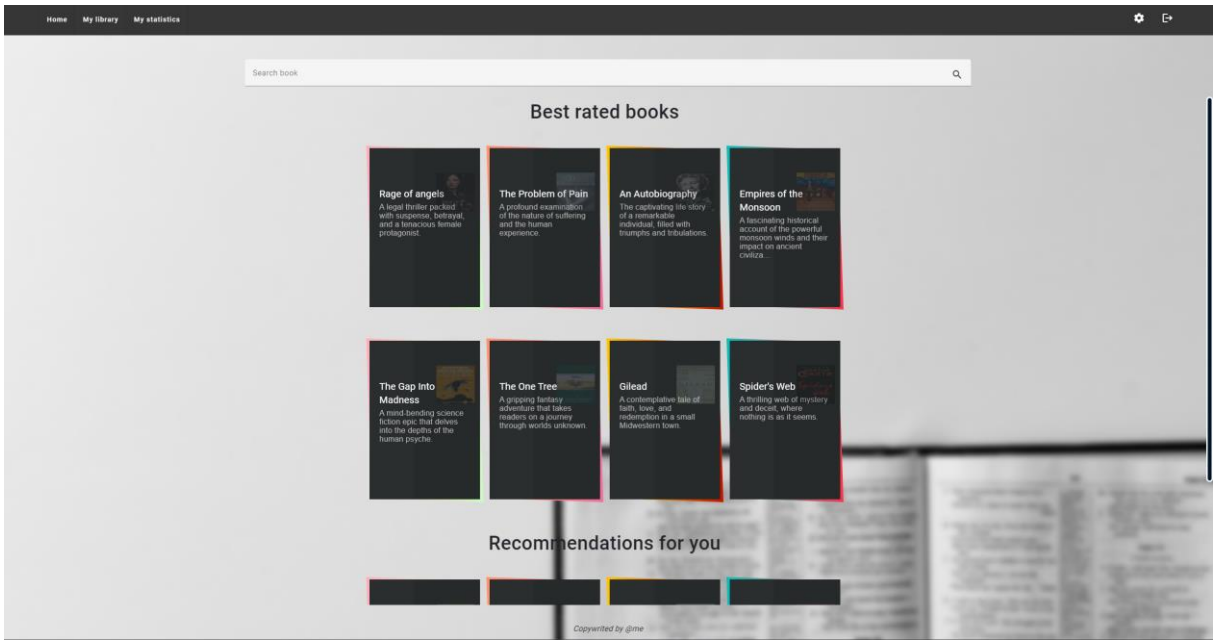



Рис. 4.5. Головна сторінка вебзастосунку

Сторінка книги містить у собі інформацію про книгу, відгуки інших користувачів, власний відгук користувача або поле для залишення його, кнопки для додавання книги до власної бібліотеки та категорії. Вигляд сторінки представлено на рис. 4.6.

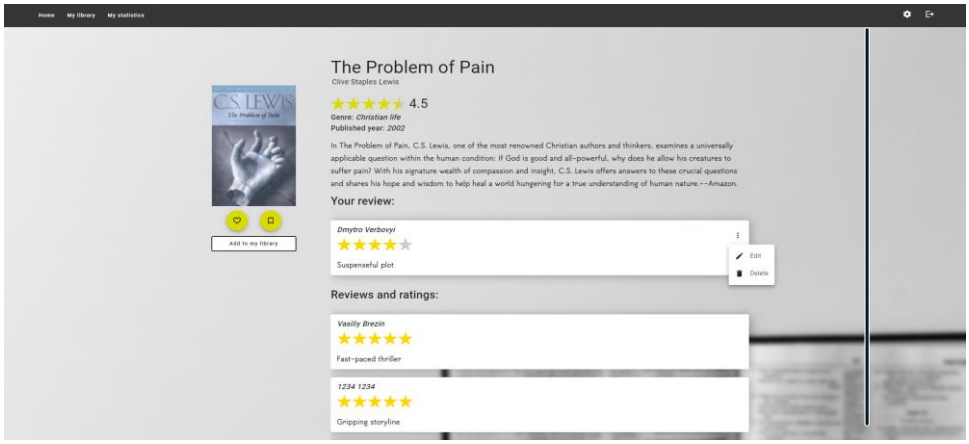



Рис. 4.6. Сторінка книги

На сторінці власної бібліотеки користувач може читати або прослуховувати книги, змінювати категорії книг. Також можливо приховати категорії та завантажувати файли для читання або прослуховування книги офлайн. Вигляд сторінки представлено на рис. 4.7.

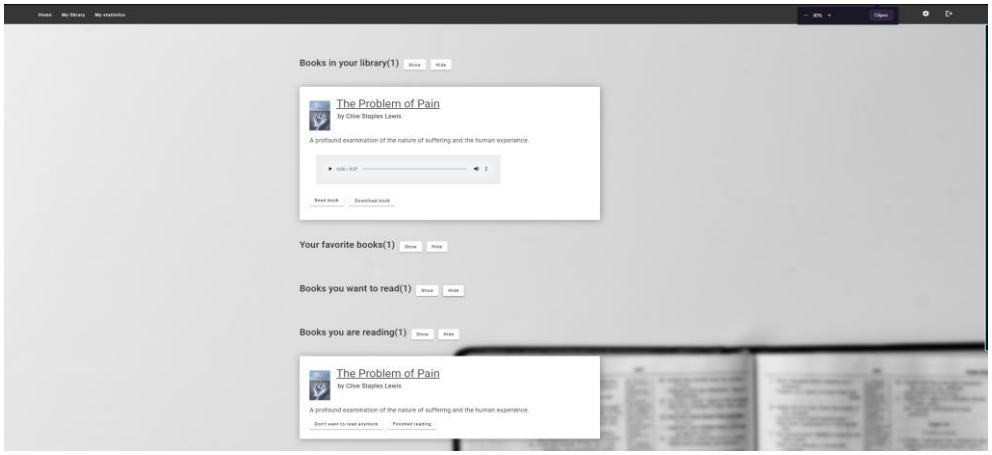



Рис. 4.7. Сторінка власної бібліотеки

Сторінка, на якій відображається статистика користувача представлена на рис. 4.8.


Рис. 4.8. Сторінка власної бібліотеки

## 4.3. Тестування вебзастосунку електронної бібліотеки

Для безперебійної та правильної роботи, недопущення багів та інших проблем для вебзастосунку було створено план тестування. План тестування наступний:

1. Перевірка авторизації.
    - Авторизація у системі за допомогою валідних даних.
    - Авторизація у системі з невалідними даними.
2. Перевірка ієрархії рівнів доступу до системи.
    - Входимо в обліковий запис адміністратора. Перевіряємо головну сторінку.
    - Входимо в обліковий запис користувача. Перевіряємо головну сторінку. Намагаємось зайти на сторінку доступу адміністратора.
3. Перевірка функцій адміністратора.
    - Додавання літературного джерела до системи.
    - Редагування літературного джерела у системі.

- Видалення літературного джерела з системи.
- Блокування користувача.
- Розблокування користувача.
- Видалення відгуку про книгу.

4. Перевірка функцій користувачів
   - Перегляд сторінки про книги.
   - Зміна налаштувань користувача.
   - Перевірка пошуку книг.
   - Перевірка додавання книги до розділів «Улюблені», «Хочу прочитати».
   - Перегляд книг, доданих до власної бібліотеки.
   - Читання книги онлайн.
   - Перевірка додавання книги до розділів «Читаю», «Прочитано».
   - Завантаження файлу з книгою.
   - Перевірка аудіокниги.
   - Перевірка статистики прочитаних книг.
   - Перевірка додавання відгуку про книгу.

Після виконання цього плану тестування можна бути впевненим, що вебзастосунок буде працювати стабільно та без проблем.

## 4.4. Рекомендації щодо вдосконалення вебзастосунку

Вебзастосунок електронної бібліотеки має достатні функціональні можливості, але існують напрямки його вдосконалення. Тому були сформовані наступні рекомендації.

– Збирання відгуків користувачів, щоб зрозуміти їхні потреби та проблеми.
– Впровадження багатофакторної автентифікації, щоб додати додатковий рівень безпеки.

– Застосування механізмів кешування для підвищення продуктивності даних, до яких часто звертаються.
– Використання інструментів профілювання коду та моніторингу продуктивності, щоб виявити та усунути вузькі місця продуктивності.
– Застосування методів кешування на стороні сервера та клієнта, щоб зменшити навантаження на мережу та покращити час відповіді.
– Впровадження системи контролю версій для відстеження змін та сприяння співпраці між розробниками.
– Використання методів горизонтального та вертикального масштабування, щоб впоратися зі збільшенням трафіку та попиту користувачів.
– Створення локалізації на інші мови.
– Створення мобільної версії додатку.

Реалізація цих пунктів сприятиме покращенню додатку, його можливостей, продуктивності та поширить його на більшу кількість пристроїв, що приверне увагу нових користувачів.

## 4.5. Висновки до розділу

Розроблений вебзастосунок електронної бібліотеки успішно реалізував функціональні та нефункціональні вимоги, забезпечивши користувачам зручну та інтуїтивно зрозумілу платформу для організації, зберігання та роботи з літературними джерелами. Системна ієрархія, яка включає рівні доступу адміністратора та користувача дозволяє належним чином керувати та контролювати функціональні можливості системи.

План тестування, викладений для розробленого вебзастосунку забезпечує ретельне тестування різних аспектів, включаючи авторизацію, рівні доступу, функції адміністратора та функції користувача.

Для подальшого вдосконалення вебзастосунку було створено ряд рекомендацій. Впроваджуючи ці рекомендації, вебзастосунок можна додатково вдосконалити та забезпечити покращений досвід роботи з користувачем, підвищити рівень безпеки, а також масштабованість та продуктивність.

# ВИСНОВКИ

Метою дипломного проєкту є розроблення вебзастосунку електронної бібліотеки, що забезпечить зручний та інтуїтивно зрозумілий простір для організації, зберігання та роботи з літературними джерелами. Дипломний проєкт передбачав аналіз існуючих рішень, обґрунтування теми, вибір засобів реалізації та розроблення вебзастосунку.

Аналіз існуючих програм електронної бібліотеки визначив недостатність функцій керування літературними джерелами, недостатність функціональних можливостей користувачів системи, складність у використанні. Тому актуальним завданням є розроблення нового вебзастосунку електронної бібліотеки, який усунув наявні проблеми та обмеження. Актуальність розроблення нового вебзастосунку також визначена зростаючим попитом на цифрові бібліотеки та переваги, які вони пропонують, зокрема легкий доступ до широкого спектра книг, зручність і портативність. Розроблений вебзастосунок призначений для задоволення вимог користувачів з такими функціями, як реєстрація та авторизація, пошук, перегляд, читання та коментування літературних джерел, розширений профіль користувачів з можливостями категоризації обраних джерел, ведення статистики та інше.

Для розроблення вебзастосунку електронної бібліотеки було обрано мову програмування серверного модуля Java та фреймворк Spring Boot, клієнтського модуля TypeScript та фреймворк Angular, базу даних PostgreSQL.

Розроблений вебзастосунок має комплексну архітектуру з реалізацією на стороні сервера та клієнта. Структура бази даних розроблена для ефективного зберігання та отримання даних, пов’язаних із книгами, профілями користувачів та функціями системи. Архітектура системи забезпечує автентифікацію користувача, контроль доступу та шифрування даних для захисту даних користувача.

Аналіз розробленого вебзастосунку продемонстрував успішну реалізацію функціональних вимог. План тестування охоплював авторизацію, контроль рівнів доступу, функції адміністратора та функції користувача. Були надані рекомендації щодо подальшого вдосконалення розробленого вебзастосунку, зокрема підвищення безпеки, оптимізація продуктивності та розроблення мобільної версії застосунку.

Таким чином, дипломний проєкт успішно досяг мети розроблення вебзастосунку електронної бібліотеки. Впроваджене рішення надає користувачам зручну та багатофункціональну платформу для доступу та керування літературними джерелами. Проєкт зробив внесок у розвиток цифрових бібліотек, задовольняючи потреби сучасних читачів та пропонуючи покращений досвід читання в епоху цифрових технологій.

## СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

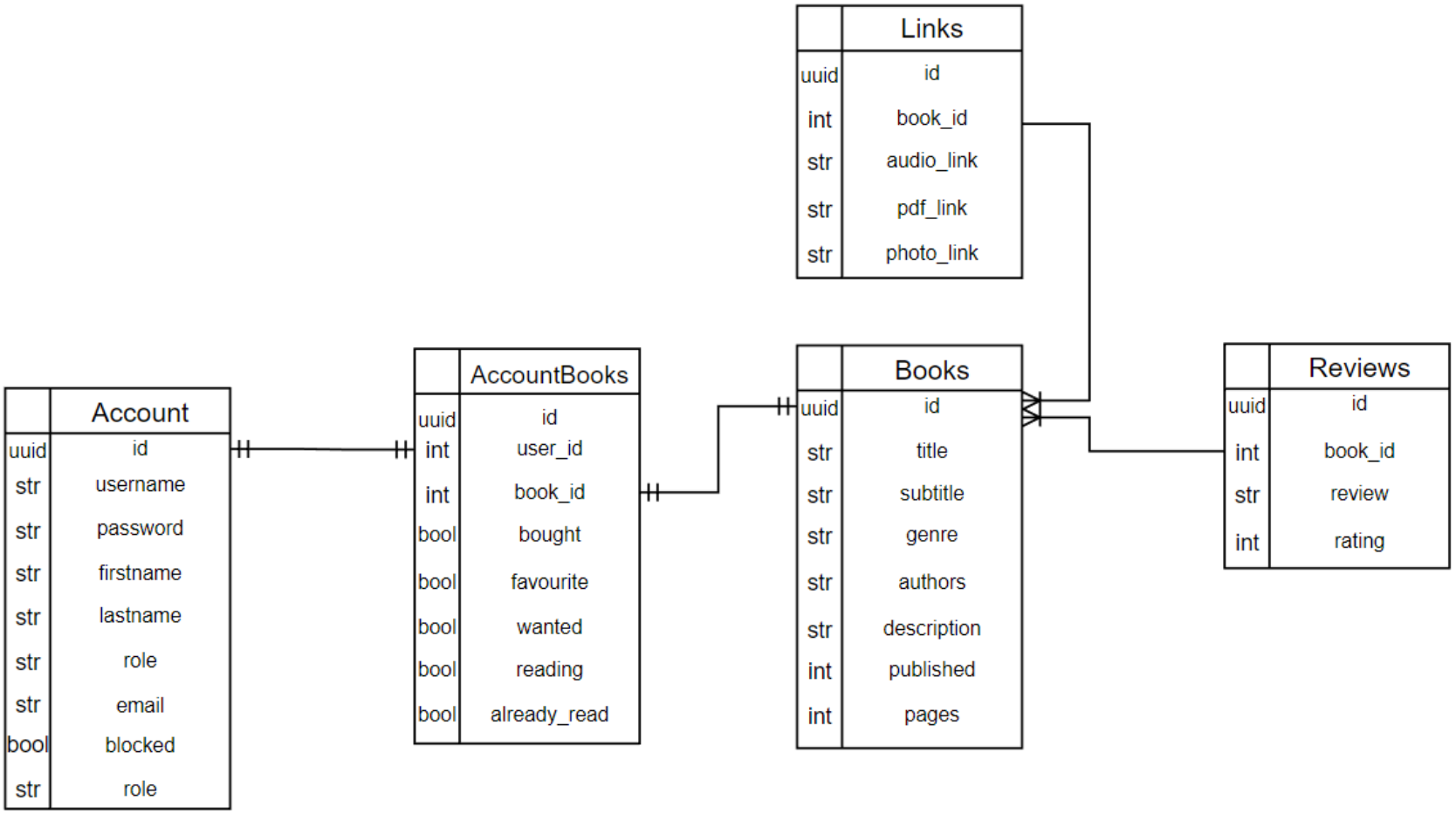

1. Python Advantages and Disadvantages - Step in the right direction - TechVidvan [Електронний ресурс] // TechVidvan. — Режим доступу: https://techvidvan.com/tutorials/python-advantages-and-disadvantages/
2. Welcome to python.org [Електронний ресурс] // Python.org. — Режим доступу: https://www.python.org/about/
3. Pros and cons of java | advantages and disadvantages of java - dataflair [Електронний ресурс] // DataFlair. — Режим доступу: https://data-flair.training/blogs/pros-and-cons-of-java/
4. What is java used for? [Електронний ресурс] // Coursera. — Режим доступу: https://www.coursera.org/articles/what-is-java-used-for/
5. Deed I. Pros and cons of javascript development | pangea.ai [Електронний ресурс] / Ian Deed // Hire flutter, mobile & JS software devs | Pangea.ai | 2023. — Режим доступу: https://www.pangea.ai/dev-javascript-resources/best-practices/
6. What is javascript used for? - lighthouse labs [Електронний ресурс] // Lighthouse Labs. — Режим доступу: https://www.lighthouselabs.ca/en/blog/what-is-javascript-used-for/
7. What is HTML? Hypertext markup language basics explained [Електронний ресурс] // Hostinger Tutorials. — Режим доступу: https://www.hostinger.com/tutorials/what-is-html/
8. What is CSS? [Електронний ресурс] // Online Courses and eBooks Library. — Режим доступу: https://www.tutorialspoint.com/css/what_is_css.htm
9. What Is TypeScript? [Електронний ресурс] // The New Stack. — Режим доступу: https://thenewstack.io/what-is-typescript/
10. Editor. The Good and the Bad of TypeScript [Електронний ресурс] / Editor // AltexSoft. — Режим доступу: https://www.altexsoft.com/blog/typescript-pros-and-cons/

11. Spring | why spring [Електронний ресурс] // Why Spring. — Режим доступу: https://spring.io/why-spring
12. Pros and cons of using spring boot [Електронний ресурс] // Insights. — Режим доступу: https://bambooagile.eu/insights/pros-and-cons-of-using-spring-boot/
13. Editor. The good and the bad of angular development [Електронний ресурс] / Editor // AltexSoft. — Режим доступу: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-angular-development/
14. Angular [Електронний ресурс] // Angular. — Режим доступу: https://angular.io/features
15. PostgreSQL: about [Електронний ресурс] // PostgreSQL: The world's most advanced open source database. — Режим доступу: https://www.postgresql.org/about/
16. PostgreSQL advantages: benefits of using postgresql [Електронний ресурс] // Prisma's Data Guide. — Режим доступу: https://www.prisma.io/dataguide/postgresql/benefits-of-postgresql
17. Terra J. What is client-server architecture? Everything you should know | simplilearn [Електронний ресурс] / John Terra // Simplilearn.com. — Режим доступу: https://www.simplilearn.com/what-is-client-server-architecture-article
18. Simplilearn. What is a JPA repository in spring boot? (with examples) | simplilearn [Електронний ресурс] / Simplilearn // Simplilearn.com. — Режим доступу: https://www.simplilearn.com/tutorials/jpa-tutorial/spring-boot-jpa
19. What is JPA? Introduction to java persistence [Електронний ресурс] // InfoWorld. — Режим доступу: https://www.infoworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html

