Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ЗАСТОСУНКУ «ПЕРСОНАЛЬНА БІБЛІОТЕКА» ДЛЯ КАТАЛОГІЗАЦІЇ І УПРАВЛІННЯ ПРОЧИТАНИМИ КНИГАМИ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-20-2
Уткін Є.І.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Тітова О.В.
(посада, прізвище, ініціали)

Допускається до захисту

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

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Уткіну Євгенію Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка застосунку «Персональна бібліотека» для каталогізації і управління прочитаними книгами

затверджена наказом університету від 20 травня 2024 року № 464 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 28 травня 2024 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, мова програмування JavaScript, бібліотека для розробки користувацького інтерфейсу React, відкрите середовище виконання Node.js, редактор коду Visual Studio Code.

4. Перелік питань, що потрібно опрацювати в роботі

1. Аналіз основних методів для створення застосунку «Персональна бібліотека» для каталогізації і управління прочитаними книгами.
2. Проектування системи, компонентів та бази даних.
3. Програмна реалізація застосунку «Персональна бібліотека».

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність каталогізації та управління своїми книжковими колекціями, постановка задачі, лістинги реалізації застосунку, тестові зображення.

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

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.04.2024	
2	Аналіз завдання, підбір літератури	08.04.24-12.04.24	
3	Аналіз літератури з досліджуваної проблеми	13.04.24-15.04.24	
4	Аналіз технічних засобів	16.04.24-21.04.24	
5	Розробка методу	22.04.24-06.05.24	
6	Програмна реалізація	07.05.24-22.05.24	
7	Оформлення пояснювальної записки	23.05.24-26.05.24	
8	Перевірка на плагіат	26.05.24	
9	Рецензування	27.05.24	
10	Підготовка презентації та доповіді	28.05.24-03.06.24	
11	Занесення роботи в електронний архів	06.06.24	
12	Попередній захист кваліфікаційної роботи	06.06.24	

Дата видачі завдання 8 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Тітова О.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 69 с., 6 табл., 22 рис., 33 джерела.

КОРИСТУВАЧ, КНИГА, ЗАСТОСУНОК, JAVASCRIPT, TYPESCRIPT, JUP, VISUAL STUDIO CODE, REACT, STYLED-COMPONENTS, REDUX-TOOLKIT, MONGODB, NODE.JS, EXPRESS.

Об’єктом роботи є каталогізація та управління прочитаними книгами.

Метою даної кваліфікаційної роботи є розробка застосунку, що призначений для легкої та ефективної каталогізації прочитаних книг та управління особистою бібліотекою користувача. Також застосунок дозволить користувачам планувати своє майбутнє читання, проводити тренування читання, відстеження прогресу по дням, книжкам, сторінкам та вести записи про книги, які вони бажають прочитати.

У процесі розробки сервісу були використані сучасні технології, бібліотеки та фреймворки, такі як React для розробки застосунку, react-hook-form для ствоерення форм, Node.js для розробки серверної частини, Express для створення API та MongoDB для зберігання даних. Було використано бібліотеку валідації данних Jup, що забезпечила коректну обробку введених даних. Також для поліпшення читання коду, зменшуючи вірогідність помилок під час розробки і підтримки проектів використовується Typescript.

USER, BOOK, APPLICATION, JAVASCRIPT, TYPESCRIPT, JUP, VISUAL STUDIO CODE, REACT, STYLED-COMPONENTS, REDUX-TOOLKIT, MONGODB, NODE.JS, EXPRESS.

The object of the work is the cataloging and management of read books.

The purpose of this qualification work is to develop an application designed for easy and effective cataloging of read books and management of the user's personal library. The app will also allow users to plan their future reading, conduct reading exercises, track progress by day, book, page, and keep records of the books they want to read.

In the service development process, modern technologies, libraries and frameworks were used, such as React for application development, react-hook-form for creating forms, Node.js for developing the server part, Express for creating APIs and MongoDB for data storage. The Jup data validation library was used, which ensured correct processing of entered data. Typescript is also used to improve code readability, reducing the likelihood of errors during project development and support.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

SPA – Single Page Application (односторінковий застосунок)

MPA – Multi Page Application (багатосторінковий застосунок)

UI – User interface (користувацький інтерфейс)

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

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

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

JSX – JavaScript XML (розширення синтаксису JavaScript, яке виглядає як XML)

SEO – Search Engine Optimization (вдосконалення сайту для пошукових систем та користувачів, що відбувається шляхом проведення заходів по внутрішній та зовнішній оптимізації. Метою SEO оптимізації є збільшення трафіку на сайт та його подальшої монетизація або перетворення в клієнтів)

ВСТУП

У сучасному світі, коли доступ до книг стає все легшим, важливість ефективного управління та каталогізації прочитаних творів набуває все більшого значення. Персональна бібліотека створює можливість для читачів зберігати інформацію про свої читацькі переживання та відстежувати свої літературні вподобання.

Завдяки сучасним технологіям, таким як смартфони, планшети і комп'ютери, книги стали доступнішими, ніж раніше. Цей розвиток викликає потребу у зручних та ефективних застосунках для організації прочитаних та бажаних книг, щоб кожен читач міг з легкістю керувати своїм читацьким досвідом і відкривати нові літературні світи.

Мета розробки цього застосунку полягає в створенні зручного та ефективного інструменту для організації особистої бібліотеки та каталогізації прочитаних книг. Застосунок буде надавати користувачам можливість легко додавати нові книги до своєї колекції, а також вносити відгуки та оцінки, щоб зберігати свої враження та рекомендації для майбутніх читань.

Актуальність даної теми полягає в тому, що все більше людей шукають зручні та ефективні способи для каталогізації та управління своїми книжковими колекціями, а також прагнуть оптимізувати час, витрачений на вибір і планування свого читання. Це створює потребу в інноваційних рішеннях, які дозволяють користувачам легко вести записи про прочитані книги та планувати майбутнє читання.

В цілому, розробка застосунку «Персональна бібліотека» для каталогізації і управління прочитаними книгами є актуальною та корисною ідеєю для тих, хто прагне зберегти систематизований підхід для свого читання, заощадити час на вибір наступної книги, а також збільшити ефективність своїх читацьких звичок.

1 АНАЛІЗ ОСНОВНИХ МЕТОДІВ ДЛЯ СТВОРЕННЯ ЗАСТОСУНКУ «ПЕРСОНАЛЬНА БІБЛІОТЕКА» ДЛЯ КАТАЛОГІЗАЦІЇ І УПРАВЛІННЯ ПРОЧИТАНИМИ КНИГАМИ

1.1 Аналіз перспективності створення вебзастосунку

Вебзастосунок – це програмне забезпечення, яке користувач може використовувати через браузер за допомогою інтернет-з'єднання. Ці застосунки не потребують завантаження та інсталяції на персональний комп'ютер користувача, оскільки всі необхідні дані обробляються на сервері та відображаються у браузері.

Вебзастосунки стали особливо популярними завдяки їх зручності та доступності: користувачам достатньо мати доступ до інтернету, щоб мати можливість користуватися всіма функціями програми з будь-якого пристрою. Це робить вебзастосунки ідеальними для бізнес-додатків, освітніх платформ, онлайн-магазинів та багатьох інших застосувань [1].

Види вебзастосунків можна поділити на дві найпопулярніші категорії:

– односторінкові застосунки (SPA – Single Page Application): це інтерактивні застосунки, які функціонують на одній вебсторінці, забезпечуючи плавність і швидкість роботи за рахунок динамічного оновлення контенту без повного перезавантаження сторінки. Вебсайт може складатися лише з однієї сторінки, але для того, щоб бути SPA, він має бути інтерактивним і здатним динамічно змінювати вміст без зміни URL адреси. Для створення SPA зазвичай використовують JavaScript бібліотеку React або фреймворки, такі як Vue, або Angular;

– багатосторінкові застосунки (MPA – Multi Page Application): традиційні вебпрограми, які завантажують нові HTTP-сторінки при кожній взаємодії користувача з сайтом, що може уповільнювати процес, особливо при

поганих інтернет-з’єднаннях або ненадійному хостингу. Типовими прикладами MPA є інтернет-магазини, такі як Rozetka або Amazon [2].

Архітектура програмного забезпечення може бути реалізована в різних стилях та конфігураціях, таких як клієнт-сервер, багатокомпонентна, монолітна, або мікросервісна.

Архітектура клієнт-сервер є ключовим архітектурним шаблоном для розробки програмного забезпечення, особливо у контексті розподілених мережних застосунків. Ця модель включає взаємодію та обмін даними між клієнтськими та серверними компонентами. В такій архітектурі взаємодія між клієнтом та сервером є фундаментальною, і кожен елемент виконує специфічну роль [3].

Сервер – це система, що надає ресурси або сервіси. Він зазвичай управляє базами даних і ресурсами, які спільні для багатьох користувачів. Сервер чекає та обробляє запити від клієнтів, надаючи їм необхідні дані чи послуги. Сервери зазвичай мають високу обчислювальну потужність, місткість пам’яті та широкосмугові мережеві з’єднання.

Клієнт – це пристрій або програма, що використовує ресурси або сервіси, надані сервером. Клієнти ініціюють запити до серверів, чекаючи відповіді та взаємодіючі з отриманими даними [4, 5].

Основні характеристики архітектури клієнт-сервер включають:

– модульність: система поділена на два окремі компоненти: клієнт (інтерфейс користувача) та сервер (управління даними), що полегшує розробку та обслуговування;

– централізоване управління: вся основна логіка та обробка даних зосереджені на сервері, що спрощує управління та оновлення системи.

– масштабованість: легше масштабувати розподілені системи, додаючи нові сервери або оновлюючи існуючі;

– взаємодія: клієнти взаємодіють із сервером через запити та отримують відповіді, що сприяє асинхронній роботі та зменшує завантаженість мережі;

– безпека: централізація даних на сервері спрощує впровадження та управління політиками безпеки.

Отже, оптимальним вибором буде використання архітектури клієнт-сервер разом з концепцією Single Page Application для розробки застосунку «Персональна бібліотека» для каталогізації та управління прочитаними книгами [6, 7].

1.2 Огляд платформ для каталогізації і управління прочитаними книгами

В цьому розділі розглядаються найпопулярніші платформи, які спрощують користувачам каталогізацію і управління прочитаними книгами. У світі існує чимало платформ для таких послуг, але обговоримо найпопулярніші з них: Goodreads, Libib та Book Catalogue.

У подальших розділах буде проведено глибокий аналіз кожної з цих платформ, детально розглядається їх ключові характеристики та обговорюються переваги, які вони пропонують своїм користувачам [8].

1.2.1 Goodreads

Goodreads є одним із найбільших соціальних каталогізаторів книг у світі, який дає можливість користувачам відстежувати книги, які вони читають, хочуть прочитати чи прочитали. Платформа славиться своєю широкою соціальною компонентою, яка включає спільноти, обговорення, опитування та рекомендації, а також можливість встановлювати читацькі цілі. Goodreads також надає доступ до великої кількості рецензій та рейтингів, що робить його ідеальним місцем для тих, хто шукає глибокі огляди книг і рекомендації.

Крім того, платформа пропонує інтеграцію з багатьма електронними книжковими магазинами, що полегшує покупку прочитаних книг. Ось деякі переваги та недоліки Goodreads [9]:

Переваги:

– велика база даних: Goodreads характеризується обширною базою книжок для зручного пошуку та додавання у колекції користувачів;

– велика база читачів: велика спільнота читачів, яка дозволяє обмінюватися рецензіями та рекомендаціями;

– рейтинги та відгуки: відгуки на Goodreads надають глибоке розуміння кожної книги, а інтеграція з соціальними мережами полегшує обмін читацькими враженнями;

– рекомендації: Goodreads виступає в ролі персонального літературного консультанта, пропонуючи індивідуальні рекомендації, що базуються на історії читання та вподобаннях;

– соціальний аспект: користувачі можуть з легкістю обмінюватися враженнями та обговорювати улюблені твори, що робить цю платформу не лише інструментом для управління особистою бібліотекою, але й центром літературного співтовариства.

Недоліки:

– UI: користувачі зазначають, що інтерфейс (UI) на Goodreads вимагає певного часу для звикання та може здатися перевантаженим, особливо для тих, хто лише починає свій шлях із цим сервісом;

– веборієнтованість: Goodreads визначається переважно як вебзастосунок платформа, з відзначеною перевагою у її браузерній версії, тоді як мобільний застосунок, хоч і функціональний, все ж таки має певні обмеження у можливостях;

– відсутність офлайн-режиму: однією з істотних відмінностей є відсутність офлайн-функціоналу, що унеможливлює доступ до певних особливостей сервісу при відсутності інтернету.

Навіть з урахуванням певних обмежень, Goodreads продовжує бути відданим помічником для упорядкування особистих бібліотек і обміну думками про книжки у читацькій спільноті.

На рисунку 1.1 показано застосунок Goodreads.