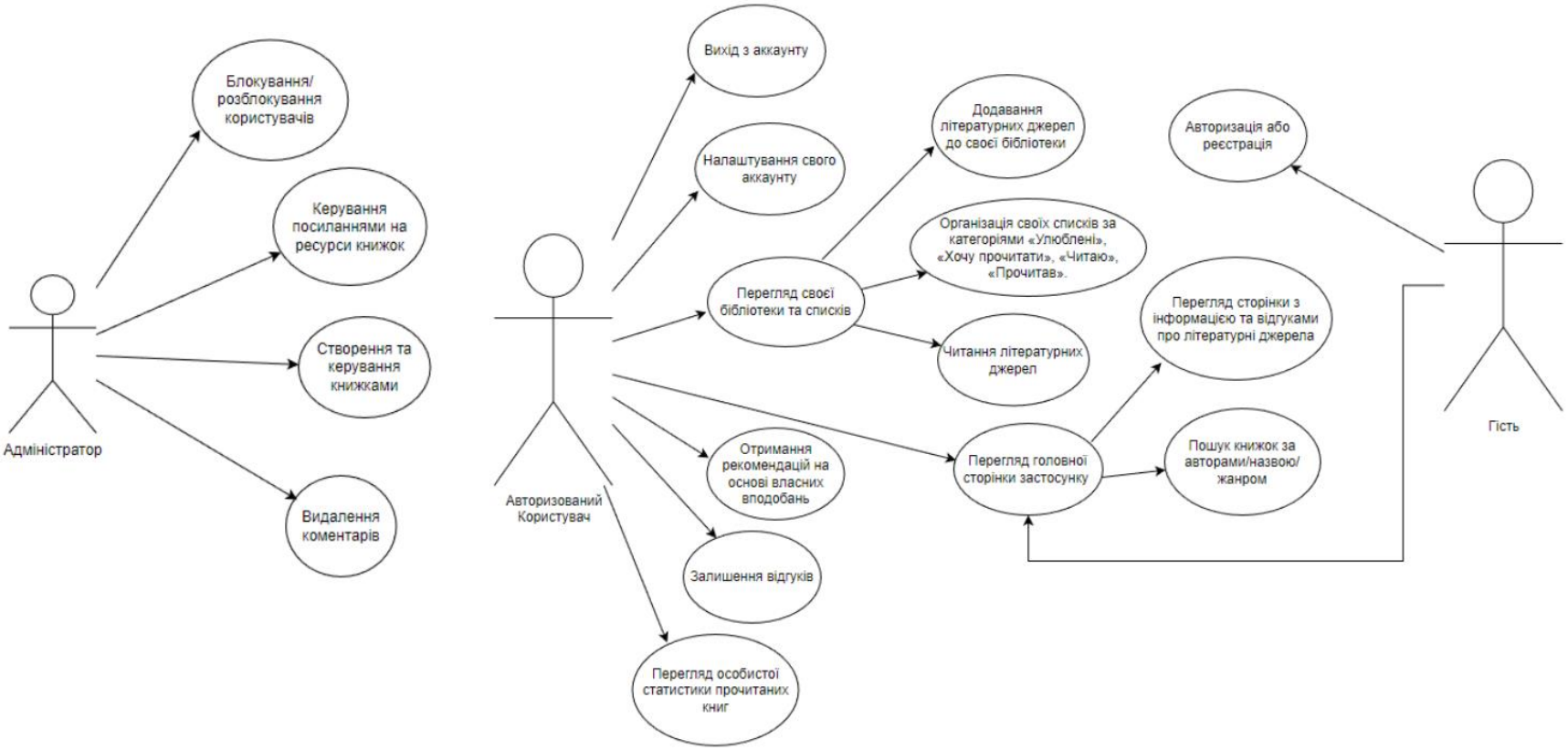
**ДОДАТКИ**

**Додаток 1**

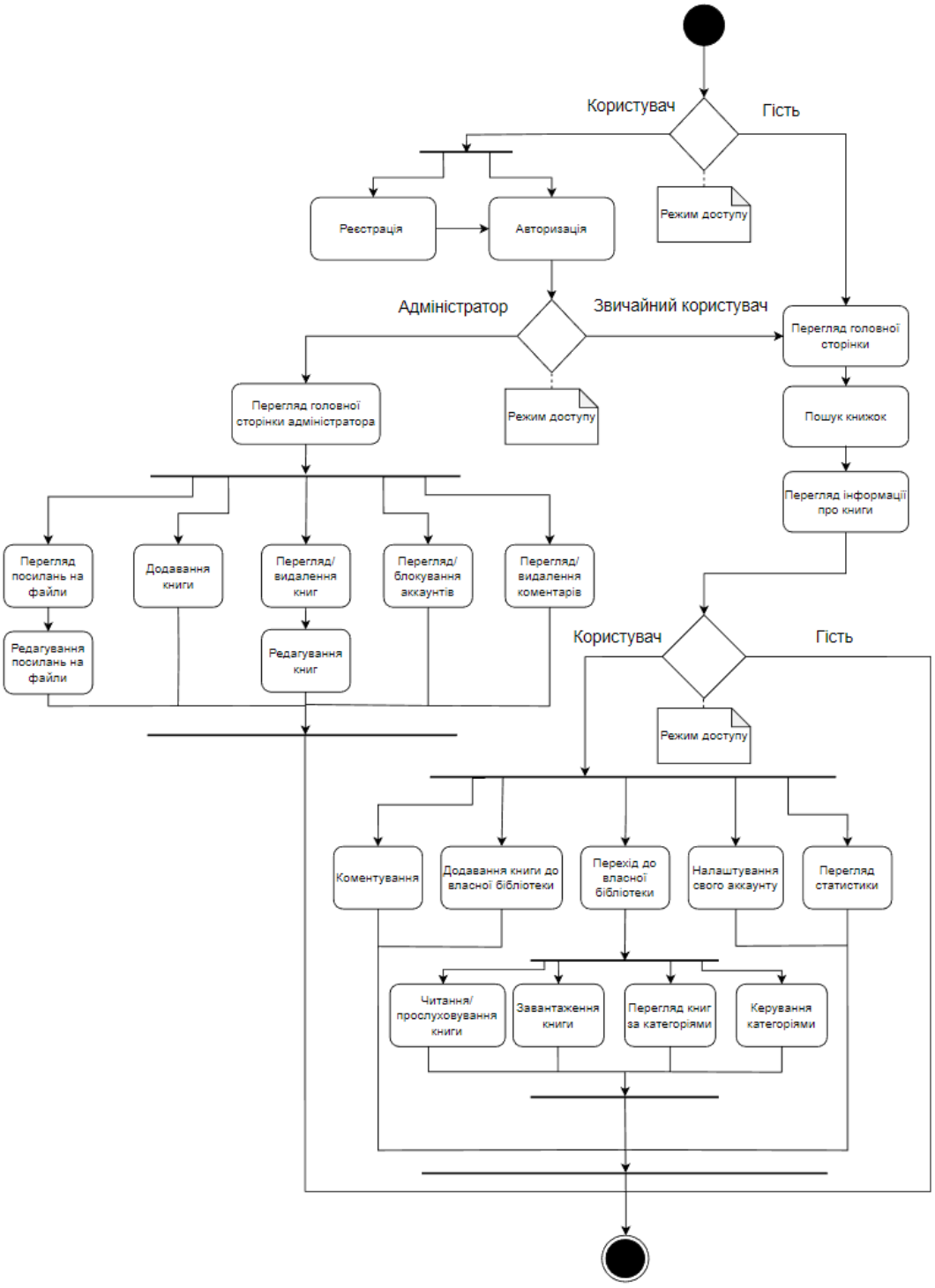
**Копії графічних матеріалів**



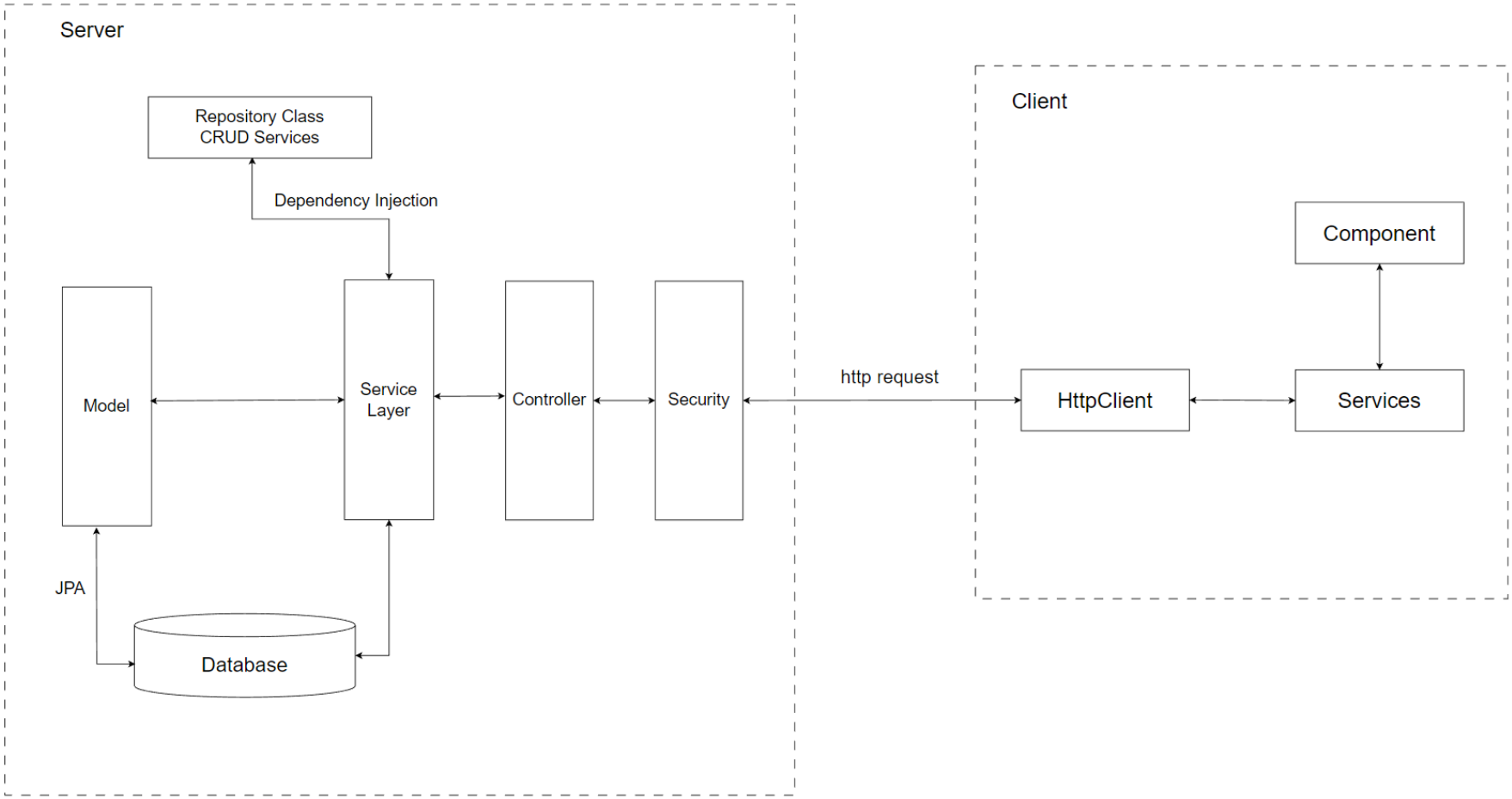

ДП.045440-06-99
Вебзастосунок електронної бібліотеки. Архітектура вебзастосунку. Схема бази даних




ДП.045440-07-99
Вебзастосунок електронної бібліотеки. Функціональні можливості вебзастосунку. Діаграма використання




Вебзастосунок електронної бібліотеки. Алгоритм роботи вебзастосунку, група КП-91, Вербовий Дмитро




Вебзастосунок електронної бібліотеки. Архітектура вебзастосунку. Діаграма компонентів, група КП-91, Вербовий Дмитро

# Додаток 2

# Фрагменти тексту програм

**SecurityConfig.java**

```
package com.diploma.elibrary.config;

import com.diploma.elibrary.security.JwtAuthenticationFilter;
import lombok.RequiredArgsConstructor;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationProvider;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebS
ecurity;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.web.SecurityFilterChain;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticat
ionFilter;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

import java.util.List;

@Configuration
@EnableWebSecurity
@RequiredArgsConstructor
public class SecurityConfig implements WebMvcConfigurer {

    private final JwtAuthenticationFilter jwtAuthFilter;
    private final AuthenticationProvider authenticationProvider;

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http)
throws Exception {

        CorsConfiguration corsConfiguration = new CorsConfiguration();
        corsConfiguration.addAllowedOrigin("http://localhost:4200");
        corsConfiguration.addAllowedHeader("*");
        corsConfiguration.addAllowedMethod("*");
        corsConfiguration.setAllowCredentials(true);
        corsConfiguration.setExposedHeaders(List.of("Authorization"));

        http
                .csrf()
                .disable()
                .authorizeHttpRequests()
                .requestMatchers("/api/auth", "/api/register",
"/api/checkEmail/**", "/api/noauth/**")
                .permitAll()
                .requestMatchers("/api/users",
"/api/admin/**").hasAuthority("ADMIN")
                .anyRequest()
                .authenticated()
                .and()
                .sessionManagement()
                .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
                .and()
                .authenticationProvider(authenticationProvider)
                .addFilterBefore(jwtAuthFilter,
UsernamePasswordAuthenticationFilter.class).cors().configurationSource(requ
est -> corsConfiguration);
        return http.build();
    }
```

```
}
```

**BookController.java**

```
package com.diploma.elibrary.controller;

import com.diploma.elibrary.model.Book;
import com.diploma.elibrary.service.BookServiceImpl;
import com.diploma.elibrary.service.S3Service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;


import java.util.HashMap;
import java.util.List;
import java.util.Map;

@RestController
@RequestMapping("api")
public class BookController {
    private final BookServiceImpl service;
    private final S3Service s3Service;
    @Autowired
    public BookController(BookServiceImpl bookService, S3Service s3Service)
{
        this.service = bookService;
        this.s3Service = s3Service;
    }

    //get all accounts
    @GetMapping("/noauth/books/search/{search}")
    public ResponseEntity<List<Book>> searchBooks(@PathVariable String
search) {
        return ResponseEntity.ok(service.searchBooks(search));
    }

    @GetMapping("/noauth/books")
    public ResponseEntity<List<Book>> getBooks() {
        return ResponseEntity.ok(service.getAllBooks());
    }

    @PutMapping("/admin/book/update/{id}")
    public ResponseEntity<Book> updateBook(@PathVariable Long id,
@RequestBody Book book) {
        return ResponseEntity.ok(service.updateBook(id, book));
    }

    @PostMapping("/admin/book/new")
    public ResponseEntity<Book> createBook(@RequestBody Book book) {
        return ResponseEntity.ok(service.createBook(book));
    }

    @DeleteMapping("/admin/book/delete/{id}")
    public ResponseEntity<Map<String, Boolean>> deleteBook(@PathVariable
Long id) {
        service.deleteBook(id);
        Map<String, Boolean> response = new HashMap<>();
        response.put("deleted", Boolean.TRUE);
        return ResponseEntity.ok(response);
    }
```

```
    @PostMapping("/admin/book/upload/{book_id}")
    public ResponseEntity<?> uploadBook(@PathVariable Long book_id,
@RequestParam("files") MultipartFile[] files) {
        if (files.length == 3) {
            MultipartFile book = null, audio = null, photo = null;
            for (MultipartFile file : files) {
                String filename = file.getOriginalFilename();
                if(filename.matches("(?i).*\\.(pdf)"))
                {
                    book = file;
                }
                else if(filename.matches("(?i).*\\.(mp3|wav|ogg|flac)"))
                {
                    audio = file;
                }
                else
if(filename.matches("(?i).*\\.(jpg|jpeg|png|gif|bmp)"))
                {
                    photo = file;
                }
            }
            if(book != null && audio != null && photo != null &&
!book.isEmpty() && !audio.isEmpty() && !photo.isEmpty())
            {
                Map<String, String> response = new HashMap<>();
                response.put("response", s3Service.uploadBook(book_id,
files[0], files[1], files[2]));
                return ResponseEntity.ok(response);
            }
            else {
                String response = "Bad files";
                return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body(response);
            }
        }
        else
        {
            String response = "Bad files";
            return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body(response);
        }
    }

    @GetMapping("/books/recommended/account/{account_id}")
    public ResponseEntity<List<Book>> getRecommendedBooks(@PathVariable
Long account_id) {
        return ResponseEntity.ok(service.getRecommendedBooks(account_id));
    }

    @GetMapping("/noauth/books/bestRated")
    public ResponseEntity<List<Book>> getBestRatedBooks() {
        return ResponseEntity.ok(service.getBestRatedBooks());
    }

    @GetMapping("/noauth/book/rating/{book_id}")
    public ResponseEntity<Double> getBookRating(@PathVariable Long book_id)
{
        return ResponseEntity.ok(service.getBookRating(book_id));
    }

    @GetMapping("/noauth/book/{book_id}")
    public ResponseEntity<Book> getBook(@PathVariable Long book_id) {
        return ResponseEntity.ok(service.getBook(book_id));
```

```
    }
    @GetMapping("/books/account/{account_id}")
    public ResponseEntity<List<Book>> getBookByAccount(@PathVariable Long
account_id) {
        return ResponseEntity.ok(service.getAllAccountBooks(account_id));
    }
}
```

**ResouseNotFoundException.java**

```
package com.diploma.elibrary.exception;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;


@ResponseStatus(value = HttpStatus.NOT_FOUND)
public class ResourceNotFoundException extends RuntimeException{
    public ResourceNotFoundException(String message){
        super(message);
    }
}
```

**Book.java**

```
package com.diploma.elibrary.model;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.fasterxml.jackson.annotation.JsonManagedReference;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

@AllArgsConstructor
@NoArgsConstructor
@Data
@Entity
@Table(name="books")
@JsonIgnoreProperties({"link", "reviews", "accountBooks"})
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(name = "title")
    private String title;
    @Column(name = "subtitle")
    private String subtitle;
    @Column(name = "authors")
    private String authors;
    @Column(name = "genre")
    private String genre;
    @Column(name = "description")
    private String description;
    @Column(name = "published")
    private Integer published;
    @Column(name = "pages")
    private Integer pages;
    @OneToOne(mappedBy = "book", cascade = CascadeType.ALL, fetch =
FetchType.EAGER)
    private Link link;
```

```
    @JsonManagedReference
    @OneToMany(mappedBy = "book")
    private List<Review> reviews = new ArrayList<>();
    @JsonManagedReference
    @OneToMany(mappedBy = "book")
    private List<AccountBook> accountBooks = new ArrayList<>();


    @Override
    public String toString() {
        return "Book{" +
                "id=" + id +
                ", title='" + title + '\'' +
                ", subtitle='" + subtitle + '\'' +
                ", authors='" + authors + '\'' +
                ", genre='" + genre + '\'' +
                ", description='" + description + '\'' +
                ", published=" + published +
                ", pages=" + pages +
                ", link=" + link +
                '}';
    }
}
```

**BookRepository.java**

```
package com.diploma.elibrary.repository;

import com.diploma.elibrary.model.Book;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.Optional;

@Repository
public interface BookRepository extends JpaRepository<Book, Long> {

    Optional<Book> findBookByTitle(String title);

    @Query(value = "SELECT * FROM books WHERE title ILIKE %?1% OR authors
ILIKE %?1% OR genre ILIKE %?1% OR subtitle ILIKE %?1% ", nativeQuery =
true)
    Optional<List<Book>> findBooksByTitleOrAuthorsOrGenreOrSubtitle(String
search);

    Optional<List<Book>> findBookByPublished(Integer published);
}
```

**JwtAuthenticationFilter.java**

```
package com.diploma.elibrary.security;

import com.diploma.elibrary.service.JwtService;
import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationT
oken;
import org.springframework.security.core.context.SecurityContextHolder;
```

```
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.web.authentication.WebAuthenticationDetailsSou
rce;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;

@Component
@RequiredArgsConstructor
public class JwtAuthenticationFilter extends OncePerRequestFilter {


    private final JwtService jwtService;

    private final UserDetailsService userDetailsService;

    @Override
    protected void doFilterInternal(@NonNull HttpServletRequest request,
                                    @NonNull HttpServletResponse response,
                                    @NonNull FilterChain filterChain)
            throws ServletException, IOException {
            final String authHeader = request.getHeader("Authorization");
            final String jwt;
            final String email;
            if(authHeader == null || !authHeader.startsWith("Bearer "))
            {
                filterChain.doFilter(request, response);
                return;
            }
            jwt = authHeader.substring(7);
            email = jwtService.extractUsername(jwt);
            if(email != null &&
SecurityContextHolder.getContext().getAuthentication() == null)
            {
                UserDetails userDetails =
this.userDetailsService.loadUserByUsername(email);
                if(jwtService.isTokenValid(jwt, userDetails)){
                    UsernamePasswordAuthenticationToken authToken = new
UsernamePasswordAuthenticationToken(
                            userDetails, null,
userDetails.getAuthorities());
                    authToken.setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));

SecurityContextHolder.getContext().setAuthentication(authToken);
                }
            }
        filterChain.doFilter(request, response);
    }
}
```

**BookServiceImpl.java**

```
package com.diploma.elibrary.service;


import com.diploma.elibrary.exception.ResourceNotFoundException;
import com.diploma.elibrary.model.Account;
import com.diploma.elibrary.model.AccountBook;
import com.diploma.elibrary.model.Book;
import com.diploma.elibrary.model.Review;
```

```
import com.diploma.elibrary.repository.AccountBookRepository;
import com.diploma.elibrary.repository.AccountRepository;
import com.diploma.elibrary.repository.BookRepository;
import com.diploma.elibrary.service.interfaces.BookService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.*;
import java.util.stream.Collectors;

@Service
public class BookServiceImpl implements BookService {
    private final BookRepository bookRepository;
    private final LinkServiceImpl linkService;
    private final AccountRepository accountRepository;
    private final AccountBookRepository accountBookRepository;
    private final ReviewServiceImpl reviewService;

    @Autowired
    public BookServiceImpl(BookRepository bookRepository,
                           LinkServiceImpl linkService,
                           ReviewServiceImpl reviewService,
                           AccountBookRepository accountBookRepository,
                           AccountRepository accountRepository) {
        this.bookRepository = bookRepository;
        this.linkService = linkService;
        this.reviewService = reviewService;
        this.accountBookRepository = accountBookRepository;
        this.accountRepository = accountRepository;
    }
    @Override
    public Book findByTitle(String title) {
        return bookRepository.findBookByTitle(title).orElseThrow(() -> new
ResourceNotFoundException("No such book"));
    }

    @Override
    public List<Book> searchBooks(String search) {
        try{
            Integer published = Integer.valueOf(search);
            return
bookRepository.findBookByPublished(published).orElseThrow(() -> new
ResourceNotFoundException("Nothing found"));
        }
        catch (NumberFormatException e) {
            return
bookRepository.findBooksByTitleOrAuthorsOrGenreOrSubtitle(search).orElseThr
ow(() -> new ResourceNotFoundException("Nothing found"));
        }
    }

    @Override
    public List<Book> getRecommendedBooks(Long account_id) {
        Account account =
accountRepository.findById(account_id).orElseThrow(()->new
ResourceNotFoundException("No account with such id"));
        List<AccountBook> accountBooks =
accountBookRepository.findAccountBooksByAccount(account).orElseGet(ArrayLis
t::new);
        Map<Long, Double> avgRatings = getAvgRatings();
        List<String> genres = accountBooks.stream()
                .map(AccountBook::getBook)
```

```
                .map(Book::getGenre)
                .distinct()
                .toList();

        List<String> authors = accountBooks.stream()
                .map(AccountBook::getBook)
                .map(Book::getAuthors)
                .distinct()
                .toList();

        List<Book> recommendedBooks = getAllBooks().stream()
                .filter(book -> genres.contains(book.getGenre()) ||
containsAnyAuthor(book.getAuthors(), authors))
                .sorted(Comparator.comparingDouble(book ->
avgRatings.getOrDefault(book.getId(), 0.0)))
                .collect(Collectors.toList());

        return recommendedBooks.subList(0,
Math.min(recommendedBooks.size(), 4));
    }

    private boolean containsAnyAuthor(String bookAuthors, List<String>
authors) {
        for (String author : authors) {
            if (bookAuthors.contains(author)) {
                return true;
            }
        }
        return false;
    }

    @Override
    public List<Book> getAllBooks() {
        return bookRepository.findAll();
    }

    @Override
    public Double getBookRating(Long book_id) {
        List<Review> reviews = reviewService.getReviewsByBook(book_id);
        Double avgRating = 0.0;
        int counter = 0;
        if(reviews.isEmpty())
            return 0.0;
        for(Review review : reviews)
        {
            avgRating += Double.valueOf(review.getRating());
            counter++;
        }
        avgRating = avgRating/counter;
        return avgRating;
    }

    public Book getBook(Long book_id) {
        return bookRepository.findById(book_id).orElseThrow(() -> new
ResourceNotFoundException("Book not found"));
    }

    @Override
    public List<Book> getBestRatedBooks() {
        Map<Long, Double> avgRatings = getAvgRatings();
        int counter= 0;
        List<Book> bestRatedBooks = new ArrayList<>();
        for(Map.Entry<Long, Double> entry : avgRatings.entrySet())
```

```
        {
            if(counter >= 8)
                break;

bestRatedBooks.add(bookRepository.findById(entry.getKey()).orElseThrow(() -
> new ResourceNotFoundException("Wrong book_id")));
            counter++;
        }
        return bestRatedBooks;
    }

    private Map<Long, Double> getAvgRatings(){
        Map<Long, Integer> ratings = new HashMap<>();
        Map<Long, Integer> counters = new HashMap<>();
        List<Review> reviews = reviewService.getAllReviews();
        for(Review review : reviews)
        {
            if(ratings.containsKey(review.getBook().getId()))
            {
                Integer new_rating = ratings.get(review.getBook().getId())
+ review.getRating();
                ratings.replace(review.getBook().getId(), new_rating);
                counters.replace(review.getBook().getId(),
counters.get(review.getBook().getId())+1);
            }
            else
            {
                ratings.put(review.getBook().getId(), review.getRating());
                counters.put(review.getBook().getId(), 1);
            }
        }
        Map<Long, Double> avgRatings = new HashMap<>();
        for(Map.Entry<Long, Integer> entry : ratings.entrySet())
        {
            avgRatings.put(entry.getKey(),
Double.valueOf(entry.getValue())/Double.valueOf(counters.get(entry.getKey()
)));
        }
        avgRatings = sortByValueDesc(avgRatings);
        return  avgRatings;
    }

    @Override
    public Book createBook(Book book) {
        Book new_book =bookRepository.save(book);
        linkService.createLink(new_book.getId());
        return new_book;
    }

    @Override
    public Book updateBook(Long id, Book bookDetails) {
        Book book = bookRepository.findById(id)
                .orElseThrow(() -> new ResourceNotFoundException("Book
doesn't exist with this id: " + id));
        book.setGenre(bookDetails.getGenre());
        book.setAuthors(bookDetails.getAuthors());
        book.setPublished(bookDetails.getPublished());
        book.setSubtitle(bookDetails.getSubtitle());
        book.setTitle(bookDetails.getTitle());
        book.setDescription(bookDetails.getDescription());
        return bookRepository.save(book);
    }
```

```
    @Override
    @Transactional
    public void deleteBook(Long id) {
        Book book = bookRepository.findById(id)
                .orElseThrow(() -> new ResourceNotFoundException("Book
doesn't exist with this id: " + id));
        reviewService.deleteReviewsByBook(book);
        accountBookRepository.deleteAllByBook(book);
        linkService.deleteLinks(book);
        bookRepository.delete(book);
    }

    @Override
    public List<Book> getAllAccountBooks(Long account_id) {
        Account account =
accountRepository.findById(account_id).orElseThrow(()-> new
ResourceNotFoundException("No account with this id"));
        List<AccountBook> accountBooks =
accountBookRepository.findAccountBooksByAccount(account).orElseThrow(()->
new RuntimeException("AccountBooks not found"));
        List<Book> books = new ArrayList<>();
        for(AccountBook accountBook: accountBooks)
        {
            Book book = accountBook.getBook();
            books.add(book);
        }
        return books;
    }

    private static <K, V extends Comparable<? super V>> Map<K, V>
sortByValueDesc(Map<K, V> map) {
        List<Map.Entry<K, V>> list = new ArrayList<>(map.entrySet());
        list.sort(Map.Entry.<K, V>comparingByValue().reversed());
        Map<K, V> sortedMap = new LinkedHashMap<>();
        for (Map.Entry<K, V> entry : list) {
            sortedMap.put(entry.getKey(), entry.getValue());
        }
        return sortedMap;
    }
}
```

**ELibraryApplication.java**

```
package com.diploma.elibrary;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;

@SpringBootApplication
@EnableWebMvc
public class ELibraryApplication {

    public static void main(String[] args) {
        SpringApplication.run(ELibraryApplication.class, args);
    }
}
```

**api.service.ts**

```
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { Account } from '../entities/account';
```

```
import { Review } from '../entities/review';
import { Book } from '../entities/book';
import { Link } from '../entities/link';
import { AccountBook } from '../entities/account-book';
import { Statistics } from '../entities/statistics';

@Injectable({
  providedIn: 'root',
})
export class ApiService {

private baseUrl = "http://localhost:8080/api";

  constructor(private http: HttpClient) {}

  login(email: string, password: string) {
    return this.http.post(`${this.baseUrl}/auth`, { email, password });
  }

  createAccount(account: Account) : Observable<Object>{
    return this.http.post(`${this.baseUrl}/register`, account)
  }
  emailExists(email: string): Observable<boolean>{
    return this.http.get<boolean>(`${this.baseUrl}/checkEmail/${email}`);
  }
  getAccount(email: string): Observable<Account>{
    return this.http.get<Account>(`${this.baseUrl}/email/${email}`);
  }
  getUsers(): Observable<Account[]>{
    return this.http.get<Account[]>(`${this.baseUrl}/users`);
  }

  getReviews() : Observable<Review[]>{
    return this.http.get<Review[]>(`${this.baseUrl}/noauth/reviews`);
  }

  getReviewsByBook(book_id: number) : Observable<Review[]>{
    return
this.http.get<Review[]>(`${this.baseUrl}/noauth/reviews/${book_id}`);
  }

  addReview(account_id:number, book_id:number, review: Review) {
    return
this.http.post(`${this.baseUrl}/review/new/account/${account_id}/book/${boo
k_id}`, review);
  }

  updateReview(review: Review) {
    return this.http.put(`${this.baseUrl}/review/update/${review.id}`,
review);
  }

  deleteReview(id: number) : Observable<Map<string, boolean>>{
    return this.http.delete<Map<string,
boolean>>(`${this.baseUrl}/review/delete/${id}`);
  }

  updateAccount(account: Account) {
    return
this.http.put(`${this.baseUrl}/admin/account/update/${account.id}`,
account);
  }
```

```
  addBook(book: Book) {
    return this.http.post(`${this.baseUrl}/admin/book/new`, book);
  }
  getBooks(): Observable<Book[]> {
    return this.http.get<Book[]>(`${this.baseUrl}/noauth/books`);
  }
  searchBooks(search: string): Observable<Book[]> {
    return
this.http.get<Book[]>(`${this.baseUrl}/noauth/books/search/${search}`);
  }

  deleteBook(id: number): Observable<Map<string, boolean>>{
    return this.http.delete<Map<string,
boolean>>(`${this.baseUrl}/admin/book/delete/${id}`);
  }

  updateBook(book: Book){
    return this.http.put(`${this.baseUrl}/admin/book/update/${book.id}`,
book);
  }

  uploadBook(book_id: number, files: FormData){
    return this.http.post(`${this.baseUrl}/admin/book/upload/${book_id}`,
files);
  }

  getLinks(): Observable<Link[]> {
    return this.http.get<Link[]>(`${this.baseUrl}/links`);
  }

  getBookLink(book_id: number): Observable<Link> {
    return this.http.get<Link>(`${this.baseUrl}/noauth/link/${book_id}`);
  }

  updateLink(link: Link){
    return this.http.put(`${this.baseUrl}/admin/link/update/${link.id}`,
link);
  }

  getBestRatedBooks(): Observable<Book[]> {
    return this.http.get<Book[]>(`${this.baseUrl}/noauth/books/bestRated`);
  }

  getBookRating(book_id: number): Observable<number> {
    return
this.http.get<number>(`${this.baseUrl}/noauth/book/rating/${book_id}`);
  }

  getBook(book_id: number): Observable<Book>{
    return this.http.get<Book>(`${this.baseUrl}/noauth/book/${book_id}`);
  }

  getAccountBook(account_id: number, book_id: number):
Observable<AccountBook>{
    return this.http.get<AccountBook>(`${this.baseUrl}/account-
book/account/${account_id}/book/${book_id}`);
  }

  getBooksByAccount(account_id: number): Observable<Book[]>{
    return
this.http.get<Book[]>(`${this.baseUrl}/books/account/${account_id}`);
  }
```

```
  updateAccountBook(account_book_id: number, accountBook: AccountBook):
Observable<AccountBook>{
    return this.http.put<AccountBook>(`${this.baseUrl}/account-
book/update/${account_book_id}`, accountBook);
  }

  getAccountBooksByAccount(account_id: number): Observable<AccountBook[]>{
    return this.http.get<AccountBook[]>(`${this.baseUrl}/account-
book/account/${account_id}`);
  }

  downloadAudioFile(book_id: number){
    return this.http.get(`${this.baseUrl}/download/audio/book/${book_id}`);
  }

  downloadPdfFile(book_id: number){
    return this.http.get(`${this.baseUrl}/download/pdf/book/${book_id}`);
  }

  downloadPhotoFile(book_id: number){
    return this.http.get(`${this.baseUrl}/download/photo/book/${book_id}`);
  }

  getRecommendedBooks(account_id: number): Observable<Book[]> {
    return
this.http.get<Book[]>(`${this.baseUrl}/books/recommended/account/${account_
id}`);
  }

  updateAccountFullname(account: Account) {
    return this.http.put(`${this.baseUrl}/account/update/${account.id}`,
account);
  }

  updatePassword(account_id: number, new_password: string) {
    let formData = new FormData();
    formData.append('password', new_password);
    return this.http.put(`${this.baseUrl}/account/${account_id}/new-
password`, formData);
  }

  getStatistics(account_id: number): Observable<Statistics>{
    return
this.http.get<Statistics>(`${this.baseUrl}/statistics/account/${account_id}
`);
  }

}
```

**auth.interceptor.ts**

```
import { HttpEvent, HttpHandler, HttpInterceptor, HttpRequest } from
'@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { AuthService } from '../services/auth.service';

@Injectable()
export class AuthInterceptor implements HttpInterceptor {

  constructor(private authService: AuthService){}

  intercept(req: HttpRequest<any>, next: HttpHandler):
Observable<HttpEvent<any>> {
```

```
    const token = localStorage.getItem("token");
    if(token)
    {
      req = req.clone({
      setHeaders: {

        'Accept'       : 'application/json',
        'Authorization': `Bearer ${token}`,
      },
    });
    }
    return next.handle(req);
  }
}
```

**auth.guard.ts**

```
import { inject } from "@angular/core";
import { Router } from "@angular/router"
import { filter, firstValueFrom, map } from "rxjs";
import { CurrentUserService } from "../services/current-user.service";
import { AuthService } from "../services/auth.service";

export const authGuard = () => {
    const currentUserService = inject(CurrentUserService);
    const router = inject(Router);
    return currentUserService.currentUser$.pipe(
        filter((currentUser) => !!currentUser),
        map((currentUser) => {
          if (!currentUser || currentUser.role != 'ADMIN') {
            router.navigateByUrl('/');
            return false;
          }
          return true;
        })
    );
}

export const isLoggedIn = () => {
  const auth = inject(AuthService);
  const currentUserService = inject(CurrentUserService);
  const router = inject(Router);

  return auth.isLoggedIn$.pipe(
    filter((isLoggedIn) => isLoggedIn !== undefined),
        map((isLoggedIn) => {
          if (isLoggedIn) {
            currentUserService.currentUser$.subscribe((value) => {
              if(!value)
              {
                return true;
              }
              if(value.role === 'ADMIN')
              {
                router.navigateByUrl('/admin');
                return false;
              }
              else if(value.role === 'USER')
              {
                router.navigateByUrl('/');
                return false;
              }
              return true;
            }
```

```
            );
          }
        })
  );
}
```