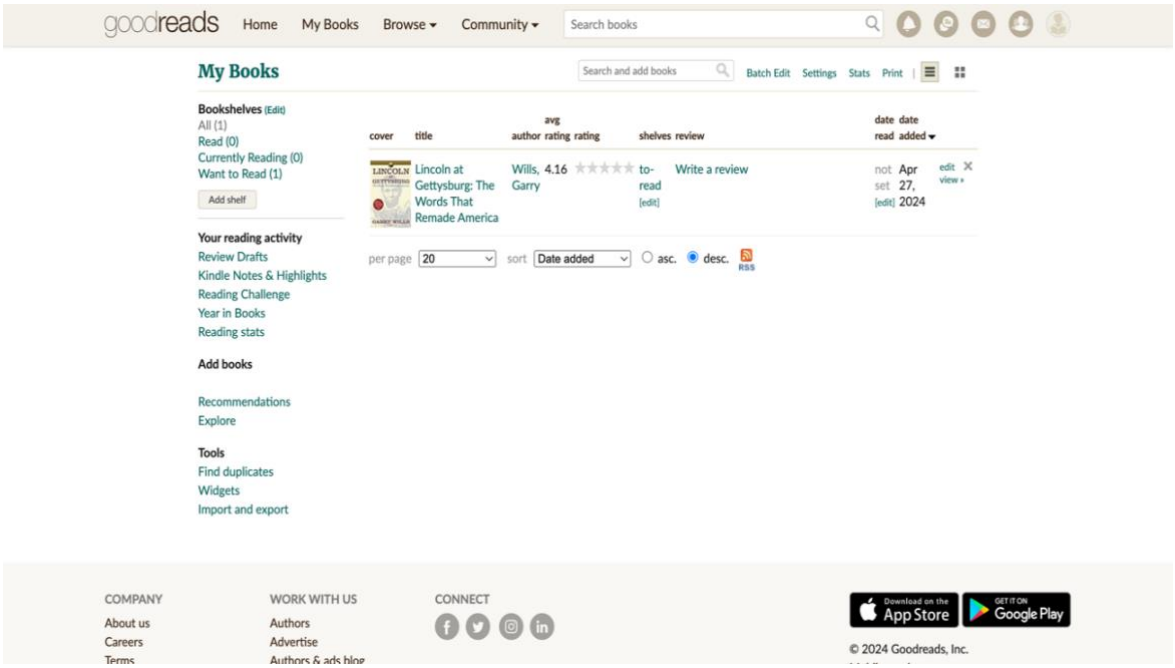

Рисунок 1.1 – Інтерфейс головної вебсторінки Goodreads

1.2.2 Libib

Libib також є вебзастосунком та мобільним додатком, призначеним для каталогізації та організації книжкових колекцій. З огляду на її функціонал, Libib стає ідеальним вибором для тих, хто шукає більш приватний та інтегрований підхід до управління своєю колекцією. Ось огляд переваг та недоліків Libib [10]:

Переваги:

– синхронізація в хмарі: завдяки хмарній синхронізації, користувачі можуть легко отримувати доступ до своїх колекцій з будь-якого пристрою, забезпечуючи надійне оновлення і збереження даних;

– Libib пропонує широкі можливості для управління колекціями різних медіа, від книг до фільмів та музики, що робить його цінним ресурсом для організації різних типів зібрань;

– сканування штрих-кодів: функція сканування штрих-кодів значно спрощує додавання нових книг до колекцій, автоматизуючи введення інформації та прискорюючи процес каталогізації;

– каталогізація та організація: застосунок надає можливість додавати книги в різні категорії, додавати теги, створювати список бажань та слідкувати за статусом прочитання;

– UI та дизайн: Libib пропонує користувачам простий і зрозумілий інтерфейс. Це робить застосунок зручним навіть для тих, хто не дуже обізнаний з цифровими технологіями, полегшуючи організацію особистих колекцій завдяки легкому доступу та простоті використання.

Недоліки:

– обмежена безкоштовна версія: головним недоліком є наявність обмеженої безкоштовної версії, яка включає тільки базові функції, що може бути не зручним для деяких користувачів;

– обмежена спільнота користувачів: у порівнянні з іншими застосунками, Libib має відносно малу спільноту користувачів, що може обмежувати обмін рекомендаціями та взаємодії, важливі для збагачення користувацького досвіду.

– обмежене відображення великих колекцій: якщо у користувача велика бібліотека, Libib може стати менш ефективним, оскільки обробка великих колекцій може займати більше часу, а інтерфейс може не підтримувати одночасне відображення великої кількості елементів.

Незважаючи на певні обмеження Libib залишається популярним інструментом для систематизації книжкових колекцій, пропонуючи інтуїтивні функції управління, що забезпечують зручність у користуванні.

На рисунку 1.2 показано застосунок Libib.

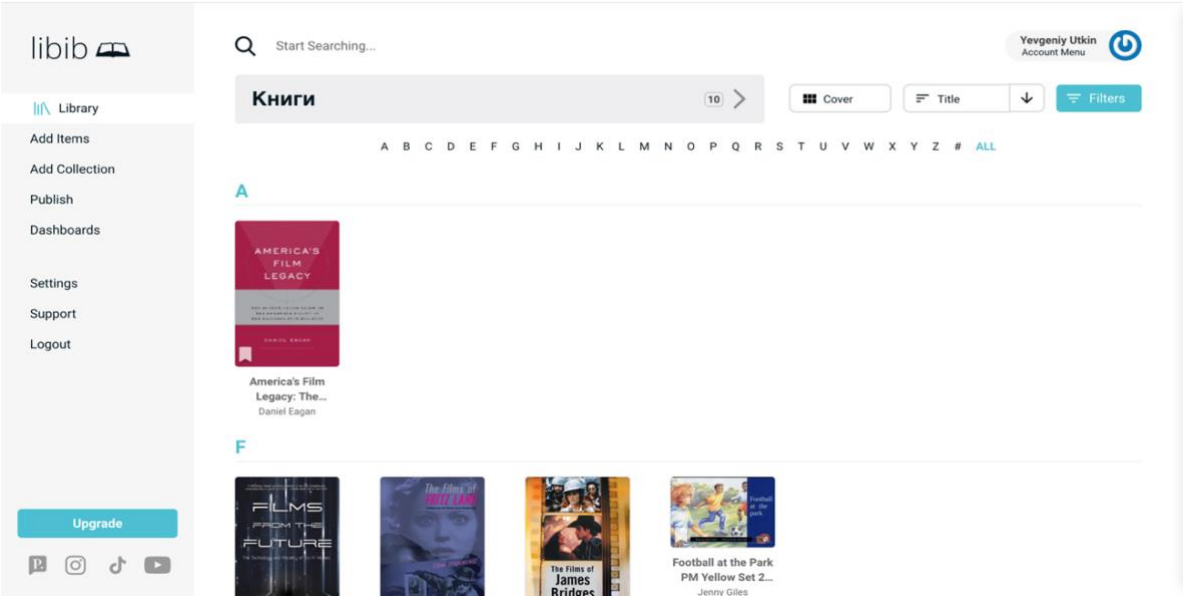

Рисунок 1.2 – Інтерфейс головної вебсторінки Libib

1.2.3 Book Catalogue

Book Catalogue – це також один із застосунків для каталогізації книг. Цей застосунок надає розширені можливості для користувачів, що дозволяє їм зручно відстежувати, додавати та керувати своїми книгами, а також синхронізувати дані з такими популярними сервісами, як Google Books. Ось огляд переваг та недоліків Book Catalogue [11]:

Переваги:

– додавання книг за ISBN: Book Catalogue забезпечує інтуїтивні інструменти для управління персональною бібліотекою, спрощуючи додавання книг через сканування штрих-коду або введення ISBN;

– відстеження прочитаних книг: застосунок також важливий для відстеження вашого читацького досвіду, дозволяючи маркувати прочитані книги та вести статистику прочитаного;

– відстеження позичених книг: функція відстеження позичених книг дозволяє контролювати, кому і коли ви позичили книгу, і нагадувати про її повернення;

– категоризація та сортування: Book Catalogue пропонує різноманітні опції для категоризації та сортування вашої колекції, включаючи створення власних категорій, додавання тегів та застосування фільтрів для оптимізації пошуку книг.

Недоліки:

– обмежений доступ: доступність Book Catalogue лише на платформі Android може становити перепону для користувачів інших систем, таких як iOS;

– обмеження соціальної взаємодії: Book Catalogue не надає розширених можливостей для обміну рекомендаціями та спілкування між користувачами, що є відмінністю від деяких інших застосунків;

– UI та дизайн: інтерфейс користувача в Book Catalogue може бути недостатньо привабливим та зручним для деяких користувачів, що може знизити загальну привабливість додатку.

Незважаючи на певні обмеження, Book Catalogue залишається ефективним інструментом для керування книжковими колекціями і моніторингу читацької активності. На рисунку 1.3 показано застосунок Book Catalogue.

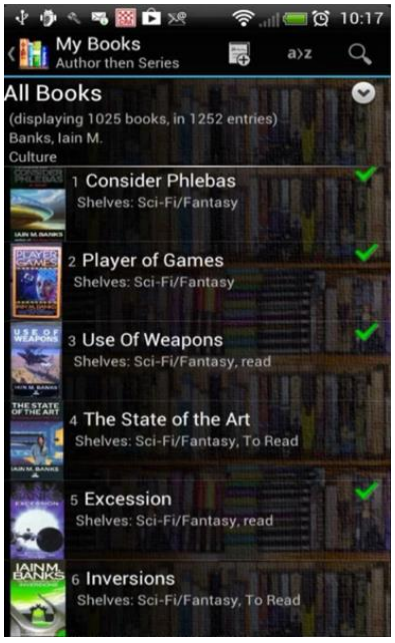

Рисунок 1.3 – Інтерфейс головної вебсторінки Book Catalogue

1.3 Огляд інструментарію для створення застосунку

Процес створення застосунку досить складний. На першому етапі розробки проекту розглядається створенням технічної документації, вибором стеку технологій для використання, а також укладанням переліку всіх технічних компонентів, які підлягають розробці. Наступним кроком йде створення дизайну. Дизайн є візитною карткою будь-якого вебзастосунку. Важливо, щоб інтерфейс був не тільки естетично привабливим, але й інтуїтивно зрозумілим та зручним у використанні. Найголовніший і найскладніший етап створення будь-якого застосунку – програмування. Розробники мають враховувати усі нюанси, щоб люди могли легко й без проблем користуватись додатком [12].

Найпопулярніша мова програмування для створення інтерактивних вебзастосунків беззаперечно JavaScript. Він вбудований у всі сучасні браузери, код може працювати на різних платформах та операційних системах. Далі ми перейдемо до вибору основних бібліотек та фреймворків.

Почнемо з бібліотеки React, який необхідний для створення фронтенду проекту. React використовується для створення елементів інтерфейсу користувача.

Для зберігання даних використовується нереляційна база даних MongoDB, яка забезпечує гнучке та ефективне управління великими обсягами даних. Водночас, для розробки серверної частини застосунку застосовуються Node.js та фреймворк Express, що дозволяють створювати API з використанням JavaScript.

Для стилізації компонентів використовується бібліотека Styled-Components, яка дозволяє дозволяє писати CSS код безпосередньо в JavaScript-файлах.

Для ефективного управління станом застосунку використовується Redux-Toolkit. Ця бібліотека надає набір інструментів, які спрощують роботу з Redux, дозволяючи розробникам легко налаштовувати магазин даних,

обробляти асинхронні запити та управляти складними станами без зайвого коду.

Також для покращення написання коду використовується Typescript. Він пропонує строгу типізацію поверх JavaScript, поліпшуючи читання коду, зменшуючи вірогідність помилок під час розробки і підтримки проектів.

Для створення та валідації форм використовується бібліотека react-hook-form та спеціальна бібліотека для валідації Yup.

1.3.1 React

React – це декларативна, ефективна і гнучка JavaScript бібліотека. Її основна задача – створення користувацьких інтерфейсів, особливо для односторінкових застосунків, де потрібна швидка взаємодія без перезавантаження сторінки.

React використовує компонентний підхід до розробки. Компоненти – це відредаговані частини інтерфейсу, які можуть бути легко повторно використані, тестовані та обслуговувані. Кожен компонент має свій власний стан, яким він може управляти, і може також приймати дані через props, що робить компоненти гнучкими.

React використовує концепцію віртуального DOM для підвищення продуктивності. Віртуальний DOM – це легка копія реального DOM. React виконує всі маніпуляції з компонентами у віртуальному DOM, після чого розраховує найоптимальніший спосіб застосування змін у реальному DOM, мінімізуючи кількість дорогих DOM-операцій.

Незважаючи на те, що React має багато переваг, у нього також є деякі недоліки:

– висока складність: для новачків у програмуванні або для тих, хто не знайомий з сучасними JavaScript-функціями, React може здатися складним

через JSX і необхідність володіння різними концепціями, такими як компоненти, стани, пропси, контекст, хуки тощо;

– тільки бібліотека інтерфейсу: на відміну від Angular, React – це лише бібліотека для побудови інтерфейсів користувача. Для деяких аспектів розробки, таких як маршрутизація або управління станом, потрібно додатково використовувати інші бібліотеки;

– часті оновлення: швидке оновлення React може бути і плюсом, і мінусом. З одного боку, це дає можливість постійно вдосконалювати функціонал. З іншого боку, це може створювати проблеми для розробників, які мають адаптувати свій код до нововведень;

– SEO оптимізація: додатки на React, які виконуються на клієнті (client-side), можуть мати проблеми з індексацією пошуковими системами, хоча це може бути вирішено за допомогою серверного рендерингу (SSR) та таких інструментів, як Next.js.