**book.ts**

```
import { Link } from "./link";

export class Book {
  id: number;
  title: string;
  subtitle: string;
  authors: string;
  genre: string;
  description: string;
  published: number;
  pages: number;
  link: Link;
}
```

**main.page.component.ts**

```
import { Component,OnInit } from '@angular/core';
import { Account } from '../../entities/account';
import { CurrentUserService } from '../../services/current-user.service';
import { Book } from 'src/app/entities/book';
import { FormControl } from '@angular/forms';
import { ReplaySubject } from 'rxjs';
import { AuthService } from 'src/app/services/auth.service';

@Component({
  selector: 'app-main-page',
  templateUrl: './main-page.component.html',
  styleUrls: ['./main-page.component.css'],
})
export class MainPageComponent implements OnInit {
  account: Account;
  isSearched: boolean = false;
  isLoggedIn: boolean;
  ngOnInit(): void {
    this.getCurrentAccount();
    this.authService.isLoggedIn$.subscribe((data)=>{
      this.isLoggedIn = data;
    })
  }

  constructor(
    private currentUserService: CurrentUserService,
    private authService: AuthService
) {}

  destroy: ReplaySubject<any> = new ReplaySubject<any>();

  ngOnDestroy(): void {
    this.destroy.next(null);
    this.destroy.complete();
  }


  private getCurrentAccount() {
    this.currentUserService.currentUser$.subscribe((data) => {
      this.account = data;
```

```
    });
  }
  searchBooks: Book[];
  searchFormControl = new FormControl('');
  searchResult: string;
  getSearchedBooks(search: string) {
    this.searchResult = search;
    this.isSearched = true;
  }
}
```

**main.page.component.html**

```
<app-header></app-header>
<div class="page-content">
  <div class="background"></div>
  <div class = "search">
    <mat-form-field class="search-input">
        <mat-label>Search book</mat-label>
        <input matInput placeholder="Enter author/title/genre/publish year"
[formControl]="searchFormControl">
        <button mat-icon-button matSuffix
(click)="getSearchedBooks(searchFormControl.value)" >
            <mat-icon>search</mat-icon>
        </button>
    </mat-form-field>
  </div>
  <div *ngIf="!isSearched" class="list">
    <div class="col">
      <app-book-list [tag]="'Best rated books'"></app-book-list>
      <app-book-list *ngIf="isLoggedIn" [tag]="'Recommendations for
you'"></app-book-list>
    </div>

  </div>
  <div *ngIf="isSearched" class="list">
    <app-search-book-list [search]="searchResult"></app-search-book-list>
  </div>
  <footer>
    <p>
      Copywrited by @me
    </p>
  </footer>
</div>
```

**main.page.component.css**

```
.col{
  display: flex;
  flex-direction: column;
}
.list {
  display: flex;
  height: 80vh;
  overflow-y: auto;
  justify-content: center;
}
.search{
  padding-top: 2%;
  padding-left: 20%;
  padding-right: 20%;
}

.search-input{
  width: 100%;
```

```
}
.search-input button {
  position: absolute;
  right: 5px;
  top: 50%;
  transform: translateY(-42%);
}
.page-content {
  position: relative;
  z-index: 0;
  height: calc(100% - 64px);
  background-image: url("bg.jpg");
  background-size: cover;
  background-repeat: no-repeat;
  background-attachment: fixed;
}
.background {
  position: absolute;
  top: 0;
  left: 0;
  height: 100%;
  width: 100%;
  backdrop-filter: blur(6px);
  z-index: -1;
  background-size: cover;
  background-repeat: no-repeat;
  background-attachment: fixed;
}
footer{
  margin-top: 40px;
  font-size: 16px;
  font-style: italic;
  display: flex;
  justify-content: center;
  align-items: center;
}
* {
  scrollbar-width: auto;
  scrollbar-color: #001529 #ffffff;
}

*::-webkit-scrollbar {
  width: 14px;
}

*::-webkit-scrollbar-track {
  background: none;

}

*::-webkit-scrollbar-thumb {
  background-color: #001529;
  border-radius: 10px;
  border: 3px solid #ffffff;
}
```

**main.page.component.spec.ts**

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { MainPageComponent } from './main-page.component';

describe('MainPageComponent', () => {
  let component: MainPageComponent;
  let fixture: ComponentFixture<MainPageComponent>;
```

```
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ MainPageComponent ]
    })
    .compileComponents();

    fixture = TestBed.createComponent(MainPageComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

**Додаток 3**

**Копія презентації**


НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ
КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
КОМП’ЮТЕРНИХ СИСТЕМ


# ВЕБЗАСТОСУНОК ЕЛЕКТРОННОЇ БІБЛІОТЕКИ

Виконав: Вербовий Дмитро Станіславович

Керівник: к.т.н., старший викладач Шкурат Оксана Сергіївна

Київ – 2023

# АКТУАЛЬНІСТЬ

1. Глобальна популярність застосунків електронної бібліотеки.
2. Недосконалість наявних рішень:
a. Обмеженість додатків кількістю доступних літературних джерел.
b. Платна частина функціональних можливостей наявних застосунків.
c. Відсутність розширеного профілю користувача.
3. Необхідність накопичення, зберігання та організації різних інформаційних джерел.

# ПОСТАНОВКА ЗАДАЧІ

**Мета проєкту**: розроблення вебзастосунку електронної бібліотеки для забезпечення доступу користувачів до літературних джерел.

1. Аналіз існуючих додатків. Формування вимог до застосунку.
2. Вибір засобів реалізації програмного застосунку.
3. Розроблення вебзастосунку електронної бібліотеки.
4. Аналіз якості розробленого вебзастосунку та відповідність вимогам. Опис напрямів подальшого вдосконалення розробленого вебзастосунку.

# ІСНУЮЧІ АНАЛОГИ

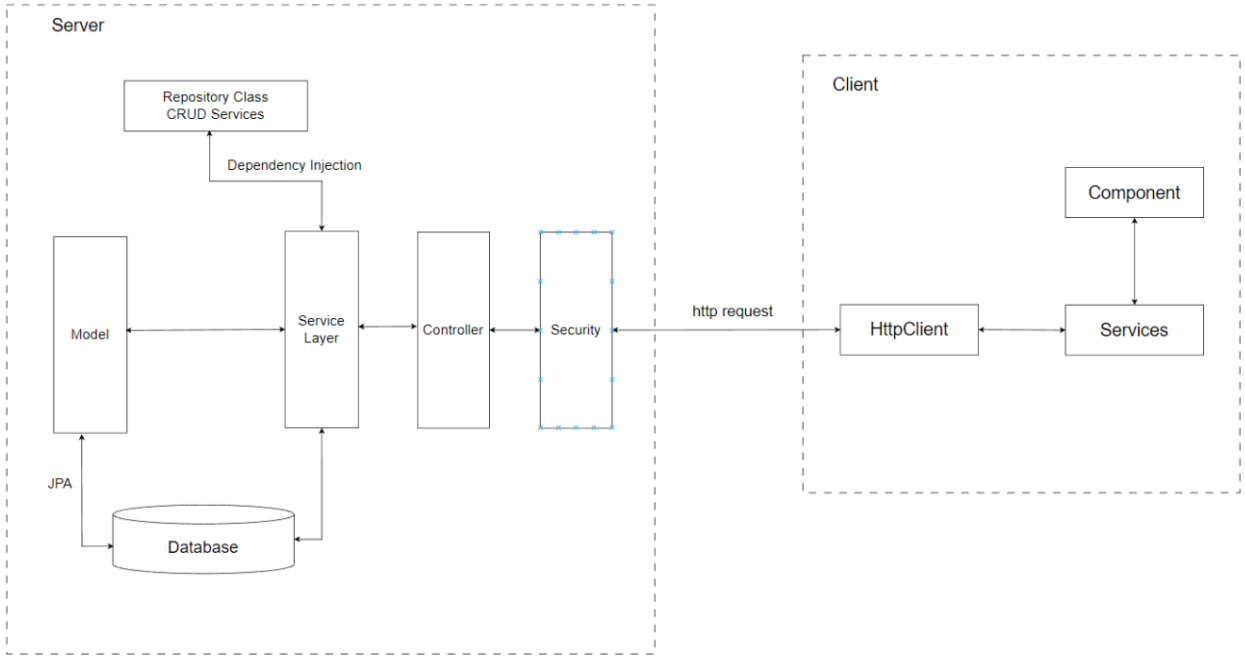
| Критерії порівняння | SCRIBD | Google books | goodreads | kobo | Apple Books |
|---|---|---|---|---|---|
| Читання онлайн/офлайн | + | + | - | + | + |
| Аудіочитання | + | - | - | + | + |
| Коментування | + | + | + | + | + |
| Рекомендації | - | - | + | - | - |
| Статистика | - | - | + | - | - |
| Безкоштовна література | - | + | + | + | - |


# ЗАСОБИ РЕАЛІЗАЦІЇ

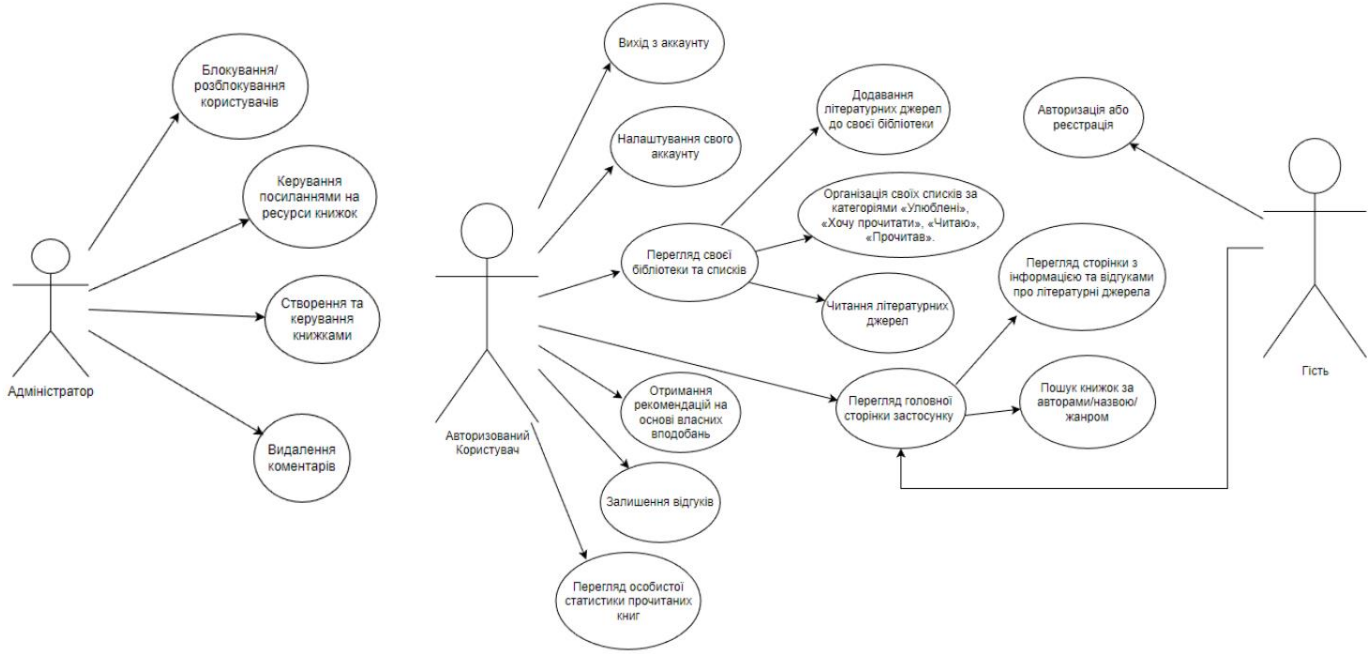

Мова програмування (сервер) — Java,
фреймворк Spring Boot.
Мова програмування (клієнт) — TypeScript,
фреймворк Angular.
База даних — PostgreSQL.



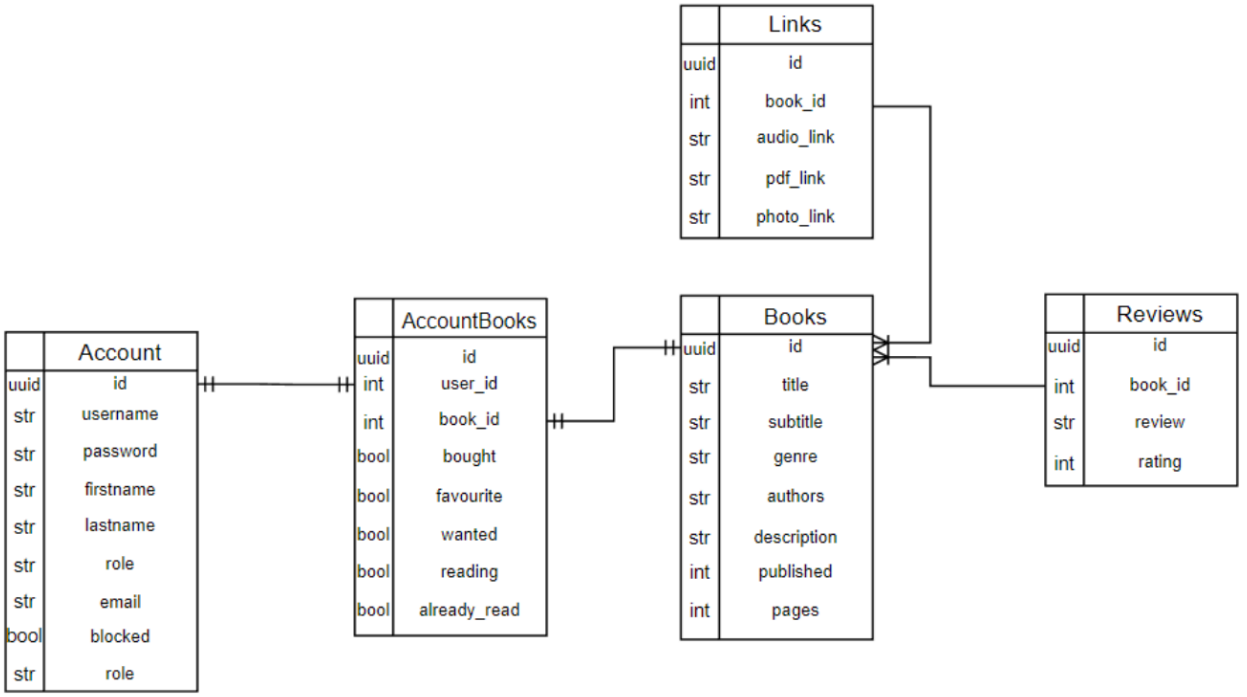

# АРХІТЕКТУРА СИСТЕМИ



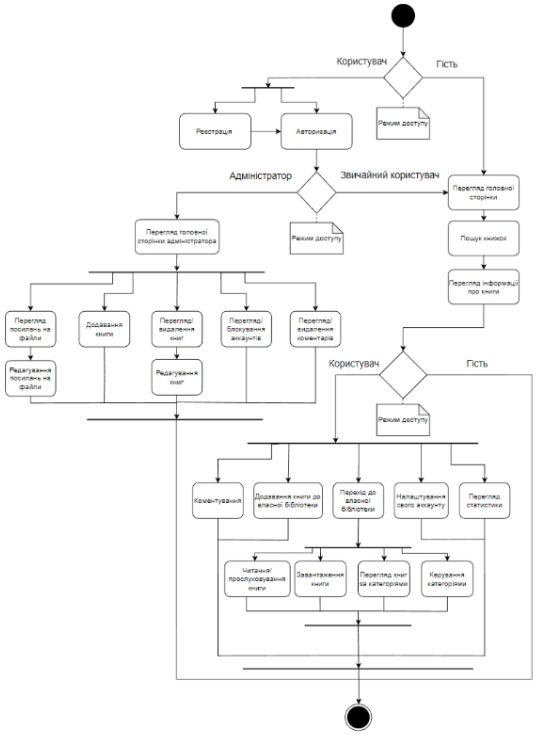
# ФУНКЦІОНАЛЬНІСТЬ ЗАСТОСУНКУ



# СТРУКТУРА БАЗИ ДАНИХ



# АЛГОРИТМ РОБОТИ ПРОГРАМИ

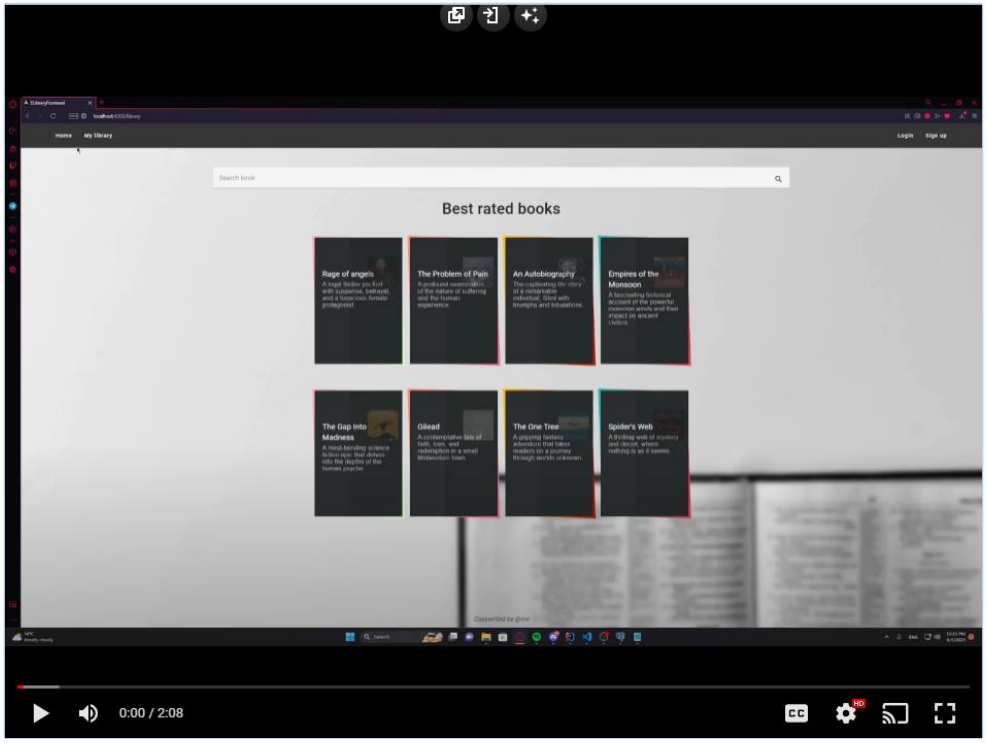



# РЕЗУЛЬТАТИ ТЕСТУВАННЯ

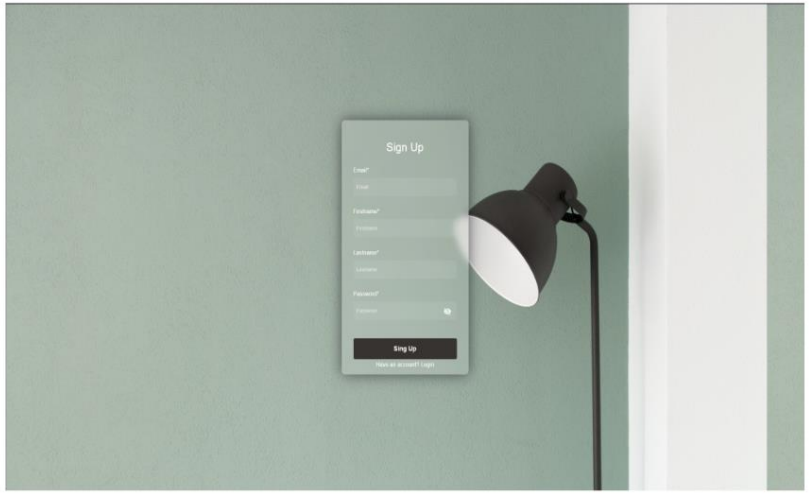

Тестування обміну даних показало, що дані коректно відображаються на клієнтській частині, коректно обробляються на серверній частині та записуються до бази даних.