Хоча ці обмеження можуть сповільнити розробку проєкту, в більшості випадків їх можна вирішити або мінімізувати, застосовуючи відповідні методики, інструменти та практики.

Наразі React залишається однією з найпопулярніших технологій для створення застосунків, а його використання в комбінації з іншими інструментами часто призводить до відмінних результатів [13].

1.3.2 Rest API

REST API (Representational State Transfer Application Programming Interface) – це архітектурний стиль вебсервісів, який дозволяє компонентам мережі (наприклад, комп’ютерам) спілкуватися через HTTP. Використання REST API стало дуже популярним у розробці вебзастосунків та мобільних застосунків через його простоту, масштабованість та зручність інтеграції. Ось декілька ключових аспектів, які важливо розуміти про REST API:

– стандартизований інтерфейс: REST використовує стандартні HTTP методи такі як GET, POST, PUT, DELETE та інші для взаємодії між клієнтом та сервером. Це спрощує реалізацію і дозволяє використовувати стандартні бібліотеки та інструменти доступу до API;

– підтримка кешування: дозволяє підвищити продуктивність застосунків та зменшити навантаження на сервери;

– незалежність клієнта і сервера: незалежність клієнтської та серверної частин дозволяє їм розвиватися та підтримуватися окремо. Це дозволяє оновлювати інтерфейс користувача без втручання у серверну частину, і навпаки, спрощує процеси обслуговування та оновлення застосунків;

– незалежність від платформи та мов програмування: REST API не залежить від платформи або мови, тому його можна використовувати з будь-якою мовою програмування, яка підтримує HTTP;

– безпека: REST API може використовувати захист на рівні протоколів безпеки, таких як SSL/TLS, що допомагає гарантувати конфіденційність даних, запобігаючи неавторизованому доступу.

Отже, REST API це не просто набір технічних правил, це стандарт, який дає нам змогу будувати застосунки та вебсервіси, які можуть взаємодіяти один з одним за допомогою HTTP-протоколу [14].

1.3.3 Styled-Components

Styled Components – це бібліотека для React, яка дозволяє стилізувати елементи додатку за допомогою надбудови CSS в JavaScript. Це дає можливість писати традиційні CSS-правила безпосередньо у компонентах JavaScript, підтримуючи при цьому повні можливості CSS та уникаючи конфліктів класів, завдяки використанню унікальних ідентифікаторів класів. Ось основні цілі та застосування Styled-Components:

– динамічне стилізування: з Styled-Components, розробники можуть легко маніпулювати стилями заснованими на пропсах компонентів. Це робить можливим зміну стилів на основі стану чи пропсів, що додає гнучкість і динаміку в поведінку компонентів;

– підтримка тем: легко імплементувати теми у застосунку з Styled-Components завдяки можливості динамічно змінювати стилі. Розробники можуть використовувати контексти, щоб визначати та змінювати теми в усьому застосунку;

– перевикористання стилів: Styled-Components дає змогу створювати стилі, які можна легко перевикористовувати у різних компонентах, сприяючи уніфікації візуального дизайну та зменшенню дублювання коду;

– легкість тестування: компоненти, стилізовані за допомогою Styled-Components, легше тестувати на відокремленості, оскільки вони не залежать від зовнішніх стилів.

Використання Styled-Components дозволяє розробникам підтримувати стилі їхніх застосунків організовано і масштабовано, роблячи код більш чистим і легким для розуміння [15].

1.4 Постановка задачі

Таким чином, створення платформи «Персональна бібліотека» є актуальною задачею для полегшення каталогізації та ведення обліку прочитаних книг.

Об’єктом роботи є каталогізація та управління прочитаними книгами.

Метою даної кваліфікаційної роботи є розробка застосунку, що призначений для легкої та ефективної каталогізації прочитаних книг та управління особистою бібліотекою користувача. Також застосунок дозволить користувачам планувати своє майбутнє читання, проводити тренування

читання, відстеження прогресу по дням, книжкам, сторінкам та вести записи про книги, які вони бажають прочитати.

Для досягнення мети необхідно вирішити такі завдання:

– розробити користувацький інтерфейс для взаємодії з застосунком, з можливістю створення профілю користувача а також додавання та організації книжкових записів;

– розробити схему бази даних для збереження детальної інформації про книги користувачів, їх читацькі звички та відгуки;

– реалізувати функціональність тренування читання, відстеження прогресу по дням, книжкам, сторінкам;

– розробити механізм написання відгуків та виставлення рейтингу книг для користувачів.

2 ПРОЕКТУВАННЯ СИСТЕМИ, КОМПОНЕНТІВ ТА БАЗИ ДАНИХ

2.1 Проєктування системи

Проєктування системи – це складний та багатоетапний процес, який включає розробку архітектури, компонентів, інтерфейсів та інших характеристик системи. Основна мета цього процесу полягає в забезпеченні того, щоб система відповідала специфікованим вимогам та очікуванням замовника чи користувача. Ось декілька ключових процесів, які входять у проєктування системи:

– визначення вимог: це перший та один з найважливіших етапів проєктування системи. Під час цього етапу інженери збирають і аналізують вимоги від замовника та майбутніх користувачів системи. Вимоги можуть бути функціональні (що система має робити) та нефункціональні (такі як безпека, продуктивність, доступність);

– проєктування архітектури: на цьому етапі розробники створюють високорівневу структуру системи, визначаючи основні компоненти та їх взаємодії. Архітектура має забезпечити ефективне виконання функцій системи, її масштабування, та враховувати можливості для подальшого розвитку;

– проєктування компонентів: далі відбувається деталізація кожного компонента системи. Розробники специфікують внутрішню логіку, інтерфейси для взаємодії з іншими компонентами та зовнішніми системами. Кожен компонент повинен бути оптимізований для ефективного виконання своїх завдань;

– проєктування бази даних: полягає у створенні її структури та організації, включно з визначенням таблиць, встановленням зв’язків між ними та визначенням правил, які забезпечують цілісність даних. Цей процес є ключовим для ефективного зберігання і управління інформацією в системі;

– розробка інтерфейсу користувача: проєктування інтерфейсу, через який користувач взаємодіятиме з системою. Це включає дизайн сторінок, елементів керування та інші аспекти, які впливають на користувацький досвід;

– тестування: важливим кроком є комплексне тестування системи на предмет відповідності специфікаціям та виявлення помилок. Тестування може включати юніт-тести, інтеграційні тести, системні тести, та приймальні тести замовником.

Крім того, інтеграція та впровадження системи, а також її подальше обслуговування та підтримка, є критично важливими процесами, що забезпечують її ефективність та стабільність після запуску. Під час інтеграції, всі компоненти системи об’єднуються у єдине ціле, з метою гарантування їх взаємодії відповідно до задуму. Після цього система впроваджується в роботу, де вона потребує регулярного оновлення, технічного супроводу для адаптації до змін та вимогам користувачів.

Кожен із цих процесів вимагає тісної співпраці між розробниками, проектними менеджерами, аналітиками, тестувальниками та замовником, щоб забезпечити успішне завершення проєкту та досягнення [16] запланованих цілей.

2.2 Визначення вимог

Визначення вимог є критично важливим етапом у процесі розробки будь якого програмного забезпечення або інформаційної системи. Цей процес дозволяє зрозуміти реальні потреби та очікування майбутніх користувачів системи. На основі цього розуміння можна визначити як функціональні, так і нефункціональні вимоги, які система має задовольняти.

Таким чином, перш ніж приступати до аналізу вимог для зазначеної системи «Персональна бібліотека» для каталогізації і управління прочитаними книгами, необхідно ретельно вивчити потреби цільової аудиторії

користувачів, їх очікування та бажані функціональні можливості майбутнього застосунку. Лише врахувавши ці аспекти, можна буде сформулювати вичерпний набір функціональних та нефункціональних вимог до системи.

Функціональні вимоги:

– реєстрація та авторизація користувачів: система повинна мати можливість реєстрації нових користувачів, які будуть використовувати її для доступу до персональної бібліотеки;

– додавання книг до бібліотеки: система повинна дозволяти користувачам додавати нові книги до своєї персональної бібліотеки, вказуючи деталі книги, такі як назва, автор, рік видання та кількість сторінок;

– розділення книг на категорії: система повинна надавати користувачам інтерфейс розділення книг на категорії («Маю намір прочитати», «Читаю», «Прочитано»);

– відстеження прогресу читання: система повинна мати індикатор прогресу читання книги, який користувачі зможуть оновлювати вручну, а також встановлювати цілі щодо кількості прочитаних сторінок або книг за певний період часу;

– резюме та оцінки: система повинна надавати користувачам можливість написання резюме на книги після їх прочитання, а також оцінювати книги за шкалою (від 1 до 5 зірок).

Нефункціональні вимоги:

– продуктивність: система повинна забезпечувати високу продуктивність, включаючи швидке завантаження сторінок, відгук інтерфейсу та обробку великих обсягів даних без сповільнень;

– безпека: система повинна забезпечувати безпеку та конфіденційність даних користувачів, шифруючи їх під час передачі та зберігання, а також використовуючи належні методи аутентифікації та авторизації;

– адаптивність: система повинна мати адаптивний дизайн та бути повністю функціональною на різних пристроях і екранах, включаючи

смартфони, планшети та комп’ютери, забезпечуючи оптимальний користувацький досвід незалежно від розміру екрану та способу взаємодії;

– розширюваність: система повинна бути розроблена з урахуванням можливості розширення функціональності в майбутньому за допомогою плагінів, модулів або інтеграцій з іншими системами;

– доступність: система повинна дотримуватися вебстандартів та рекомендацій з доступності, забезпечуючи можливість використання людьми з обмеженими можливостями, включаючи підтримку технологій читання з екрану та альтернативних пристроїв введення.

Функціональні та нефункціональні вимоги є дуже важливим етапом у процесі розробки застосунку «Персональна бібліотека» для каталогізації і управління прочитаними книгами. Функціональні вимоги описують очікувану поведінку системи та її конкретні можливості і функції, необхідні користувачам. Тоді як нефункціональні вимоги визначають критерії безпеки, продуктивності, адаптивності та надійності, яким система має відповідати для належного виконання своїх функцій.

Активне залучення користувачів для глибокого аналізу їх потреб, а також ретельне оформлення функціональних і нефункціональних вимог є ключовими фактором для створення вебзастосунку, яке не тільки задовольняє очікування користувачів, але й забезпечує якісний [17, 18] користувацький досвід.

2.3 Опис сторінок та компонентів системи

Сторінка – це основний будівельний блок вебзастосунку, який служить окремим екраном або інтерфейсом для взаємодії з користувачем. Вона містить визначену частину контенту чи функціональності і має свою унікальну URL-адресу, що дозволяє користувачам прямо переходити на неї через браузер.

Сторінка вебзастосунку не тільки служить як окремий інтерфейс для взаємодії з користувачем, але й є центральним елементом у структурі будь-якого сайту. Кожна сторінка може включати різноманітний контент: текст, зображення, відео, інтерактивні елементи, такі як форми для введення даних та кнопки. Завдяки такій структурованості, сторінки можуть відігравати різні ролі в залежності від їхнього призначення, від простого представлення інформації до складних інтерактивних операцій.

Кожна сторінка вебзастосунку може бути адаптована під різні типи пристроїв, такі як комп’ютери, планшети та смартфони, що забезпечує користувачам однаково якісний досвід взаємодії незалежно від того, який пристрій вони використовують. Ця відповідність адаптивності заснована на принципах адаптивного вебдизайну, який дозволяє сторінкам динамічно реагувати на зміни розмірів екрану [19].

Враховуючи визначені функціональні та нефункціональні вимоги до проєкту, можливе створення системи, що включатиме такі ключові компоненти: «Реєстрація», «Логін», «Персональна бібліотека» та «Тренування».

На сторінці «Реєстрації» розташовані компоненти, які забезпечують користувачам можливість створення нового облікового запису. Центральним елементом є форма реєстрації, яка містить поля для введення імені, електронної адреси та пароля, а також поля для підтвердження пароля. Кожне поле супроводжується підказками та валідацією даних, щоб забезпечити введення коректної інформації. Після успішної реєстрації користувач перенаправляється на сторінку «Логін». Окрім форми, на сторінці присутній інформаційний розділ, що надає додаткові відомості про переваги реєстрації в системі. Тут користувачі можуть дізнатись, що система допоможе їм формулювати та досягати особисті цілі, ефективно розподіляти час та відстежувати успіхи, покращувати професійні навички, та ставати більш цікавими співрозмовниками.

На сторінці «Логін» користувачі мають можливість входу до свого профілю через спеціалізовану форму, яка є основним компонентом цієї сторінки. Форма логіну включає поля для введення електронної адреси та пароля, кожне з яких обладнане системою валідації для перевірки правильності введених даних. Введення даних відбувається через інтерфейс, який вказує на обов'язковість цих полів, що позначені символом зірочки. У разі успішного входу, користувач перенаправляється на сторінку «Персональна бібліотека». Додатково, на сторінці присутній інформаційний блок у вигляді картки з цитатою, яка має на меті надихнути користувачів. Це не тільки додає естетичний елемент до сторінки, але й підкреслює культурне значення читання [20].

На сторінці «Персональна бібліотека» користувачі мають доступ до різноманітних функцій та компонентів, які дозволяють ефективно управляти їх колекцією книг. Далі детальніше розглядається основні функції та можливості цього інтерфейсу:

– повідомлення та навчання: для нових користувачів передбачено вікно з навчальними матеріалами, яке пояснює, як користуватися бібліотекою. Це вікно може з'являтися автоматично при першому візиті та допомагає користувачам швидше адаптуватися до функціоналу сайту;

– секція додавання книг: якщо користувач відвідує сторінку з мобільного пристрою, він бачить кнопку переходу до форми додавання книг. У протилежному випадку, на планшетах та комп'ютерах відображається повноцінна секція для додавання книг, що включає форму з полями для введення назви книги, автора, року видання та загальної кількості сторінок;

– списки книг: книги у бібліотеці розподілені за категоріями, такими як «Прочитано», «Читаю» та «Маю намір прочитати». Кожна категорія містить відповідний список книг, що відображається з відповідним заголовком. У разі, якщо користувач ще не додав жодної книги, показується повідомлення про відсутність книг;

– взаємодія з книгами: користувачі можуть видаляти книги зі списку «Маю намір прочитати»;

– форма оцінки книги: ця форма дозволяє користувачам оцінювати книги (за шкалою від 1 до 5 зірок) і надавати текстовий відгук. Форма з’являється у модальному вікні, коли користувач натискає кнопку для оцінювання книги, яка вже була прочитана.

На сторінці «Тренування» користувач має можливість переглянути та управляти своїм тренуванням, що базується на їх планах читання. Ця сторінка динамічно адаптується залежно від поточного статусу користувача, перемикаючись між секціями планування та активного тренування.

Секція планування відображає:

– форму для користувача яка дозволяє вибирати дати початку і завершення тренування, які він хоче витратити на тренування;

– форму яка забезпечує користувачам інтерфейс для додавання книг до їхнього читацького плану. Надає можливість вибрати книгу з тих, що ще не прочитали;

– панель з метриками, яка відображає кількість книг які користувач запланував прочитати на тренуванні та кількість днів які він планує витратити на читання;

– кнопку початку тренування;

– список книг, які плануються до читання, з можливістю їх видалення.

Секція активного тренування відображає:

– відображає панель з таймером, який відлічує час до кінця року та до завершення тренування;

– кнопку з можливістю скасування тренування;

– список книг, які наразі використовуються в поточному тренуванні, з можливістю перегляду детальної інформації по книзі;

– панель з метриками, яка відображає кількість книг, загальну кількість днів для читання, та скільки книг залишилося прочитати;

– графік статистики читання користувача, розбитої по днях. Він допомагає користувачам наочно побачити свій прогрес, відстежуючи кількість прочитаних сторінок за кожен день тренування;

– форму для введення кількості прочитаних сторінок з обраної книги, а також список всіх внесених статистичних даних, який відображає дату, час, і кількість прочитаних сторінок за запис. При відсутності даних показується інформативне повідомлення.

Кожна окрема сторінка застосунку «Персональна бібліотека» обладнана спеціальними можливостями, які надають користувачам засоби для виконання конкретних завдань та доступу до потрібної інформації. В сукупності, ці сторінки створюють систему, яка допомагає у каталогізації, управлінні колекцією книг, реєстрації користувачів та аутентифікації [21].

2.4 Огляд та проєктування бази даних

Бази даних – це організовані колекції даних, які дозволяють зберігати, керувати та отримувати інформацію. В сучасних додатках бази даних відіграють ключову роль, дозволяючи обробляти великі об’єми даних та надавати швидкий доступ до них за допомогою запитів. Існує два основних типи баз даних: реляційні (SQL) та нереляційні (NoSQL).

Реляційна база даних – це база, де інформація організована у формі таблиць, з чітко визначеною структурою та взаємозв’язками між ними. У таких таблицях дані розміщені в рядках (кожен рядок являє собою окремий запис) та стовпцях (кожен стовпець представляє певне поле з визначеним типом даних). Кожна комірка таблиці містить дані, які відповідають заздалегідь заданому формату.

Нереляційна база даних – це тип бази даних, що зберігає інформацію без строго визначених структур і зв’язків між елементами. Замість традиційних таблиць, NoSQL бази містять різноманітні типи даних, такі як документи,

відео, зображення, а також дані з соціальних мереж. Вони не підтримують стандартні SQL-запити, що використовуються в реляційних базах даних [22].

Для розробки бази даних застосунку «Персональна бібліотека» для каталогізації і управління прочитаними книгами було обрано MongoDB – систему керування нереляційними базами даних, що дозволяє зберігати всю необхідну інформацію для забезпечення ефективного функціонування застосунку.

MongoDB – це документо-орієнтована нереляційна база даних, яка зберігає дані у формі документів у стилі JSON. Це робить MongoDB ідеальною для зберігання складних даних, динамічних схем, і для застосунків, де потрібна висока доступність і горизонтальне масштабування. MongoDB використовується для розробки застосунків, де потрібне швидке збереження, читання та обробка великих обсягів неструктурованих даних. Вона добре підходить для вебзастосунків, мобільних застосунків, а також для великих баз даних, де схеми даних можуть змінюватися. Нижче представлено деякі важливі особливості, які виділяють MongoDB:

– гнучкість: документи в MongoDB можуть мати різну структуру, що означає, що можна зберігати різні типи даних в одній і тій же базі;

– висока продуктивність: запити в MongoDB часто працюють швидше, ніж в традиційних реляційних базах даних, завдяки оптимізованій роботі з документами та можливості виконувати операції на рівні документа;

– легкість використання: JSON-подібний формат даних робить MongoDB інтуїтивно зрозумілою та легкою для розробників, що знайомі з JavaScript та іншими сучасними технологіями;

– сильна підтримка спільноти та розвиток: як відкрита база даних, MongoDB має активну спільноту розробників і постійно оновлюється з новими функціями та покращеннями.

Після ознайомлення з MongoDB, перейдемо до схеми бази даних даного вебзастосунку [23]. Отже, у проєкті використовується база даних MongoDB для зберігання і управління інформацією про книги та користувачів. Загалом

маємо п’ять колекції (таблиць) – «users», «books», «plans», «statistics», «histories». Кожна з них представляє різні типи даних і використовується на відповідних сторінках застосунку.

Колекція «user» відповідає за збереження інформації про користувачів системи. Вона містить дані про кожного користувача, такі як ім’я (name), електронна адреса (email), пароль (password), токен доступу (accessToken) і токен оновлення (refreshToken). Ім’я та електронна пошта перевіряються за встановленими шаблонами і вимогами до довжини, а електронна адреса є унікальною для кожного користувача, що забезпечує їх ідентифікацію. Додатково, кожен користувач має поля для зберігання токенів, що використовуються для аутентифікації. Ця інформація використовується на сторінках реєстрації, логіна, а також для підтримки активної сесії на сайті. Детальну схему даної колекції представлено у таблиці 2.1.

Таблиця 2.1 – Схема колекції Користувачі (users)

Зміст поля	**Назва поля**	**Тип даних**	**Нотатки**
Ідентифікатор користувача	_id	ObjectId	
Ім’я	name	String	
E-mail	email	String	
Пароль	password	String	Зберігається у зашифрованому вигляді
Токен доступу	accessToken	String	
Токен оновлення	refreshToken	String	

Колекція «book» відповідає за збереження інформації про книги, які користувачі використовують у своїх особистих бібліотеках. Вона містить дані про кожну окрему книгу, такі як назва книги (title), автор (author), рік публікації (publishYear), загальна кількість сторінок (pagesTotal), кількість прочитаних сторінок (pagesFinished), оцінка книги (rating), резюме (feedback)

та ідентифікатор користувача (owner). Загальна кількість сторінок та кількість прочитаних сторінок дозволяє користувачам відстежувати прогрес читання. Оцінки та відгуки надають можливість висловлювати думку про прочитане. Ідентифікатор власника зв’язує книгу з конкретним користувачем. Ця інформація використовується для каталогізації книг, пошуку в бібліотеці, ведення статистики читання. Детальну схему даної колекції показано у таблиці 2.2.

Таблиця 2.2 – Схема колекції Книги (books)

Зміст поля	**Назва поля**	**Тип даних**	**Нотатки**
Ідентифікатор книги	_id	ObjectId	
Назва книги	title	String	
Автор	author	String	
Рік публікації	publishYear	Number	
Загальна кількість сторінок	pagesTotal	Number	
Кількість прочитаних сторінок	pagesFinished	Number	
Оцінка книги	rating	Number	Може мати оцінку від 1 до 5 зірок
Резюме	feedback	String	
Ідентифікатор користувача	owner	ObjectId	

Колекція «plan» відповідає за збереження інформації про плани читання користувачів. Вона містить дані про кожен окремий план, такі як дата початку (startDate), дата закінчення (endDate), список книг (books), масив

ідентифікаторів статистики по кожному дню читання (statistics), статус плану (status) та ідентифікатор користувача (owner). Список книг дозволяє користувачам відстежувати, які книги вони планують читати протягом визначеного періоду. Статус плану показує його поточний стан, наприклад, чи план активний чи завершений. Ідентифікатор власника зв’язує план з конкретним користувачем. А масив ідентифікаторів статистики зв’язує поточний план з днями коли користувач читає книги. Ця інформація використовується для управління планами читання, відстеження прогресу. Детальну схему даної колекції показано у таблиці 2.3.

Таблиця 2.3 – Схема колекції Планів (plans)

Зміст поля	**Назва поля**	**Тип даних**	**Нотатки**
Ідентифікатор плану	_id	ObjectId	
Дата початку	startDate	String	
Дата закінчення	endDate	String	
Список книг	books	Array	Містить в собі інформацію про книги
Ідентифікатори статистики	statistics	Array	Містить в собі інформацію про статистику по кожному дню
Статус плану	status	String	Може мати три стани – idle, active, finished, timeover
Ідентифікатор користувача	owner	ObjectId	

Колекція «statistic» відповідає за збереження даних про статистику читання користувачів. Вона містить дані про кожен окремий день читання, такі як дата (date), кількість прочитаних сторінок за день (pagesPerDay), інформація

про конкретні сесії читання в цей день (currentDateStats), які включають кількість прочитаних сторінок (pagesRead), час читання (time), ідентифікатор прочитаної книги (book) та позначку про завершення книги (isFinishedBook). Також колекція «statistic» містить ідентифікатор плану, з яким пов’язана статистика (plan), та ідентифікатор користувача (owner). Ця інформація використовується для оцінки прогресу за планами читання та загальної оцінки активності користувачів у системі. Детальну схему даної колекції показано у таблиці 2.4.

Таблиця 2.4 – Схема колекції Статистики (statistics)

Зміст поля	**Назва поля**	**Тип даних**	**Нотатки**
Ідентифікатор статистики	_id	ObjectId	
Дата	date	String	
Кількість прочитаних сторінок за день	pagesPerDay	Number	
Інформація про конкретні сесії читання в цей день	currentDateStats	Object	Об’єкт містить: pagesRead - Number time - String book - ObjectId isFinishedBook - Boolean
Ідентифікатор плану	plan	ObjectId	
Ідентифікатор користувача	owner	ObjectId	

Колекція «history» відповідає за збереження історії завершених чи скасованих планів читання. Вона містить дані про кожен окремий план, такі як дата початку (startDate), дата завершення (endDate), дата реального

завершення плану (completionDate), статус (status), масив ідентифікаторів статистики (statistics), та ідентифікатор користувача (owner). Ця інформація дозволяє відстежувати історію активностей користувачів, визначати частоту та ефективність виконання планів. Детальну схему даної колекції показано у таблиці 2.5.

Таблиця 2.5 – Схема колекції Історії (histories)

Зміст поля	**Назва поля**	**Тип даних**	**Нотатки**
Ідентифікатор історії	_id	ObjectId	
Дата початку	startDate	String	
Дата завершення	endDate	String	
Дата реального завершення	completionDate	String	
Статус	status	String	Може мати три стани - cancel, finished, timeover
Ідентифікатори статистики	statistics	Array	
Ідентифікатор користувача	owner	ObjectId	

Отже, було описано структуру п’яти різних колекцій: «users», «books», «plans», «statistic» і «history», кожна з яких відіграє ключову роль у функціонуванні застосунку «Персональна бібліотека» для каталогізації і управління прочитаними книгами.