Тестування інтерфейсу користувачів показало, що елементи відображаються коректно та дають змогу зручного використання функціональних можливостей вебзастосунку.

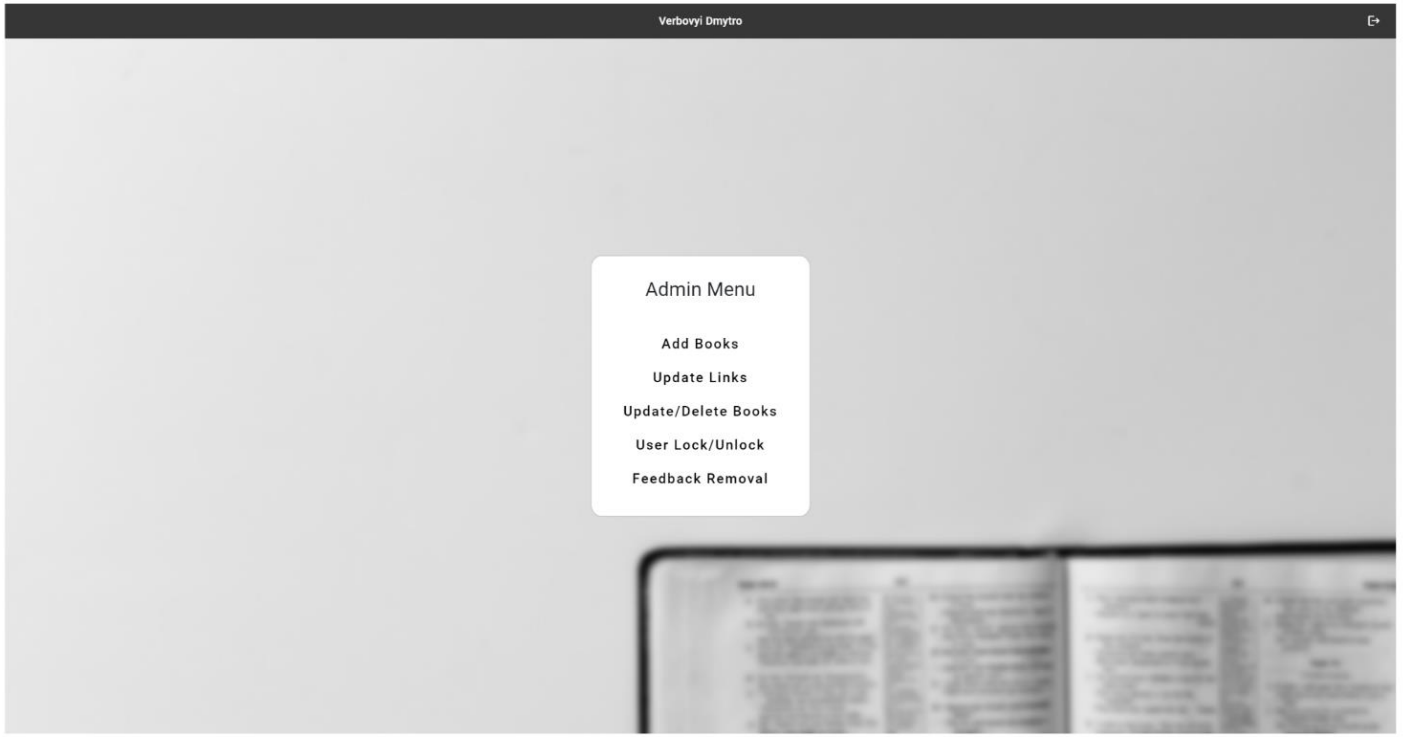
# ДЕМОНСТРАЦІЯ ВЕБЗАСТОСУНКУ



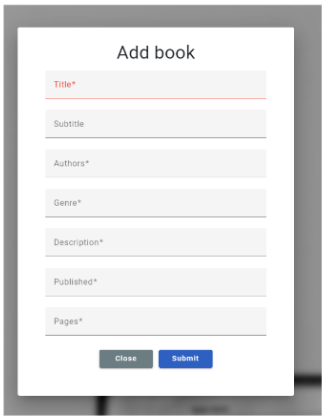
# АВТОРИЗАЦІЯ ТА РЕЄСТРАЦІЯ

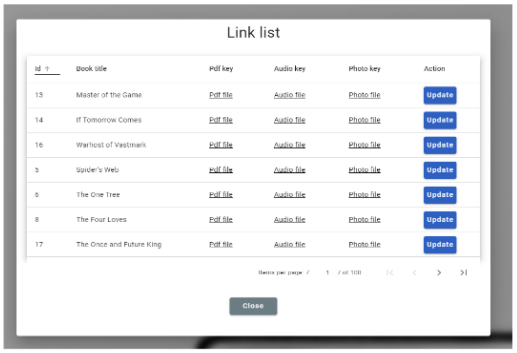



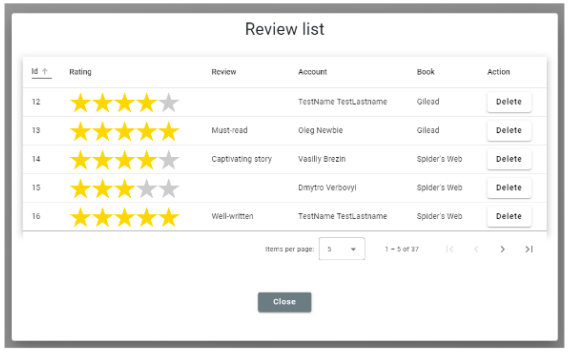
# ГОЛОВНА СТОРІНКА АДМІНІСТРАТОРА

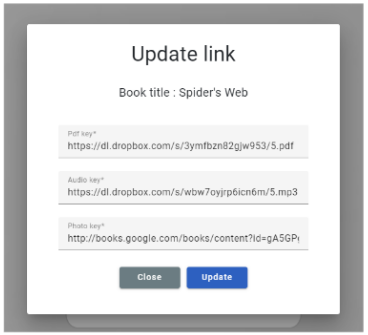


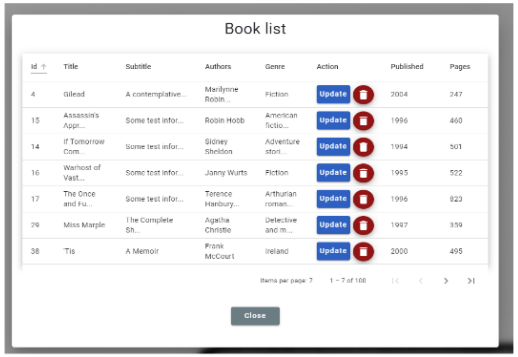
# ФУНКЦІОНАЛЬНІ ВІКНА АДМІНІСТРАТОРА

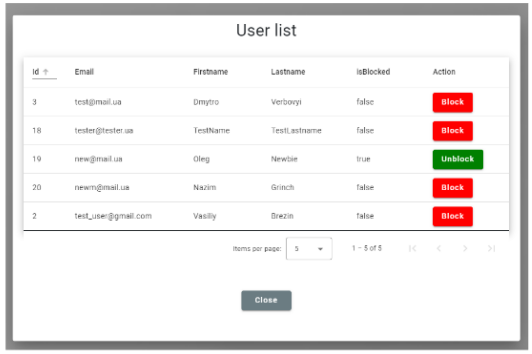


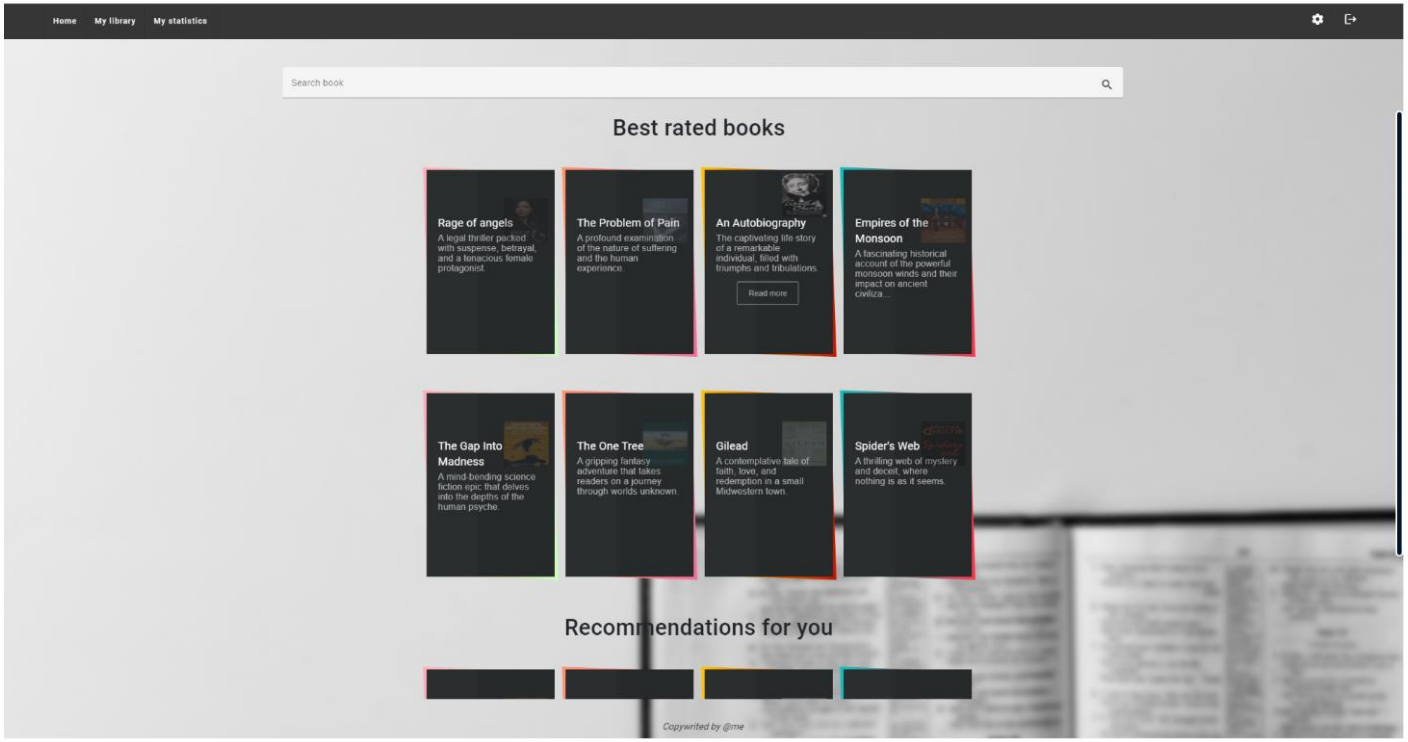



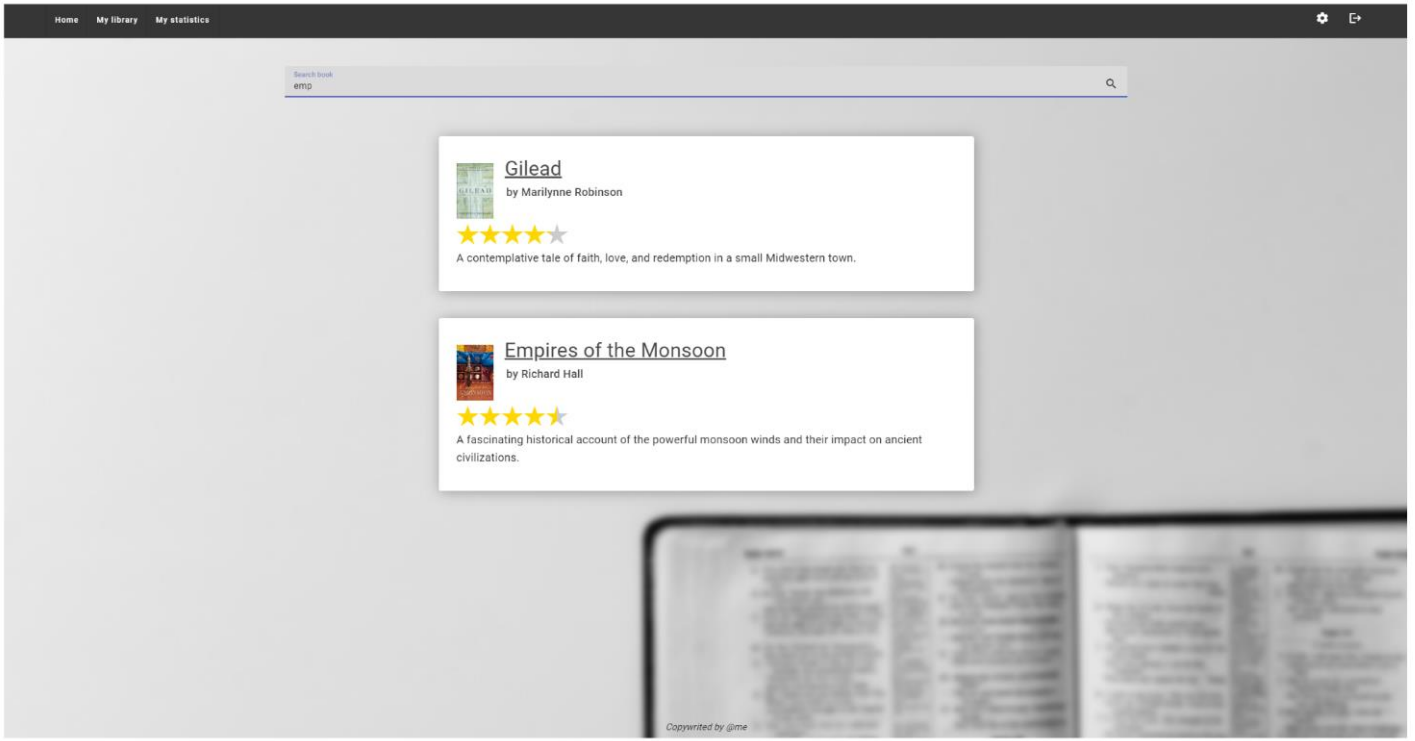



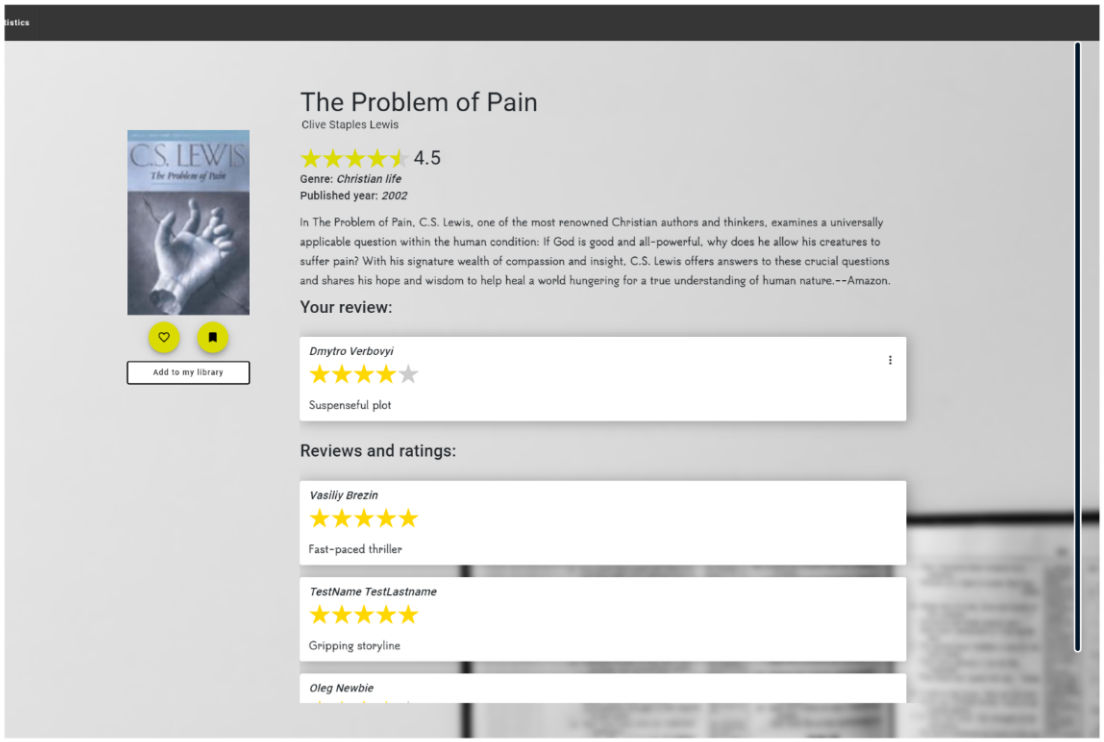
# ГОЛОВНА СТОРІНКА КОРИСТУВАЧА

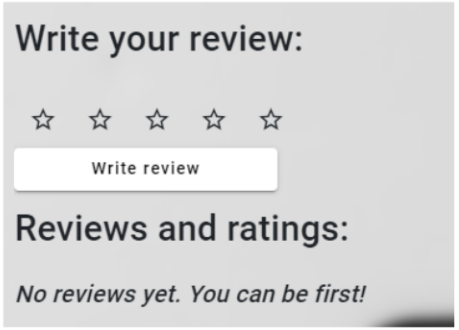


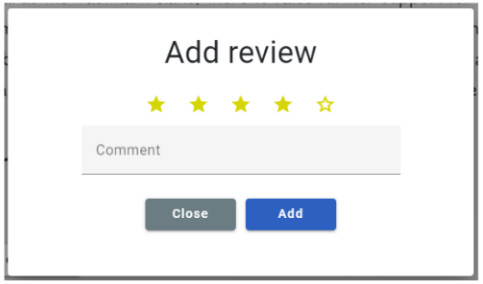
# СТОРІНКА ВІДОБРАЖЕННЯ ДАНИХ ПОШУКУ



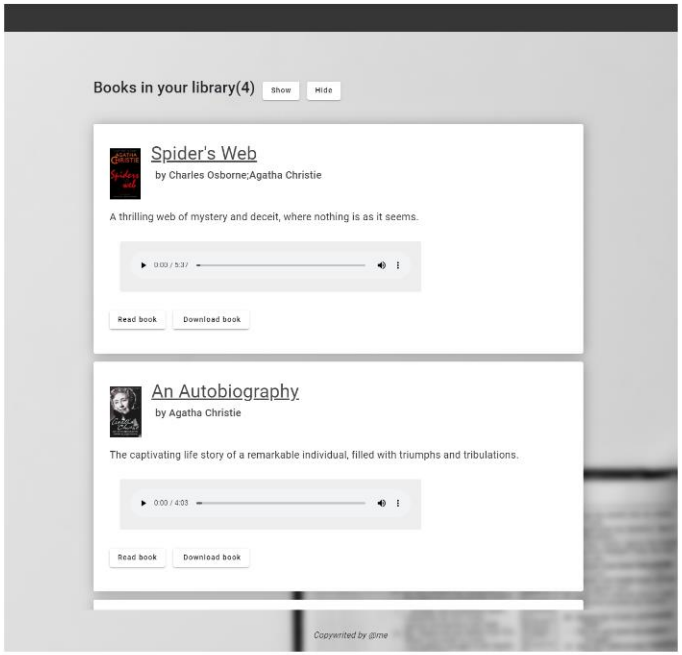
# ІНФОРМАЦІЙНА СТОРІНКА КНИГИ

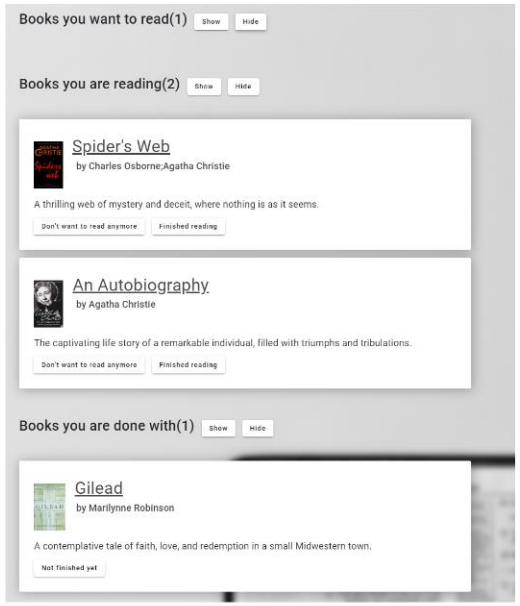


# СТОРІНКА ВЛАСНОЇ БІБЛІОТЕКИ

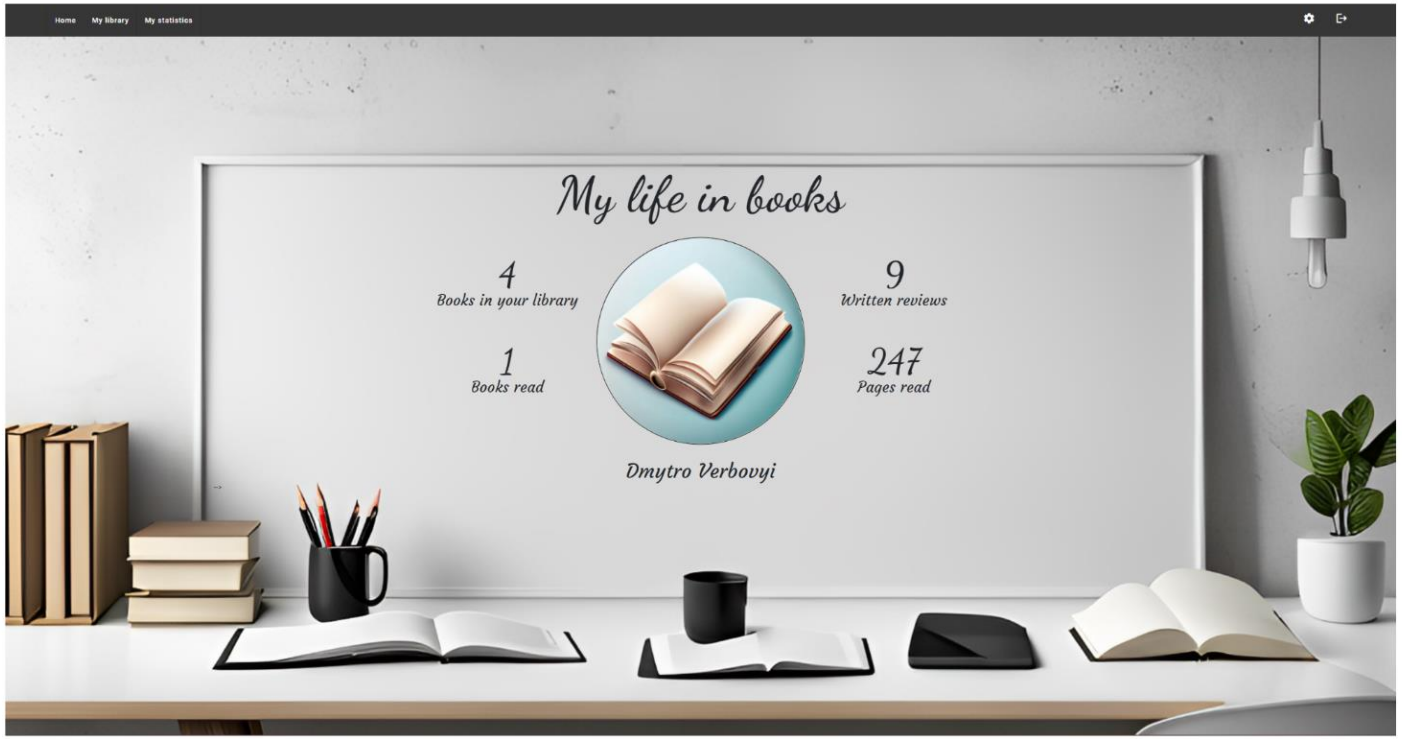




MARILYNNE ROBINSON, *GILEAD*, AND THE BATTLE FOR THE SOUL

# СТОРІНКА СТАТИСТИКИ КОРИСТУВАЧА





# Висновки

1. Проаналізовано існуючі програмні рішення, визначено їх переваги та недоліки.

2. Визначено архітектуру та засоби реалізації програмного застосунку.

3. Визначено функціональні вимоги для різних рівнів доступу в системі.

4. Розроблено вебзастосунок електронної бібліотеки відповідно до визначених вимог.

5. Проведено тестування розробленого програмного застосунку.

# Перевірка на плагіат

UNICHECK

Ім'я користувача:
**Шкурат Оксана Сергіївна**

ID перевірки:
**1015484648**

Дата перевірки:
**07.06.2023 14:45:04 EEST**

Тип перевірки:
**Doc vs Internet + Library**

Дата звіту:
**08.06.2023 20:44:58 EEST**

ID користувача:
**100007484**

Назва документа: **Вербовий_Д_С**

Кількість сторінок: **38** Кількість слів: **6989** Кількість символів: **56377** Розмір файлу: **316.38 KB** ID файлу: **1015076422**

**2.65%**
**Схожість**

Найбільша схожість: **0.37%** з Інтернет-джерелом (https://ela.kpi.ua/jspui/bitstream/123456789/35657/1/Sadrytskyi_baka..

0.82% Джерела з Інтернету 63 ..... Сторінка 40

2.52% Джерела з Бібліотеки 111 ..... Сторінка 40

**0% Цитат**

Не знайдено жодних цитат

Вилучення списку бібліографічних посилань вимкнене

**0%**
**Вилучень**

Немає вилучених джерел



# ДЯКУЮ ЗА УВАГУ!

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп'ютерних систем**

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2022 р.

# ВЕРБЗАСТОСУНОК ЕЛЕКТРОННОЇ БІБЛІОТЕКИ

## Програма та методика тестування

ДП.045440-03-91

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Оксана ШКУРАТ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Дмитро ВЕРБОВИЙ

2022

**ЗМІСТ**

## 1. ОБ’ЄКТ ВИПРОБУВАНЬ

Вебзастосунок електронної бібліотеки. Програмний застосунок забезпечує користувачів доступом до літературних джерел та адміністраторів засобами керування системи. Розроблене програмне забезпечення може бути впроваджене як у навчальних закладах та наукових установах як інструмент для забезпечення доступу до електронних ресурсів та підтримки наукової діяльності, так і у будь-яку бібліотеку.

## 2. МЕТА ТЕСТУВАННЯ

Під час виконання тестування мають бути перевірені наступні аспекти:

- функціональні можливості вебзастосунку;
- коректність передачі даних між базою, сервером та клієнтом;
- коректність відображення даних на клієнтській частині;
- зручність роботи з застосунком;
- безпечність під час використовування додатка.

## 3. МЕТОДИКА ТЕСТУВАННЯ

Тестування виконується за наступними методиками:

1. Функціональне тестування. Перевірка функціональних вимог програмного продукту, зокрема:
   - Перевірка можливості реєстрації користувача і входу до системи.
   - Перевірка функції пошуку за різними критеріями.
   - Перевірка можливості перегляду та завантаження літературних джерел.

2. Тестування інтерфейсу. Перевірка зручності та ефективності взаємодії користувача з вебзастосунком, включаючи:
   - Перевірка зовнішнього вигляду, структури та навігації сайту.
   - Перевірка правильності розташування елементів інтерфейсу та їх функціональності.
   - Перевірка відповідності дизайну інтерфейсу загальним принципам вебдизайну та уніфікації.
3. Тестування безпеки. Перевірка заходів безпеки програмного продукту, включаючи:
   - Перевірка захищеності передачі даних через застосований протокол шифрування.
   - Перевірка захисту від несанкціонованого доступу до конфіденційної інформації.