Кожна колекція використовується на різних сторінках системи для забезпечення зв’язку між даними і функціоналом застосунку. Ідентифікатори користувачів (owner), ідентифікатор книг (book), ідентифікатор планів (plan), ідентифікатори статистики (statistics) в колекціях, які містять користувацькі дані (книги, плани, статистика, історія), забезпечують зв’язок між даними і

конкретними користувачами, що дозволяє ефективно організувати і відображати інформацію в контексті кожного користувача.

Ця структуризація даних сприяє не тільки ефективному збереженню інформації, але й підвищує зручність використання застосунку, оскільки кожна колекція відповідає за свою унікальну функціональну область, забезпечуючи відповідність даних потребам користувачів.

2.5 Обґрунтування вибору середовища розробки

Для реалізації застосунку «Персональна бібліотека» для каталогізації і управління прочитаними книгами було обрано Visual Studio Code (VS Code), який є одним із найпопулярніших середовищ розробки для програмістів у всьому світі, завдяки своїй гнучкості, потужності та широкому спектру функціональних можливостей, а також завдяки своєму привабливому дизайну. Інтерфейс цього інструменту має мінімалістичний стиль, який допомагає користувачам зосередитися на програмуванні без зайвих перешкод. Нижче розглядаються ключові переваги, які роблять VS Code ідеальним вибором для розробки застосунків:

– багатомовна підтримка: VS Code підтримує численні мови програмування та розмітки, включаючи HTML, CSS, JavaScript, TypeScript, Python, PHP, C++ і багато інших. Це робить його універсальним інструментом для розробників різних спеціалізацій;

– розширення та налаштування: ринок розширень Visual Studio Code надає доступ до тисяч розширень, які можуть бути легко додані через вбудовану платформу. Вони можуть допомогти покращити функціональність і адаптувати редактор до конкретних потреб проекту чи особистих переваг розробника;

– вбудована підтримка контролю версій: VS Code має вбудовану підтримку Git, що дозволяє розробникам використовувати функціонал Git та інші операції без необхідності виходити з редактора;

– кросплатформність: VS Code можна встановити на різні операційні системи, включаючи Windows, macOS та Linux, що забезпечує гнучкість для розробників, які працюють на різних платформах;

– висока продуктивність та мінімальні вимоги до системи: незважаючи на свої розширені можливості, VS Code залишається легким і швидким, що робить його ідеальним для розробників, які цінують швидкий відгук від своїх інструментів розробки;

– режим «Розділення екрану»: ця функція дозволяє відкривати кілька файлів одночасно в одному вікні редактора. Це особливо корисно при роботі з великими проектами, де необхідно одночасно переглядати кілька файлів.

Обрання Visual Studio Code як середовища розробки є обґрунтованим рішенням, що підкріплене його гнучкістю, розширюваністю та інтеграцією з ключовими інструментами та платформами, які є необхідними для ефективної розробки сучасних вебзастосунків. VS Code не тільки задовольняє технічні потреби проєкту, але й сприяє підвищенню продуктивності розробників через свої оптимізовані робочі процеси та користувацькі можливості. Це робить його ідеальним вибором для реалізації застосунку «Персональна бібліотека» для каталогізації і управління прочитаними книгами [24].

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ «ПЕРСОНАЛЬНА БІБЛІОТЕКА»

3.1 Програмна реалізація

У рамках кваліфікаційної роботи був розроблений застосунок «Персональна бібліотека» для каталогізації і управління прочитаними книгами. Для реалізації було обрано такі інструменти: JavaScript, TypeScript, Node.js, Express, React та інші.

3.1.1 Створення серверної частини

Для зберігання даних використовується нереляційна база даних MongoDB, яка забезпечує гнучке та ефективне управління великими обсягами даних. Для створення серверної частини застосунку застосовуються Node.js – це середовище виконання JavaScript на стороні сервера, яке використовується для створення масштабованих серверних застосунків. Також використовується фреймворк Express, що дозволяє створювати API з використанням JavaScript. Для Для створення API використовується технологія REST API. Це бекенд, який є прошарком між вебзастосунком та базою даних. API надає стандартний інтерфейс для доступу до ресурсів. Відправляємо HTTP-запит з клієнта на сервер, у відповідь ми отримуємо не HTML-сторінку, а дані у форматі JSON.

Підключення до бази даних є критично важливим елементом більшості вебзастосунків, які використовують серверну логіку для зберігання та обробки даних. Для підключення до бази даних використовується рядок підключення DB_HOST, який містить всі необхідні дані.

Розглянемо програмний код підключення до бази даних. Його подано у лістингу 3.1.

Лістинг 3.1 Реалізація підключення до бази даних MongoDB за допомогою бібліотеки Mongoose [25, 26]:

```
const app = require('./app');
const mongoose = require('mongoose');
const { DB_HOST, PORT = 8000 } = process.env;
mongoose
  .connect(DB_HOST)
  .then(() => {
    app.listen(PORT);
  })
  .catch(error => {
    process.exit(1);
  });
```

У створеному застосунку підключення до бази даних відбувається наступним чином:

Крок 1. Для того, щоб підключитися до бази даних використовується mongoose.connect(DB_HOST), де DB_HOST – це змінна середовища, що містить URL підключення до MongoDB.

Крок 2. Після успішного підключення, серверний застосунок починає слухати вхідні HTTP запити на вказаному порті.

Крок 3. У випадку помилки підключення застосунок завершує свою роботу.

Далі детально розглядається процес реалізації реєстрації та авторизації користувачів у системі вебзастосунку. Ці дві ключові функції є фундаментальними для забезпечення безпеки та індивідуалізації досвіду користувача, дозволяючи користувачам створювати особисті облікові записи

та безпечно входити в систему для доступу до персоналізованих ресурсів і сервісів [27].

Спочатку розглядається програмний код реєстрації користувача в системі. Його подано у лістингу 3.2.

Лістинг 3.2 Реалізація функції реєстрації користувача в системі:

```
const register = async (req, res, next) => {
    try {
        const { error } = schemas.userRegister.validate(req.body);
        const { email, password } = req.body;
        if (error) {
            throw HttpError(400, error.message);
        }
        const user = await User.findOne({ email });
        if (user) {
            throw HttpError(409, 'Email on use');
        }
        const passwordHash = await bcryptjs.hash(password, 10);
        const newUser = await User.create({
            ...req.body,
            password: passwordHash,
        });
        res.status(201).json({
            name: newUser.name,
            email: newUser.email,
        });
    } catch (error) {
        next(error);
    }
};
```

Дана функція реєстрації здійснюється за наступною схемою:

Крок 1. Проходить валідація вхідних даних – req.body за допомогою схеми Joi, які включають електронну пошту та пароль. Це забезпечує, що введені дані відповідають очікуваному формату та вимогам. Якщо валідація не пройдена, виникає помилка, зі статусом 400 і повідомленням про помилку.

Крок 2. Далі перевіряється, чи існує вже у базі даних користувач з такою самою електронною поштою. Якщо користувач з вказаною електронною поштою вже існує, виникає помилка зі статусом 409 (конфлікт) та повідомленням про те, що електронна пошта вже використовується.

Крок 3. Наступним кроком використовується бібліотека bcryptjs [28] для безпечного хешування пароля користувача перед його зберіганням у базі даних. Це запобігає збереженню пароля у відкритому вигляді.

Крок 4. Далі створюється новий запис користувача з усіма вхідними даними та хешованим паролем у базу даних.

Крок 5. Якщо усі попередні кроки пройдені успішно, клієнту надсилається відповідь із статусом 201, яка містить ім’я та електронною поштою нового користувача.

Цей підхід дозволяє ефективно реєструвати користувачів, забезпечуючи при цьому високий рівень безпеки та надійності обробки даних. Далі розглядається програмний код авторизації користувача в системі. Його подано у лістингу 3.4.

Лістинг 3.3 Створення токенів доступу та оновлення для управління сесіями вебзастосунку:

```
const jwt = require('jsonwebtoken');
const createTokens = id => {
   const payload = {
      id,
   };
```

```
    const accessToken = jwt.sign(payload,
  process.env.JWT_ACCESS_SECRET, {
        expiresIn: '1h',
    });
    const refreshToken = jwt.sign(payload,
  process.env.JWT_REFRESH_SECRET, {
        expiresIn: '23h',
    });
    return { accessToken, refreshToken };
  };
```

Лістинг 3.4 Реалізація функції авторизації користувача в системі:

```
const login = async (req, res, next) => {
    try {
        const { error } = schemas.userLogin.validate(req.body);
        const { email, password } = req.body;
        if (error) {
            throw HttpError(400, error.message);
        }
        const doc = await User.findOne({ email });
        if (!doc) {
            throw HttpError(401, 'Email or password is wrong');
        }
        const passwordCompare = await bcryptjs.compare(password,
  doc.password);
        if (!passwordCompare) {
            throw HttpError(401, 'Email or password is wrong');
        }
        const { accessToken, refreshToken } = createTokens(doc._id);
```

```
        const user = await User.findByIdAndUpdate(
          { _id: doc._id },
          { accessToken, refreshToken }
        );
        res.json({
          userData: {
            name: user.name,
            email: user.email,
          },
          accessToken,
          refreshToken,
        });
      } catch (error) {
        next(error);
      }
};
```

Дана функція авторизації здійснюється за наступною схемою:

Крок 1. Спочатку функція перевіряє вхідні дані (req.body), які містять електронну пошту та пароль користувача за допомогою схеми Joi. Якщо дані не відповідають вимогам схеми, надсилається відповідь зі статусом 400 і повідомленням про помилку.

Крок 2. Далі перевіряється, чи існує вже у базі даних користувач з такою самою електронною поштою. Якщо користувач з вказаною електронною поштою вже існує, виникає помилка зі статусом 409 (конфлікт) та повідомленням про те, що електронна пошта вже використовується.

Крок 3. Далі функція використовує бібліотеку bcryptjs для порівняння введеного користувачем пароля з захешованим паролем у базі даних. Якщо паролі не співпадають, надсилається відповідь зі статусом 401 з повідомленням про невірний логін або пароль.

Крок 4. Якщо перевірка пароля пройшла успішно, функція createTokens (продемонстрована у лістингу 3.3), викликається з ідентифікатором користувача (doc._id) та використовуючи бібліотеку jsonwebtoken [29, 30] створює пару токенів – accessToken та refreshToken.

Крок 5. Наступним кроком після створення токенів поля accessToken та refreshToken оновлюються у профілі користувача у базі даних.

Крок 6. Якщо усі попередні кроки пройдені успішно, функція відправляє відповідь у форматі JSON, яка містить основні дані користувача (ім’я та електронну пошту) та обидва токени (accessToken та refreshToken). Структура API-endpoint сервера показана на рисунку 3.1.

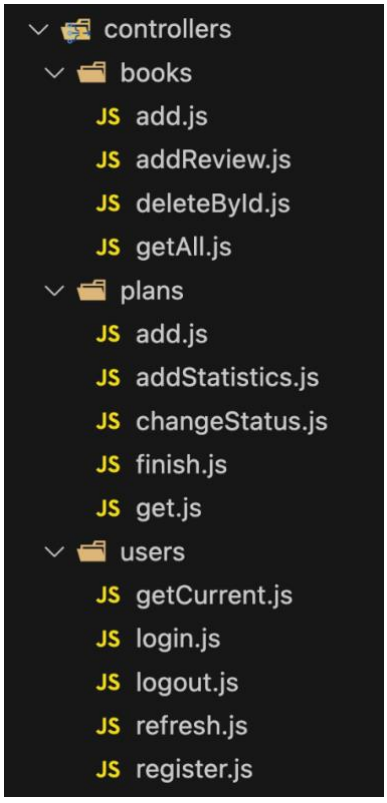

Рисунок 3.1 – Структура API-endpoint сервера

Після успішної реєстрації та входу в профіль, користувачу відкривається уся функціональність застосунку «Персональна бібліотека» для каталогізації і управління прочитаними книгами. Розглянемо усі API-endpoint серверної частини вебзастосунку. Коротко про кожен з них наведено у таблиці 3.1.

Таблиця 3.1 – Опис API-endpoint

Назва API-endpoint	**Назва файлу**	**Опис**
1	**2**	**3**
Реєстрація	users/register.js	Призначена для створення нових облікових записів користувачів у системі
Авторизація	users/login.js	Призначена для авторизації користувачів у системі
Вихід з застосунку	users/logout.js	Призначена для виходу користувачів із системи.
Отримання поточного користувача	users/getCurrent.js	Призначена для отримання та відправлення поточної інформації про користувача, який вже авторизований у системі.
Оновлення токенів	users/refresh.js	Призначена для оновлення токенів доступу та оновлення сесії.
Отримання списків книг	books/getAll.js	Призначена для отримання списків книг з бази даних, розподілених за категоріями стану читання: «Маю намір прочитати», «Зараз читаю» та «Прочитано».
Додавання книги	books/add.js	Відповідає за додавання нової книги до бази даних користувача у вебзастосунку.
Видалення книги	books/deleteById.js	Відповідає за видалення книги з бази даних вебзастосунку.

Продовження таблиці 3.1

1	2	3
Додавання відгуку	books/addReview.js	Призначена для додавання відгуку до книги в базі даних користувача.
План читання	plans/get.js	Призначена для отримання інформації про план читання користувача.
Додавання нового плану	plans/add.js	Призначена для створення нового плану читання користувача у вебзастосунку.
Завершення плану	plans/finish.js	Призначена для завершення існуючого плану читання користувача.
Додавання статистики	plans/addStatistics.js	Відповідає за додавання статистики читання користувачем книги в межах активного плану читання.
Зміна статусу	plans/changeStatus.js	Призначена для зміни статусу плану читання користувача відповідно до його актуальності відносно поточної дати.

Ці програмні інтерфейси є важливою частиною архітектури застосунку «Персональна бібліотека» для каталогізації і управління прочитаними книгами, оскільки вони активно використовуються різними компонентами системи для забезпечення її основної функціональності. Вони дозволяють різним частинам застосунку ефективно взаємодіяти між собою, передаючи

дані та команди через добре визначені інтерфейси. В результаті, використання цих API є критично важливим для розвитку та підтримки вебзастосунку [31].

3.1.2 Створення клієнтської частини

Після створення серверної частини розробляється клієнтська частина вебзастосунку, що є важливим компонентом для забезпечення інтерактивності та зручності користувача. Для створення клієнтської частини використовується бібліотека React, яка дозволяє будувати інтуїтивно зрозумілі та ефективні користувацькі інтерфейси за допомогою компонентного підходу. Структура клієнтської сторони вебзастосунку показана на рисунку 3.2.

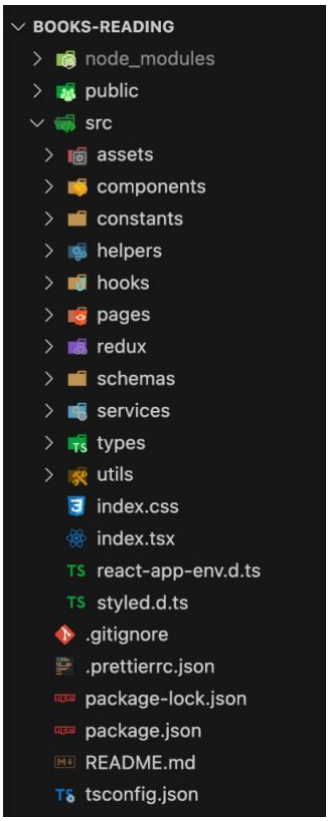

Рисунок 3.2 – Структура клієнтської сторони вебзастосунку

Компоненти сторінки – це основні будівельні блоки, які складають інтерфейс користувача в вебзастосунку. Кожен компонент відповідає за певну

частину інтерфейсу та має власну логіку і стан. Компоненти можуть бути повторно використані в різних частинах застосунку, що сприяє модульності та підтримці коду. Для зв'язку між компонентами та серверною частиною використовується API, який дозволяє різним частинам застосунку ефективно взаємодіяти між собою, передавати та отримувати дані з сервера [32].

Перейдемо до опису головних компонентів вебзастосунку. Компонент RegisterForm – це форма реєстрації користувачів у застосунку. Вона призначена для збору інформації від користувачів, які хочуть створити новий обліковий запис. Форма складається з чотирьох основних текстових елементів: ім'я, електронна адреса, пароль та підтвердження пароля, а також кнопки відправлення даних. Кожне текстове поле марковано зірочкою, що позначає його як обов'язкове для заповнення. Додатково, форма має посилання на авторизацію, якщо користувач вже має обліковий запис. При відправленні форми відбувається перенаправлення на сторінку авторизації при успішний реєстрація користувача в системі або у випадку виникнення помилки отримає відповідне сповіщення. Компонент містить стилізацію за допомогою Styled Components, який відображає форму з привабливим дизайном.

Компонент LogInForm – це форма входу користувача у застосунок. Вона призначена для авторизації користувачів шляхом введення електронної адреси та пароля. Також присутня кнопка для відправлення даних. Форма складається з двох основних текстових елементів, кожне з яких позначено зірочкою, що вказує на обов'язковість заповнення. Додатково, форма має посилання на реєстрацію, якщо користувач ще не має облікового запису. Це дозволяє легко перейти до форми реєстрації. При відправленні форми відбувається перенаправлення на сторінку персональної бібліотеки при успішний авторизації або у випадку виникнення помилки отримає відповідне сповіщення.

Компонент BookAddSection відповідає за додавання нових книг у персональну бібліотеку. В цьому компоненті є форма яка складається з поля для введення назви книги, автора, року випуску та кількості сторінок. Кожне

поле форми супроводжується текстом опису для зручності користувача. Компонент також має кнопку для відправки форми, яка стає неактивною під час процесу додавання книги. Форма використовує механізми обробки і перевірки введених даних, щоб забезпечити коректність інформації. При успішному додаванні книги, форма автоматично очищує введені дані, чекаючи наступне введення даних. У випадку помилок при введенні, відображаються повідомлення з вказівками про помилки для користувача.

Компоненти в яких присутні форми використовують React Hook Form для зберігання та обробки даних форми, включаючи управління полями за допомогою контролера. Для валідації даних використовується бібліотека yup. Щоб забезпечити валідацію, використовується yup-схема, що визначає правила для кожного поля.

Компонент BooksList призначений для відображення списку книг. Цей компонент є гнучким та налаштовуваним, оскільки він може приймати кілька пропсів, що визначають його поведінку та зовнішній вигляд. Він використовує вкладений компонент BookCard, який відображає детальну інформацію про конкретну книгу. Це універсальний компонент, який адаптовано для різних станів книги, наприклад таких як «Маю намір прочитати», «Читаю», «Прочитано». Компонент складається з елементів: назви книги, автора, року випуску, кількості сторінок, кнопки видалення книги, прапорця прочитаної книги (checkbox), а також модального вікна для оцінки книги та надання резюме.

Компонент BookReviewForm використовується для додавання відгуків та оцінок до книги. Користувачі можуть вибрати рейтинг від 1 до 5 зірок і написати текстовий відгук, який не повинен перевищувати 3000 символів. Компонент включає валідацію даних і показує помилки, якщо введені дані не відповідають вимогам. Компонент містить кнопку для збереження відгуку та оцінки.

Компонент BookSelectForm – це форма для вибору книг доступних для тренування та визначення дат для тренування. Він включає у себе календар

для вибору дат початку та завершення читання, а також випадаючий список для вибору книги. Дати можуть бути налаштовані з мінімальною датою сьогодні та максимальною датою, визначеною в системі. Форма також має кнопку для додавання вибраних книг. При виборі книги користувач може переглядати назву, автора, рік випуску та кількість сторінок відповідної книги у зручному форматі.

Компонент StatisticsChart відображає графік читання книг, де представлені дві лінії: одна для планової кількості сторінок на день і друга для фактично прочитаних сторінок. Використання даних з Redux дозволяє відслідковувати прогрес читання на основі вибраних дат. Графік адаптується під різні типи пристроїв, використовуючи змінну кількість відображуваних значень відповідно до розміру екрану: 3 для мобільних, 6 для планшетів та 7 для комп'ютерів. Опції конфігурації графіка включають налаштування для легенди, осей та інтерактивності, щоб забезпечити зручне відображення інформації. Для формування даних використовуються мемоізовані обчислення, які гарантують, що інформація буде оновлюватися тільки при зміні вхідних даних. Це забезпечує високу продуктивність компонента, оскільки уникається непотрібне перерахування даних. Компонент також включає секцію з лічильником, яка відображає поточну кількість сторінок, які необхідно читати на день.

Компонент StatisticsForm призначений для додавання статистичних даних про прочитані сторінки книги. Він включає форму з двома основними полями: вибір книги та кількість прочитаних сторінок. Для кожної книги можна вказати, скільки сторінок було прочитано, що допомагає користувачу вести облік свого прогресу в читанні. Книги для вибору фільтруються за критерієм, що кількість вже прочитаних сторінок не дорівнює загальній кількості сторінок книги. Компонент використовує випадаючий список для вибору книги, і кожен варіант у цьому списку містить назву та автора книги. Компонент також містить кнопку для відправлення форми, яка стає неактивною під час процесу додавання даних, запобігаючи подвійного

надсилання. Після успішного додавання даних, форма автоматично очищається, дозволяючи внести наступний запис.

Таким чином, компоненти сторінок створюють зручний і інтуїтивно зрозумілий інтерфейс для користувача, полегшують процес розробки та роблять код більш масштабованим і підтримуваним.

3.2 Інструкція користувача

При переході за посиланням на застосунок «Персональна бібліотека» для каталогізації і управління прочитаними книгами, користувач потрапляє на сторінку реєстрації (рис. 3.3).

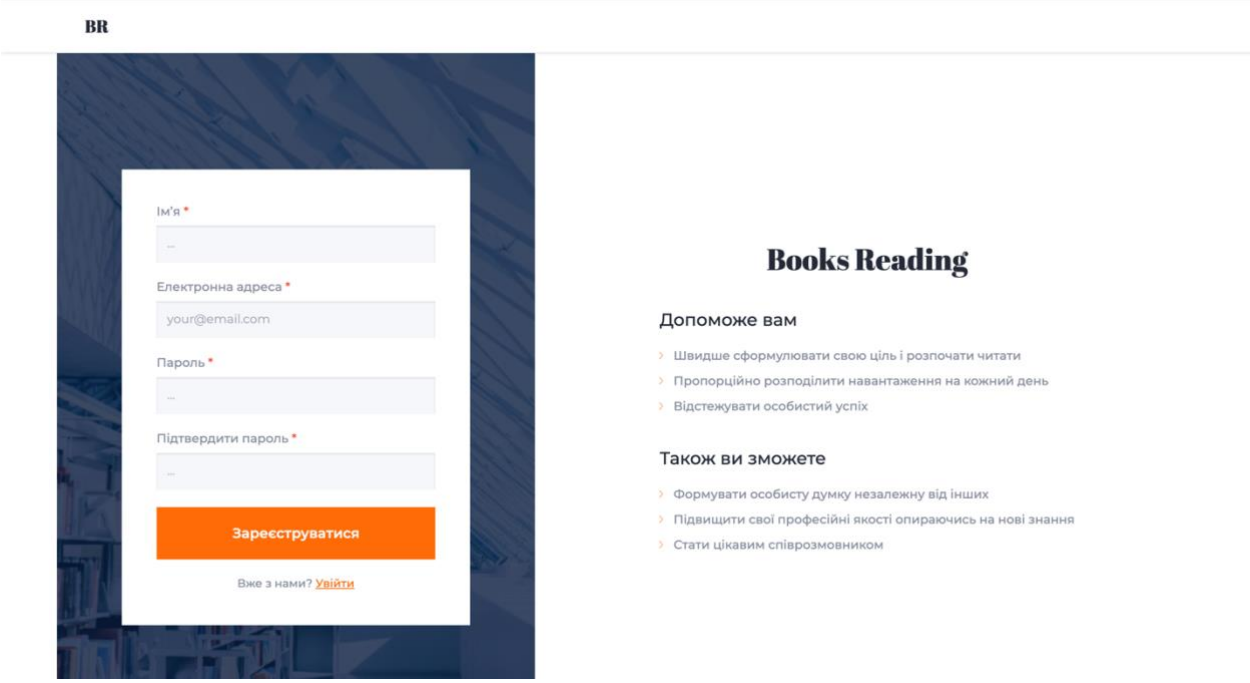

Рисунок 3.3 – Головна сторінка вебзастосунку «Персональна бібліотека» для каталогізації і управління прочитаними книгами

На сторінці реєстрації користувач може створити новий обліковий запис, заповнивши форму, яка складається з полів: ім’я, електронна адреса, пароль та підтвердження пароля. Праворуч на сторінці користувач бачить

інформаційний блок, що система допоможе їм формулювати та досягати особистих цілей, ефективно розподіляти час, відстежувати успіхи, покращувати професійні навички та ставати більш цікавими співрозмовниками. Пароль має певні обмеження щодо валідності даних. Він повинен складатися не менше, ніж з 8 символів. Також на сторінці є посилання «Увійти», яке перенаправляє користувача на сторінку входу в обліковий запис, якщо він вже зареєстрований (рис 3.4).