   - Перевірка доступності розробленого функціоналу відносно ролі.

## 4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування відбувається механічно, тобто все тестується вручну наступним чином:

1. Тестування функціональних можливостей додатку.
2. Тестування інтерфейсу додатку.
3. Тестування правильності передачі та обробки даних.
4. Тестування правильності обмежень полів додатку.
5. Тестування режимів доступу по ролям додатку.
6. Тестування завантаження файлів з додатку.
7. Тестування можливостей прослуховування та читання онлайн.
8. Тестування додатку у різних браузерах.

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп'ютерних систем**

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___» _____ 2023 р.

# ВЕРБЗАСТОСУНОК ЕЛЕКТРОННОЇ БІБЛІОТЕКИ

## Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Оксана ШКУРАТ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Дмитро ВЕРБОВИЙ

2023

**ЗМІСТ**

# 1. ОПИС СТРУКТУРИ ЗАСТОСУНКУ

Вебзастосунок електронної бібліотеки передбачає 3 рівні доступу: неавторизований користувач, авторизований користувач, адміністратор. Кожен рівень доступу передбачає певний набір доступних функціональних можливостей.

Далі детальніше наведено функціональні сторінки за кожним рівнем доступу у вебзастосунку:

1. Неавторизований користувач:
   - Головна сторінка.
   - Сторінка інформації після пошуку.
   - Сторінка перегляду інформації про літературне джерело.
   - Сторінка реєстрації.
   - Сторінка авторизації.
2. Авторизований користувач містить всі попередні функціональні сторінка неавторизованого користувача, а також:
   - Сторінка додавання коментарів.
   - Сторінка власних налаштувань.
   - Сторінка власної бібліотеки.
   - Сторінка читання літературного джерела.
   - Сторінка власної статистики.
3. Адміністратор:
   - Головна сторінка адміністратора.
   - Сторінка додавання книг до системи.
   - Сторінка вибору та видалення книг.
   - Сторінка оновлення книг.
   - Сторінка оновлення посилань на файли книг.
   - Сторінка перегляду та блокування користувачів.
   - Сторінка перегляду та видалення коментарів.

Кожна сторінка складається з декількох компонентів. У кожної сторінки незалежно від ролі є навігаційна панель, за допомогою якої виконуються переходи між сторінками. У кожного рівня доступу панель різниться, детальніше вона буде описана далі у описі функціоналу за відповідним рівнем доступу.

## 2. ОПИС СТОРІНОК НЕАВТОРИЗОВАНОГО КОРИСТУВАЧА

В першу чергу користувач потрапляє на головну сторінку вебзастосунку. На ній відображається навігаційна панель, рядок пошуку за категоріями та книги з найкращим рейтингом.

Навігаційна панель вміщає в себе 4 кнопки. Кнопка Home виконує перехід на головну сторінку. Кнопка My library для неавторизованого користувача заблокована, тому вона виконує перехід на сторінку авторизації. Кнопка Login виконує перехід на сторінку авторизації. Кнопка Sigh up виконує перехід на сторінку реєстрації. Вигляд навігаційного меню представлено на рис. 1.


Рис. 1. Навігаційна панель неавторизованого користувача

В рядок пошуку вводиться, що користувач хоче шукати. Пошук відбувається по входженню введеного слова до певних атрибутів книги. Для виконання пошуку потрібно натиснути на кнопку лупи у кінці пошукового рядку.

Книги з найкращим рейтингом розміщені у вигляді карток з короткою інформацією, а саме картинка обкладинки книги, її назва та короткий опис. Картки анімовані, при наведенні з’являється навігаційна кнопка для переходу до інформаційної сторінки книги. Вигляд головної сторінки представлено на рис. 2.

Після виконання пошуку знайдені книги будуть відображатись у вигляді списку з короткою інформацією та рейтингом. При натисканні на назву книги можемо перейти до інформаційної сторінки книги. Вигляд сторінки пошуку представлено на рис. 3.

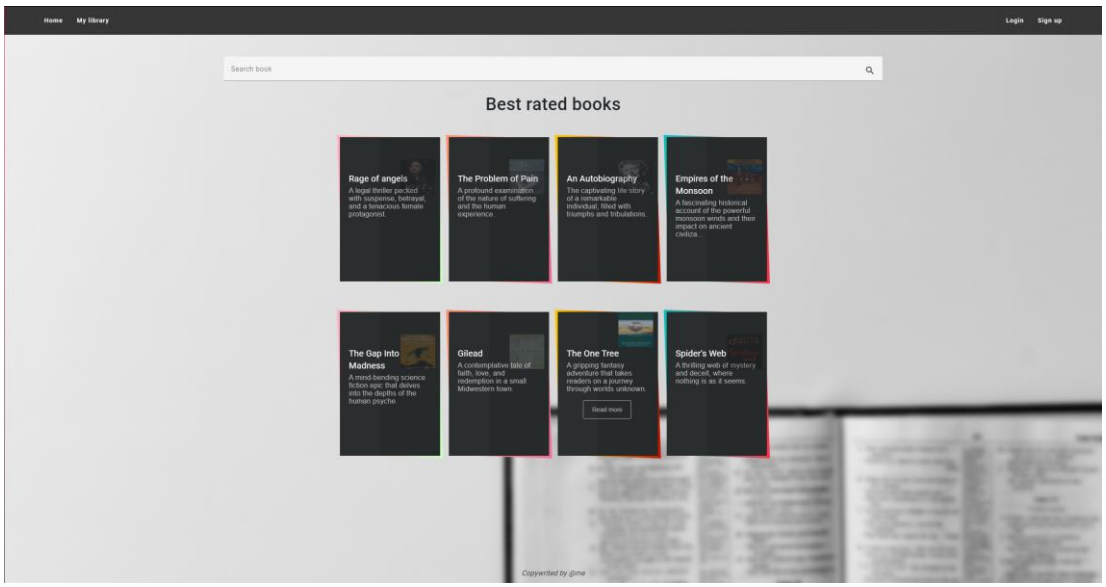



Рис. 2. Головна сторінка вебзастосунку

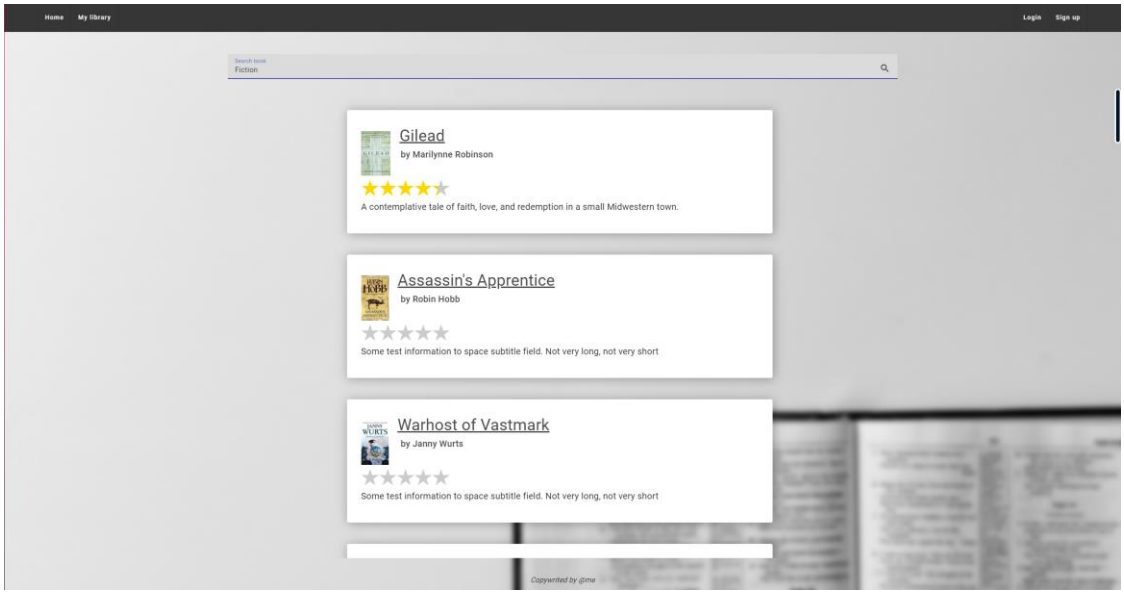



Рис. 3. Сторінка пошуку книг

Після переходу до сторінки книги можемо побачити повну інформацію про книгу, а саме титульна картинка, назва, автор, середній рейтинг та повний опис. Також можна переглянути оцінки та коментарі інших користувачів. Присутні кнопки додавання книги до категорій «Улюблені», «Хочу прочитати» та додавання для власної бібліотеки. Для неавторизованого користувача ці функціональні можливості заблоковані,

тому вони перенаправлять користувача на сторінку авторизації. Вигляд сторінки пошуку представлено на рис. 4.

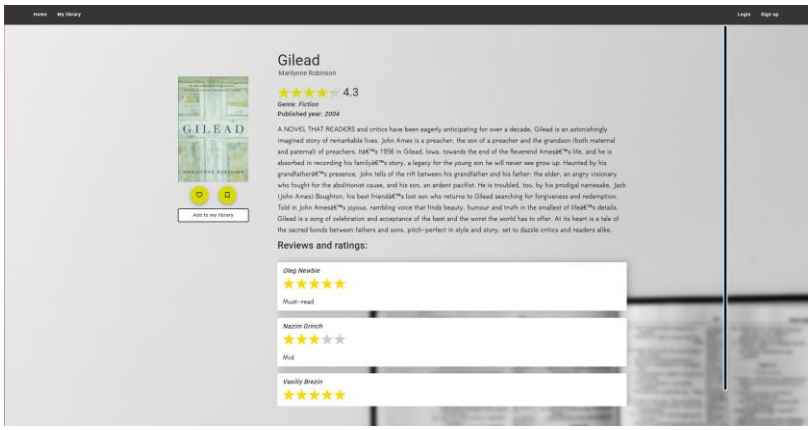



Рис. 4. Інформаційна сторінка книги

Сторінка авторизації містить у собі 2 поля для входу: email та пароль. Поле пароль за замовчуванням приховане, можна натиснути на іконку ока для його відображення. Якщо введені дані некоректні користувач отримає помилку. Якщо введений аккаунт був заблокований користувач отримає спливаюче повідомлення про це. Також є навігаційна кнопка для переходу до сторінки реєстрації. Вигляд сторінки авторизації представлено на рис. 5.

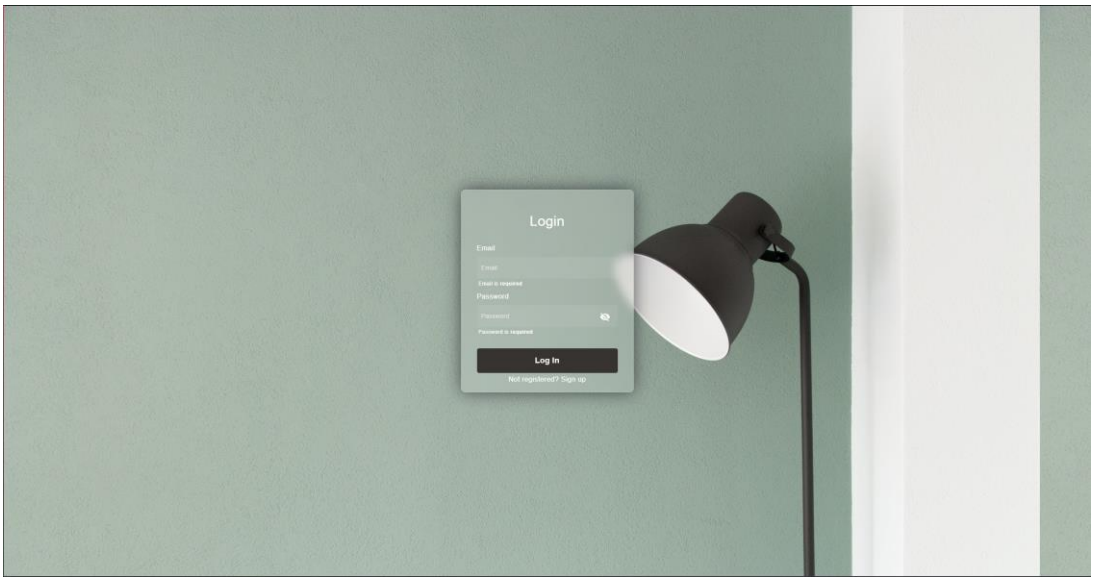



Рис. 5. Сторінка авторизації

Сторінка реєстрації містить в собі 5 обов’язкових полів для вводу: email, ім'я, прізвище, пароль. Email має бути унікальним, інакше буде помилка про вже існуючий аккаунт у системі. Також є навігаційна кнопка для переходу до сторінки авторизації. Вигляд сторінки реєстрації представлено на рис. 6.

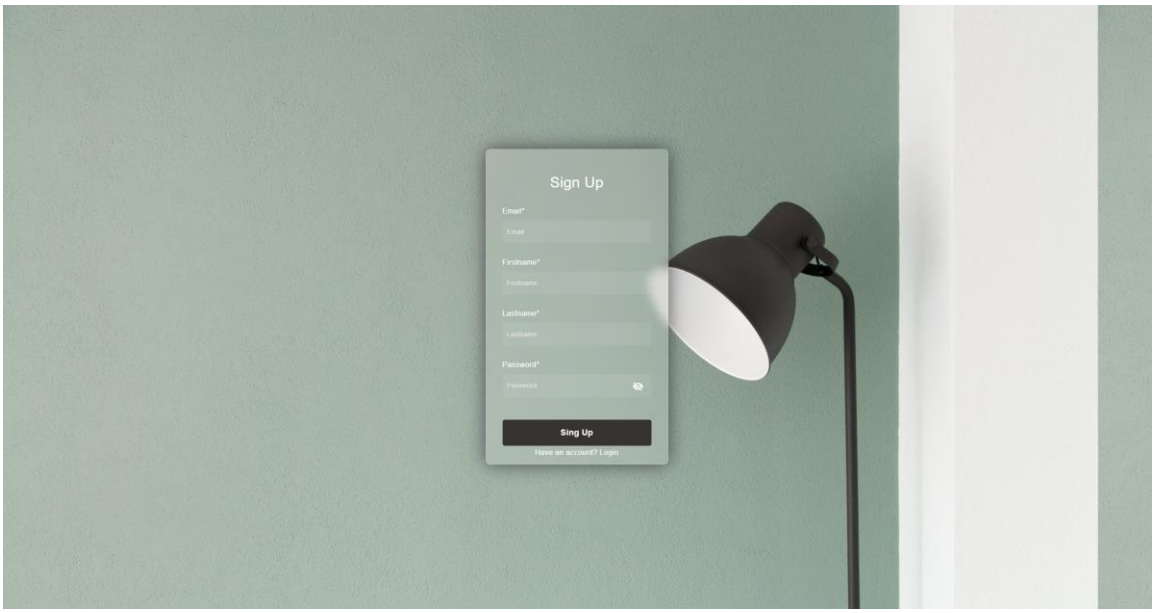



Рис. 6. Сторінка реєстрації

Після виконання реєстрації користувача буде автоматично переадресовано на сторінку авторизації, де він повинен зробити вхід до аккаунту.

# 3. ОПИС СТОРІНОК АВТОРИЗОВАНОГО КОРИСТУВАЧА

Після авторизації користувач одразу потрапляє до головної сторінки. Вона майже ідентична до неавторизованого користувача, окрім навігаційного меню та рекомендацій.

Навігаційна панель вміщає в себе 5 кнопок: кнопка Home, що виконує перехід на головну сторінку, кнопка My library, що виконує перехід до сторінки власної бібліотеки, кнопка My statistics виконує перехід на сторінку власної статистики користувача, кнопка налаштувань відкриває спливаюче вікно для зміни даних аккаунту, кнопка виходу з аккаунту. Вигляд навігаційного меню авторизованого користувача представлено на рис. 7.

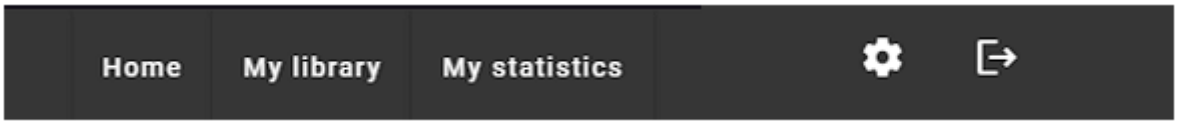



Рис. 7. Навігаційна панель авторизованого користувача

Сторінка перегляду літературного джерела додатково має кнопки для залишення відгуку. Також розблоковані функціональні можливості додавання до власної бібліотеки та категорій. Це все представлено на рис. 8.

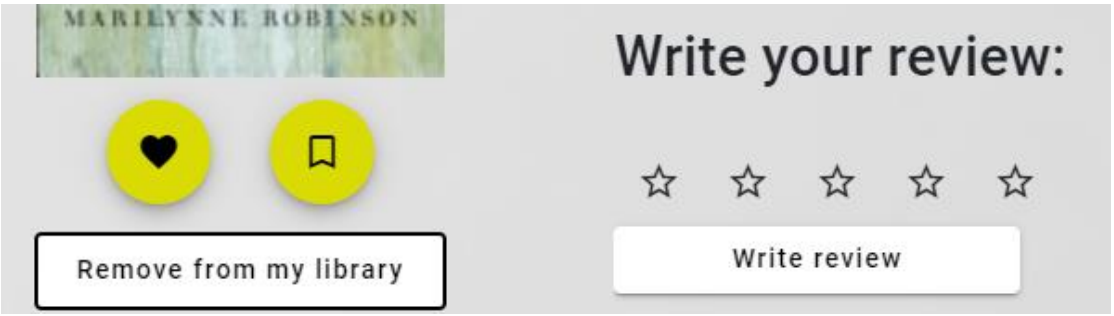



Рис. 8. Функціонал залишення відгуку

Після натиску на кнопку залишення відгуку буде доступне спливаюче вікно, де користувач може вибрати рейтинг та написати коментар. Спливаюче вікно представлене на рис. 9.

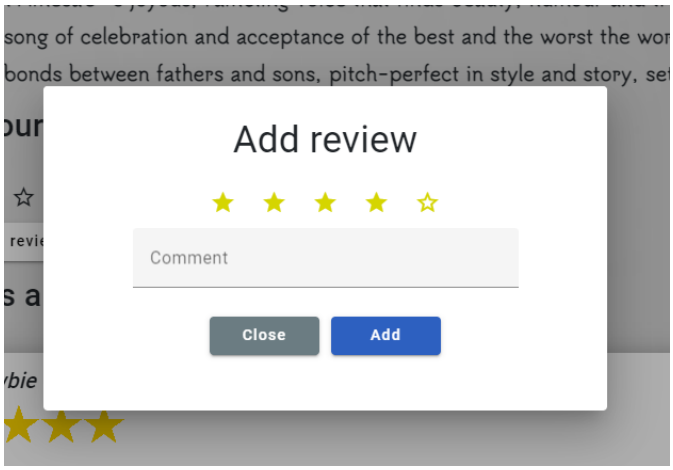



Рис. 9. Спливаюче вікно для залишення відгуку

Після додавання відгуку він буде відображений та буде можливість його змінити/видалити. Вигляд доданого відгуку представлено на рис. 10.

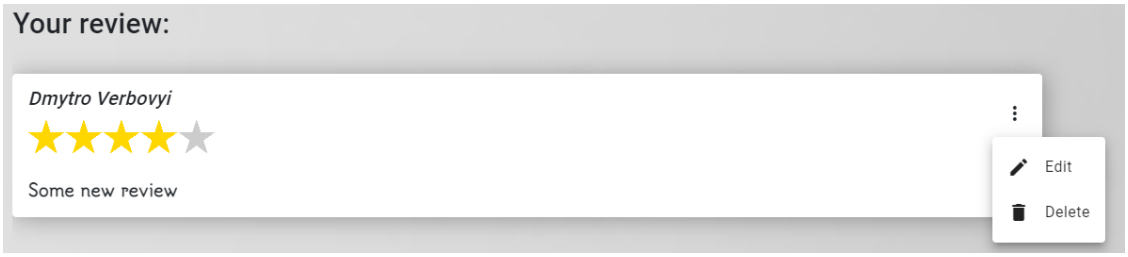



Рис. 10. Вигляд доданого відгуку

При видаленні відгуку він буде видалений з системи та функціонал відображатиметься у вигляді, представленому на рис. 8. При зміні відгуку буде відкрите спливаюче вікно ідентичне до вікна для залишення відгуку представленого на рис. 9.

Залишені користувачем відгуки будуть доступні до перегляду у всіх користувачів, на них відображаються ім’я та прізвище, рейтинг та коментар. Відгуки мають дотримуватись правил, а саме не містити нецензурної

лексики, зневажної та образливої поведінки до інших. Це може призвести до видалення відгуку або ж блокування акаунту.

Сторінка налаштувань аккаунту розроблена у вигляді спливаючого вікна, у якому можна змінити ім’я та прізвище та змінити пароль. Стандартний вигляд зображено на рис. 11.

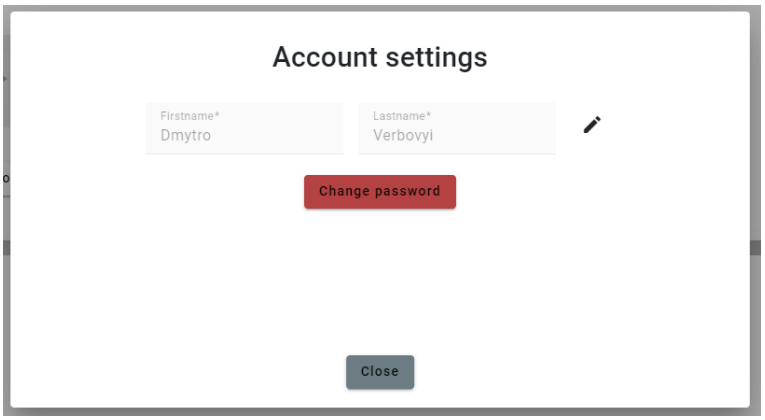



Рис. 11. Спливаюче вікно налаштувань аккаунту

Для зміни ім’я та прізвища потрібно натиснути на кнопку редагування. Після цього можна ввести нові дані та зберегти їх, або ж відмінити дію. Стадія зміни ім’я та прізвища буде представлена на рис. 12.

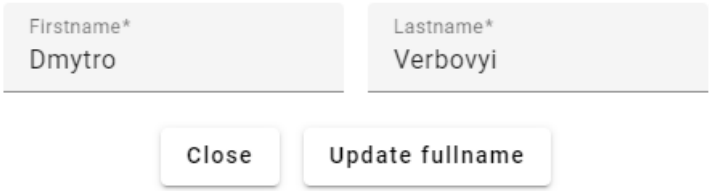



Рис. 12. Стадія зміни ім’я та прізвища акаунту

Для зміни паролю треба натиснути на кнопку зміни паролю. Після цього потрібно ввести новий пароль, підтвердити його. Далі можна зберегти або відмінити дії. Стадія зміни паролю буде представлена на рис. 13.

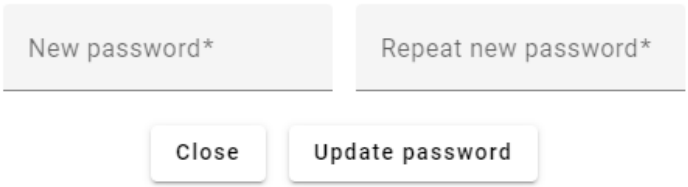



Рис. 13. Стадія зміни паролю акаунту

Після додавання книги до певних категорій або власної бібліотеки їх можна переглянути у вигляді списку на сторінці власної бібліотеки. Вона містить 5 різних категорій: «Книги у моїй бібліотеці», «Улюблені», «Хочу прочитати», «Читаю», «Прочитав». Кожну категорію можна приховати або відобразити після натиску на кнопку. Кожна категорія містить загальну коротку інформацію про книгу та можливість переходу на сторінку цієї книги після натиску на назву. Загальний короткий вигляд мають категорії «Улюблені» та «Хочу прочитати», вигляд представлено на рис. 14.

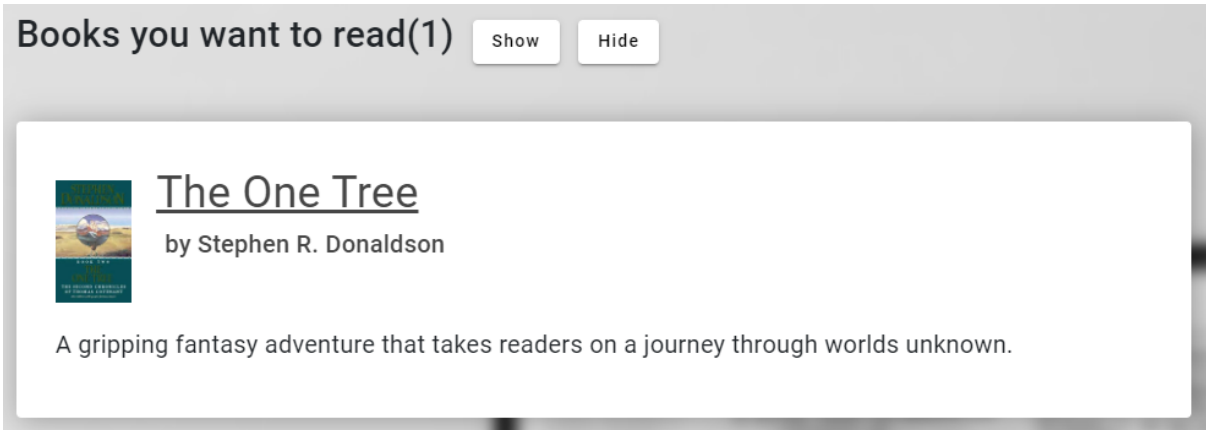



Рис. 14. Вигляд книги у категорії «Хочу прочитати»

Категорія «Книги у моїй бібліотеці» містить можливість читати книгу, прослухати аудіоверсію книги, завантажити відновідні файли для читання або прослуховування офлайн. Вигляд книги у цій категорії представлено на рис. 15.