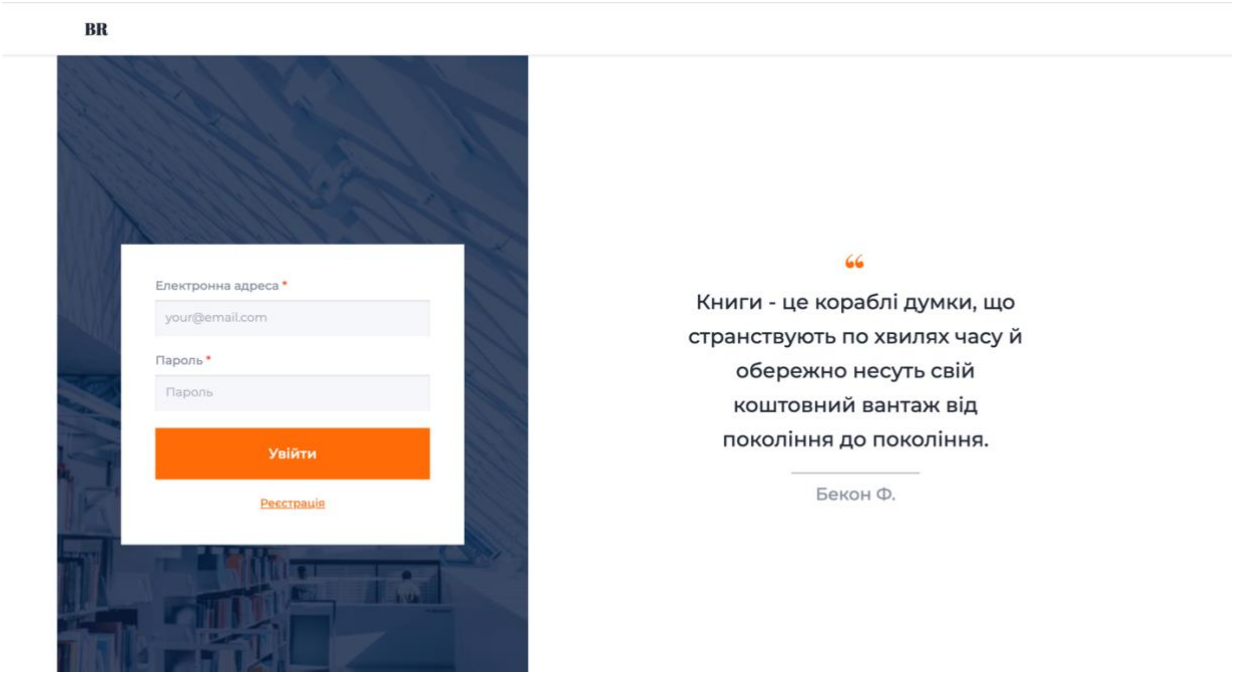

Рисунок 3.4 – Сторінка авторизації користувача

На сторінці входу користувач може увійти в свій обліковий запис, заповнивши форму, яка складається з двох полів: електронна адреса та пароль. Також на сторінці є посилання «Реєстрація», яке перенаправляє користувача на сторінку створення нового облікового запису, якщо він ще не зареєстрований. У випадку, якщо користувач має акаунт та ввів коректні персональні дані, та успішно зареєструвався, при натисканні на кнопку «Увійти» він потрапить на персональну сторінку (рис. 3.5).

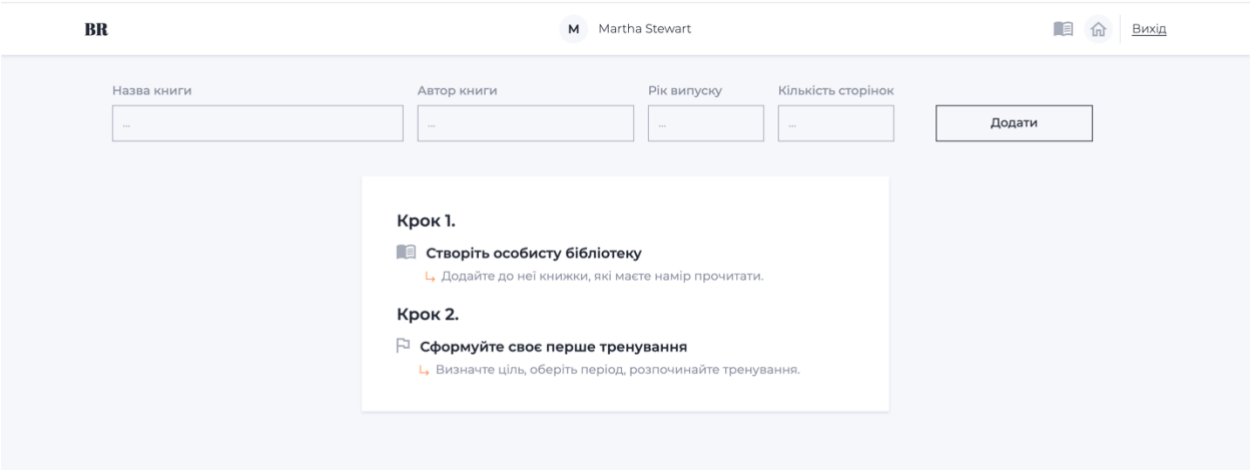

Рисунок 3.5 – Сторінка персональної бібліотеки

Вгорі сторінки знаходиться панель, на якій відображається ім’я користувача, навігаційні посилання на сторінки «Персональна бібліотека» і «Тренування читання» та кнопка «Вихід» для виходу с застосунку. Під панеллю розташована форма для додавання книг, яка включає такі поля: назва книги, автор, рік випуску та кількість сторінок. Після додавання книг, відображається список книг, які були додані до бібліотеки, на яких присутня вся інформація про кожну книгу. Крім того, є можливість видалити книгу з персональної бібліотеки (рис 3.6).

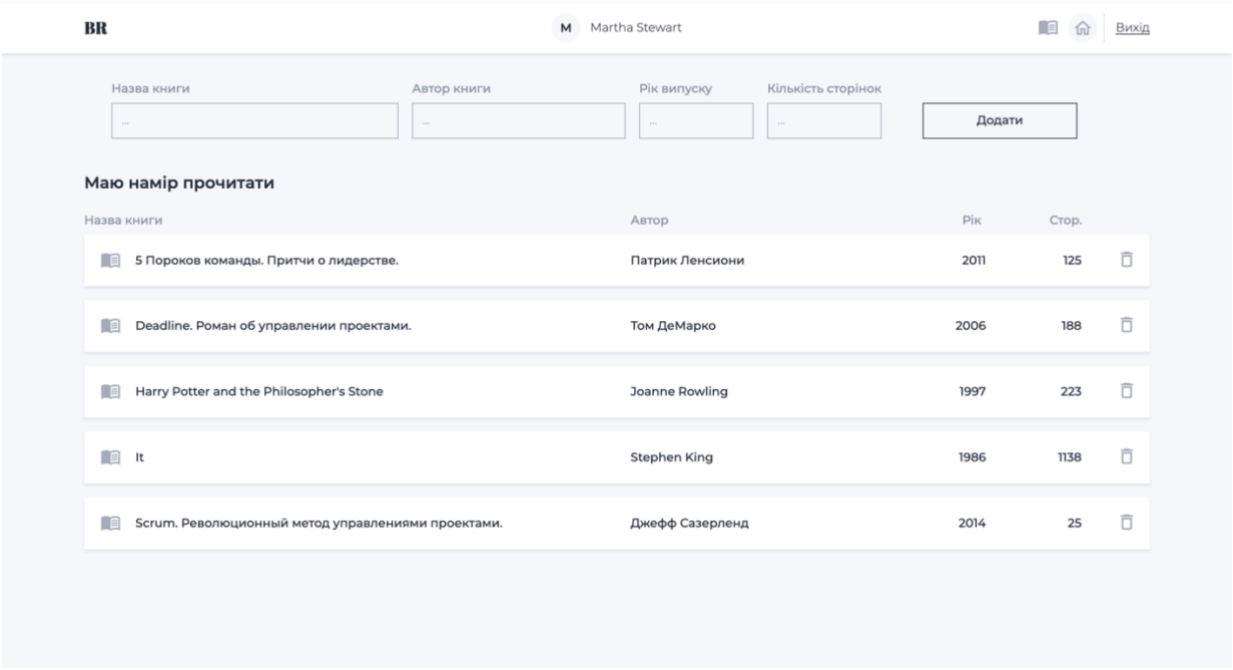

Рисунок 3.6 – Представлення доданих книг у персональній бібліотеці

Поруч із формою є кнопка «Додати», яка дозволяє додати книгу до персональної бібліотеки. Крім того, на сторінці присутній інформаційний блок з покроковими інструкціями щодо користування вебзастосунком.

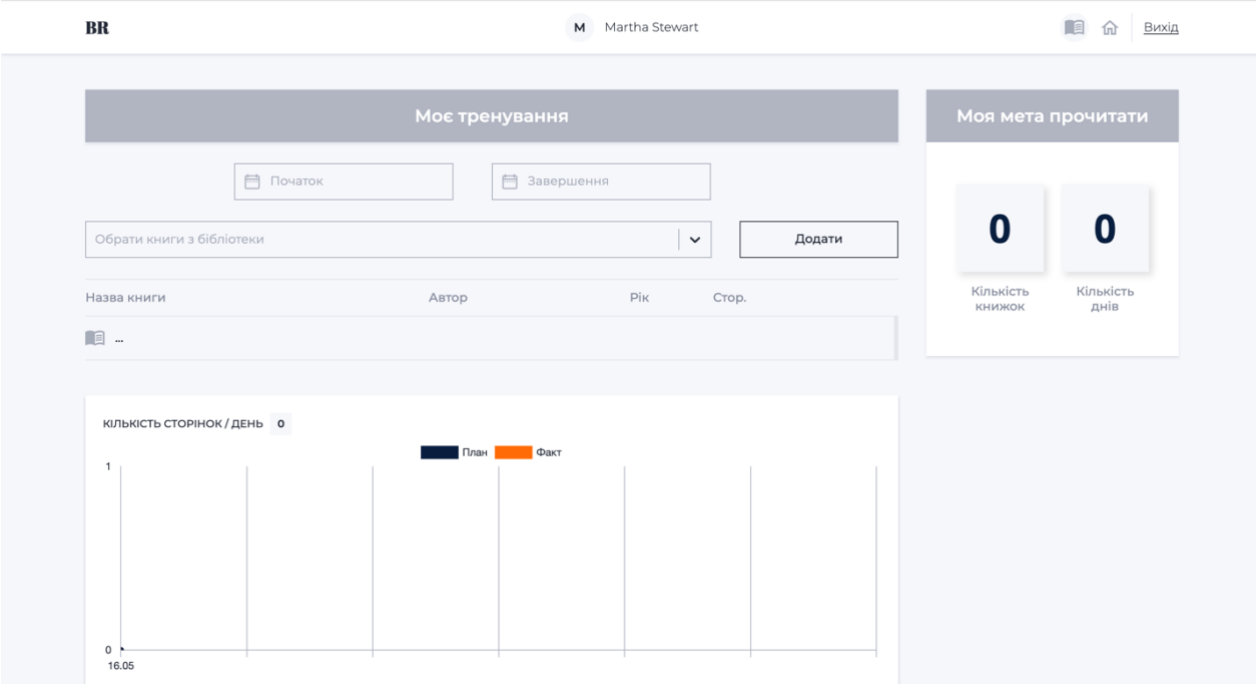

Рисунок 3.7 – Сторінка тренування читання (секція планування)

На сторінці «Тренування читання» відображається секція планування читання, де присутня форма для вибору книг доступних для тренування та визначення дат. Він включає у себе календар для вибору дат початку та завершення читання, а також список для вибору книги. Форма також має кнопку для додавання вибраних книг. При додаванні книги користувач може переглядати назву, автора, рік випуску та кількість сторінок. А також є кнопка видалення книги зі списку книг, які плануються для тренування. Праворуч розташована панель з метриками, яка відображає кількість книг, запланованих для читання, та кількість днів, які користувач планує витратити на читання. Також на панелі представлений графік читання, де в процесі тренування буде відображатися весь прогрес (рис 3.8).

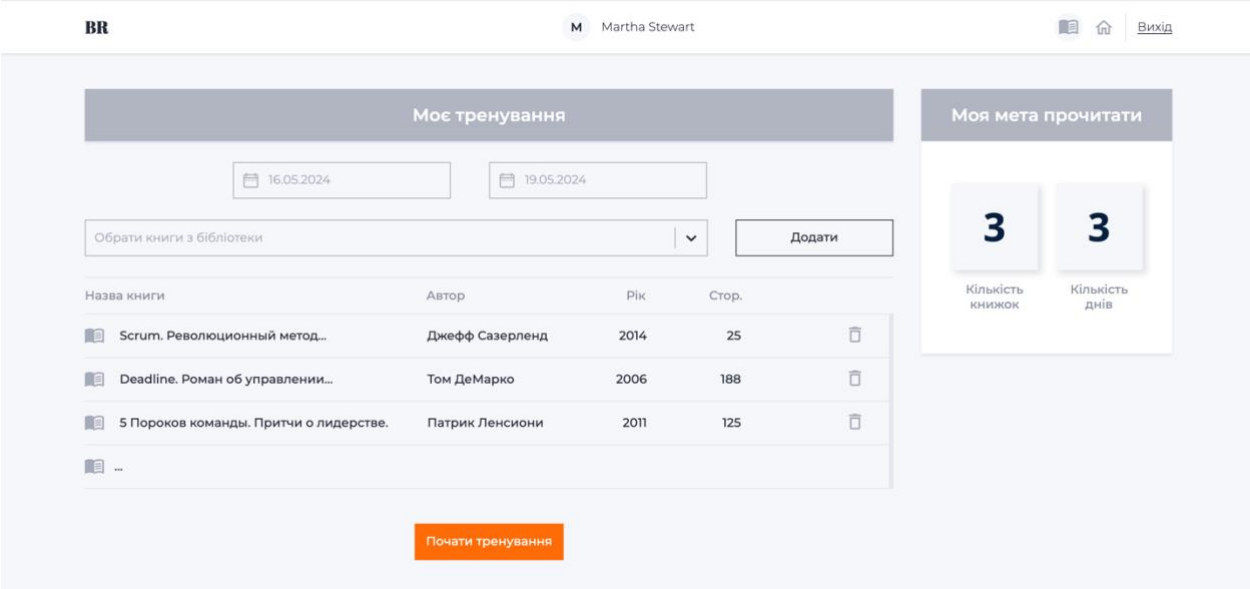

Рисунок 3.8 – Сторінка тренування читання (секція планування) з вибраними датами та книгами

Після натискання на кнопку «Почати тренування» відбувається перенаправлення користувача на секцію тренування – сторінки тренування (рис 3.9).