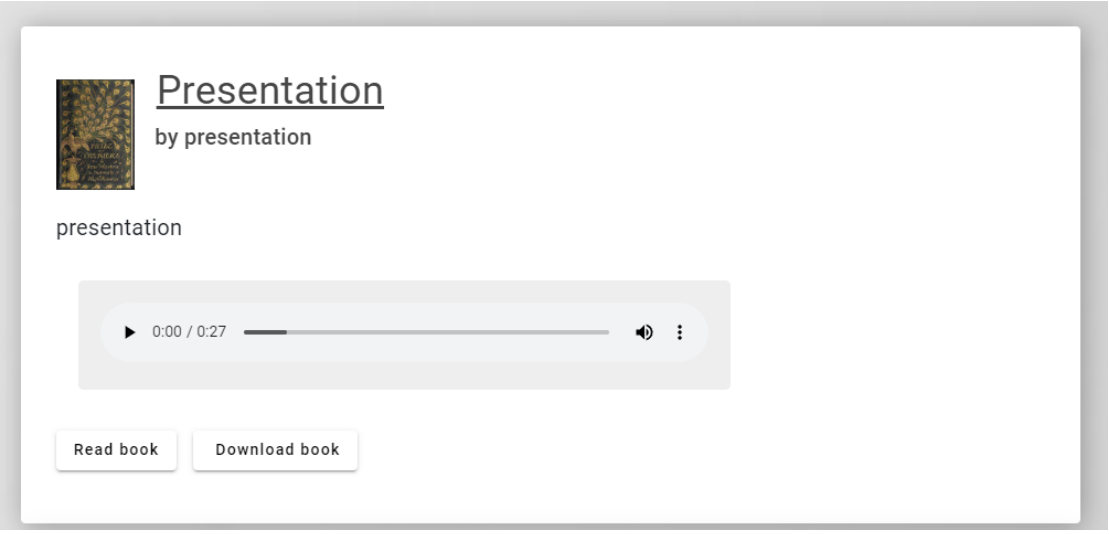



Рис. 15. Вигляд книги у категорії «Книги у моїй бібліотеці»

Після початку читання книги вона буде додана до категорії «Читаю». Книгу можна видалити з категорії або ж закінчити її читати. Вигляд книги у категорії «Читаю» представлено на рис. 16.

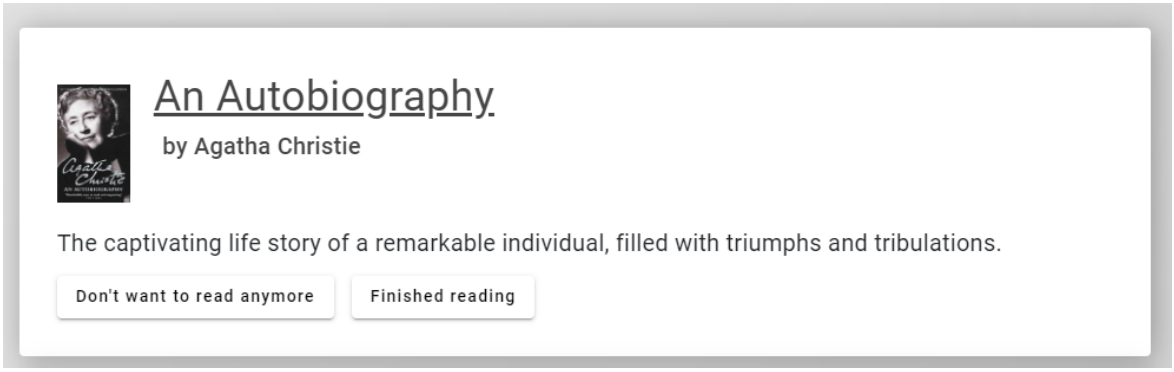



Рис. 16. Вигляд книги у категорії «Читаю»

Після завершення читання книги її можна знайти у відповідній категорії «Прочитав». Книгу можна видалити з цієї категорії. Вигляд книги у категорії «Прочитав» представлено на рис. 17.


Рис. 17. Вигляд книги у категорії «Прочитав»

На сторінці власної статистики можна переглянути певну статистику, а саме кількість книг у своїй бібліотеці, кількість прочитаних книг, кількість прочитаних сторінок, кількість залишених коментарів. Вигляд сторінки статистики представлено на рис. 18.


Рис. 18. Вигляд сторінки статистики

# 4. ОПИС СТОРІНОК АДМІНІСТРАТОРА

Адміністратор одразу потрапляє на головну сторінку для адміністратора. На ній присутні кнопки для переходу до функціональних сторінок та навігаційна панель.

Навігаційна панель містить ім’я та прізвище адміністратора, кнопку для виходу з акаунту. Вигляд навігаційного меню авторизованого користувача представлено на рис. 19.


Рис. 19. Навігаційна панель адміністратора

З головної сторінки можемо перейти до функціональних сторінок: додавання книги, редагування та видалення книги, редагування посилань на книги, зміни статусу користувачів, видалення коментарів. Всі функціональні сторінки розроблені у вигляді спливаючих вікон. Вигляд головної сторінки представлено на рис. 20.

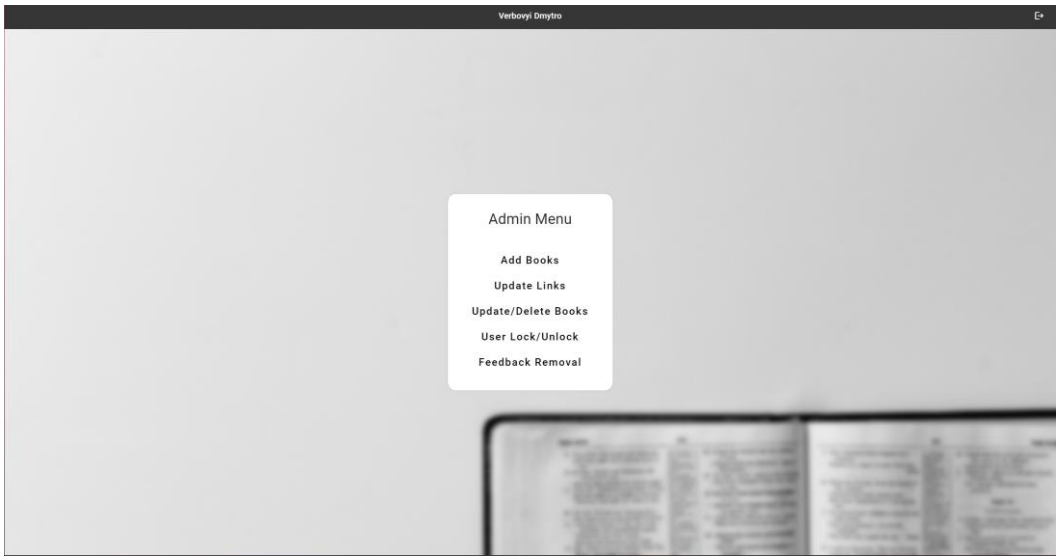



Рис. 20. Вигляд головної сторінки адміністратора

Сторінка додавання книги має 7 обов’язкових полів: назва, короткий опис, автори, жанр, детальний опис, рік видання, кількість сторінок. Вигляд вікна додавання книги представлено на рис. 21.

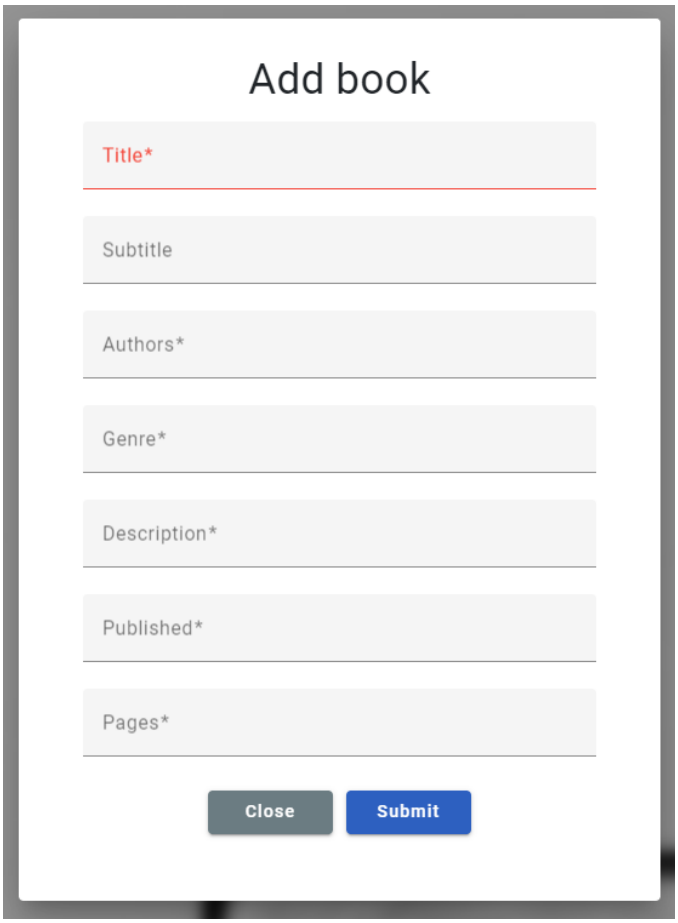



Рис. 21. Сторінка додавання книги

Сторінка редагування та видалення книг має пагіновану таблицю з книгами. Таблиця дозволяє сортувати за атрибутами книги. Вона містить атрибути книги та поле Action, у якому можна видалити книгу або перейти до функціонального вікна редагування вмісту книги. Функціональне вікно редагування вмісту книги аналогічне до вікна додавання книги, тільки має заповнені поля. Вигляд вікна редагування та видалення книг представлено на рис. 22.

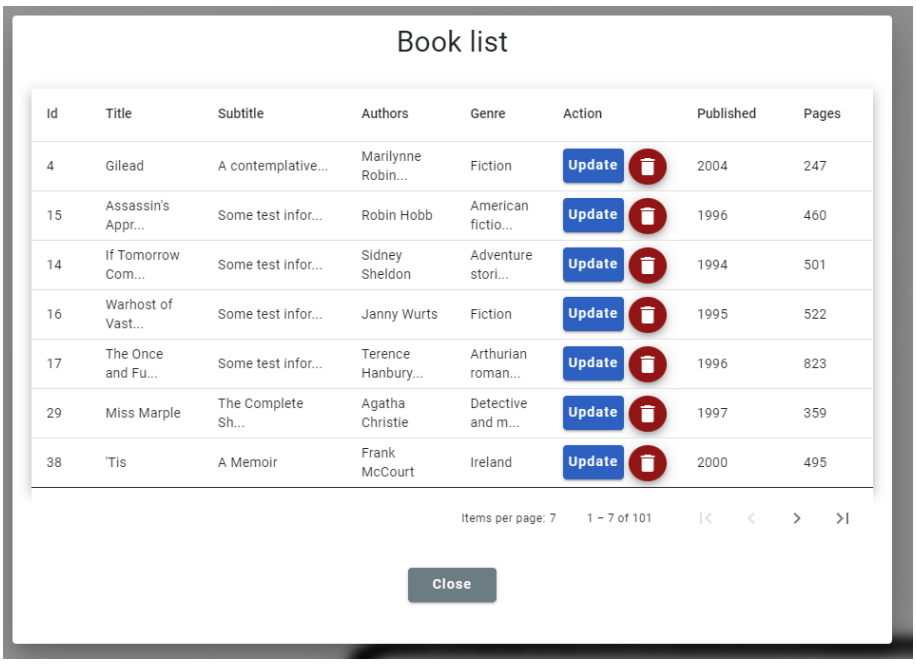



Рис. 22. Сторінка редагування та видалення книги

Сторінка редагування посилань на файли має пагіновану таблицю з книгами та їх файлами. Вона містить назву книги, поля з посиланнями на файли та поле Action, у якому можна перейти до функціонального вікна редагування посилання на файли. Функціональне вікно редагування посилань на файли представлена на рис. 23. Сторінка редагування посилань на файли представлена на рис. 24.

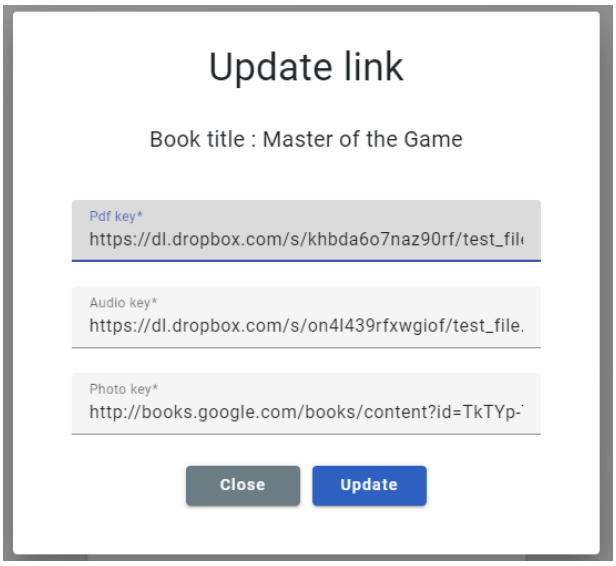



Рис. 23. Функціональне вікно редагування посилань на файли

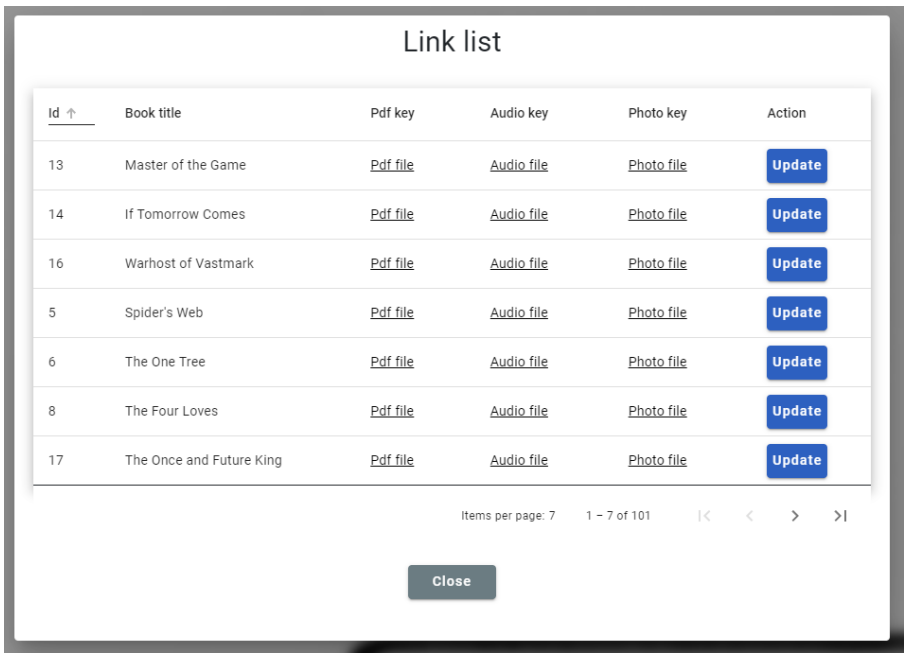



Рис. 24. Сторінка редагування посилань на файли

Сторінка зміни статусу користувачів має пагіновану таблицю з акаунтами. Вона містить атрибути акаунтів книги та поле Action, у якому можна змінити статус користувача на заблокований/розблокований. Сторінка зміни статусу користувачів представлена на рис. 25.

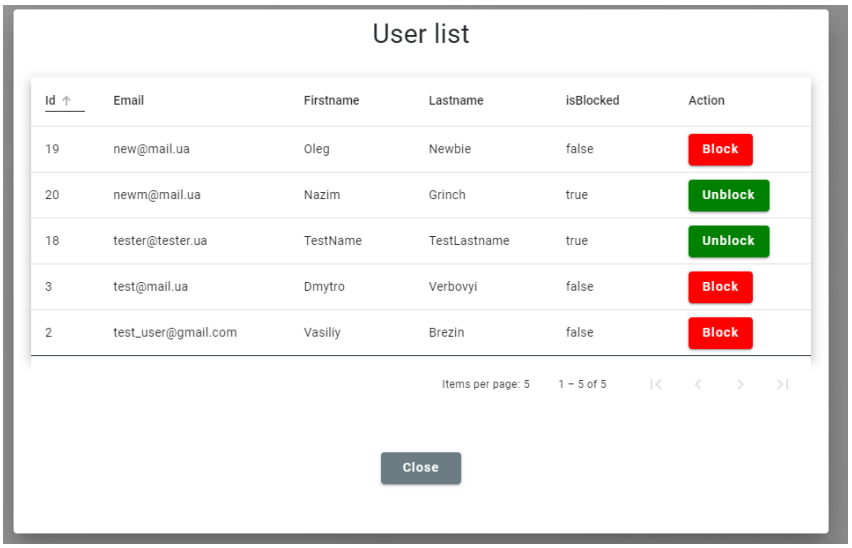



Рис. 25. Сторінка зміни статусу користувачів

Сторінка видалення відгуків має пагіновану таблицю з відгуками. Вона містить атрибути відгуків книги та поле Action, у якому можна

видалити відгук. Сторінка видалення відгуків користувачів представлена на рис. 26.

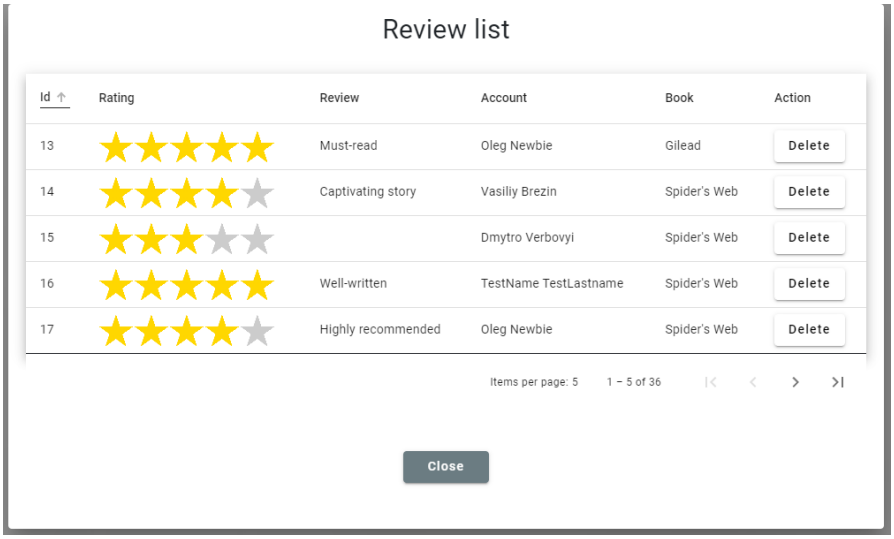



Рис. 26. Сторінка видалення відгуків