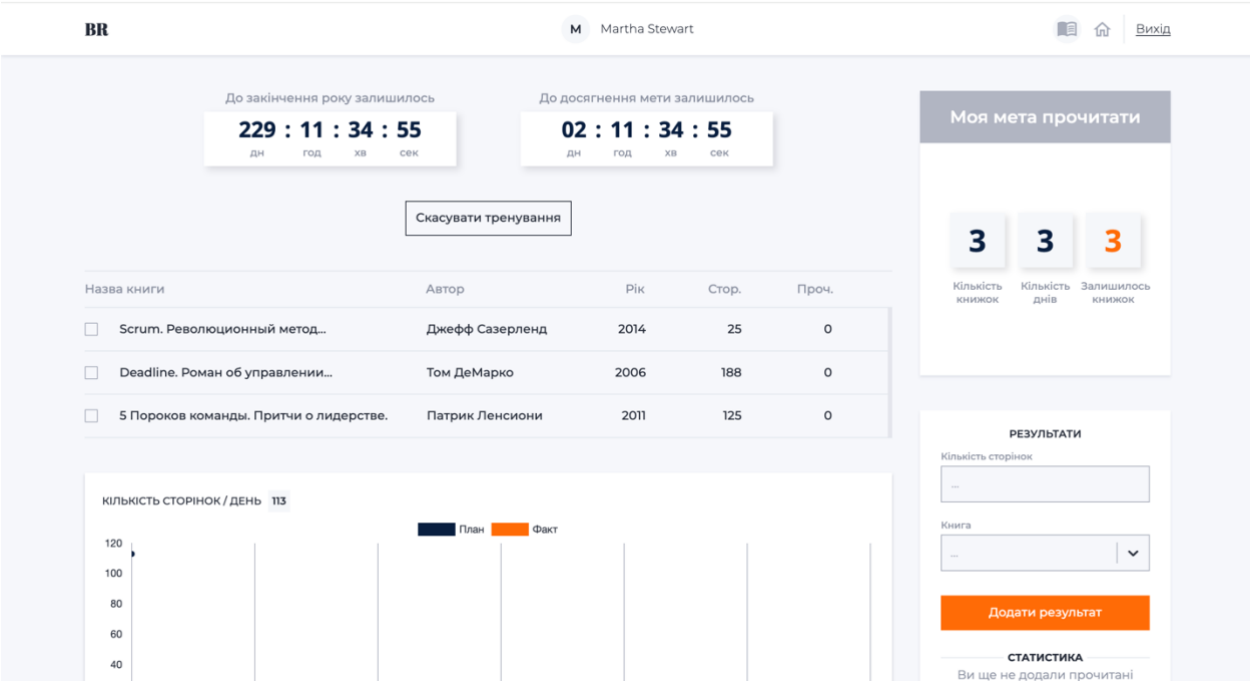

Рисунок 3.9 – Сторінка тренування читання (секція тренування)

Під верхньою панеллю знаходяться два таймери, що відлічують час до кінця року і час до досягнення мети. Нижче розташована кнопка «Скасувати тренування». Далі йде список книг, де вказані назва книги, автор, рік випуску, кількість сторінок та кількість прочитаних книг. Кожна книга має чекбокс який активується коли користувач прочитає книгу. Праворуч від таблиці розміщений блок «Моя мета прочитати», який показує кількість запланованих для читання книг, кількість днів для читання, та кількість книг, що залишилось прочитати. У нижній частині сторінки знаходиться графік, що показує кількість сторінок, прочитаних щодня, порівнюючи планові та фактичні прочитані сторінки. Праворуч від графіку розташований блок «Результати», де можна ввести кількість прочитаних сторінок, вибрати книгу з випадаючого списку і додати результат, натиснувши кнопку «Додати результат». Нижче цього блоку розташована секція «Статистика», яка показує детальну статистику по дням та кількості сторінок, які читає користувач (рис 3.10).

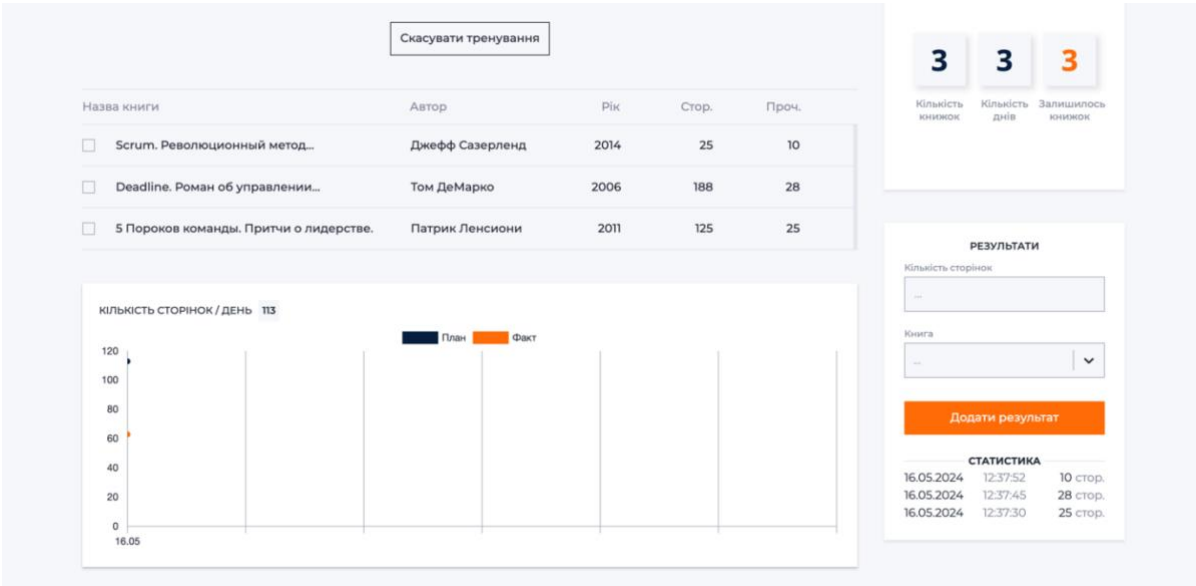

Рисунок 3.10 – Відображення статистики читання

Також є можливість переглядати динаміку статистики щодо кількості прочитаних сторінок. Під час додавання нових даних до статистики, автоматично оновлюється поле з кількістю прочитаних сторінок в списку книг.

Після завершення читання книги з’являється спливаюче вікно з повідомленням, яке підтверджує успішне завершення читання книги (рис 3.11).

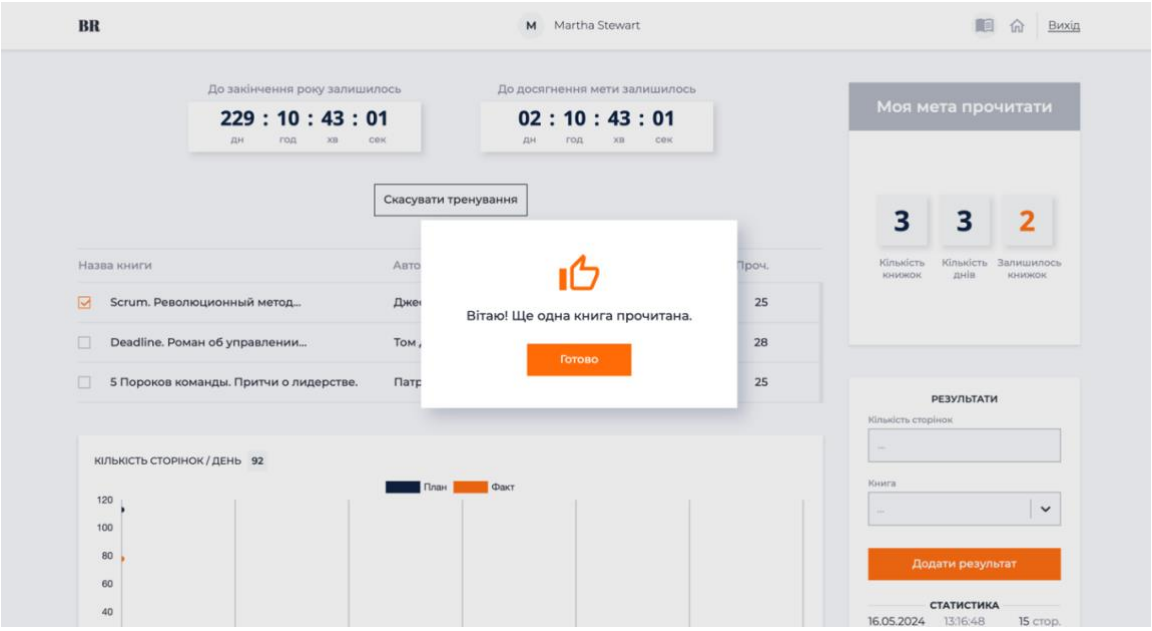

Рисунок 3.11 – Відображення повідомлення про прочитану книгу

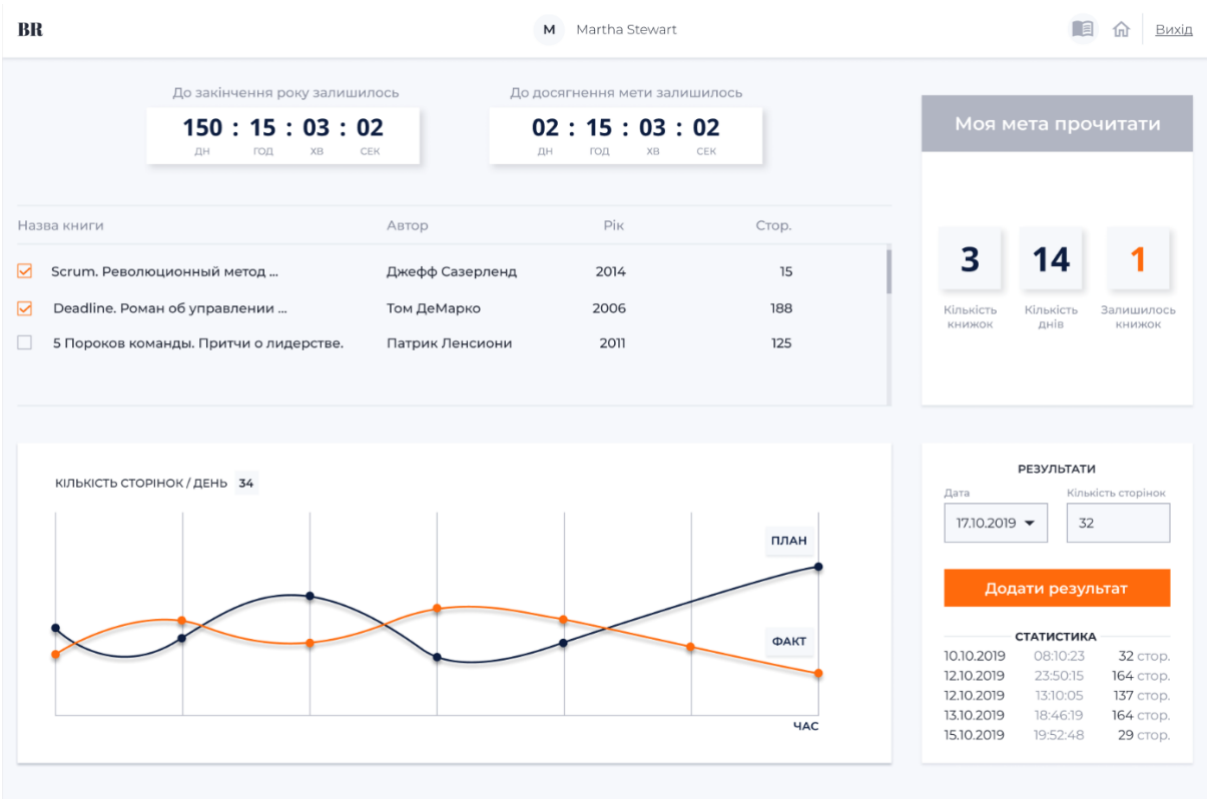

Рисунок 3.12 – Відображення графіку статистики читання

Після того як користувач прочитав книгу в списку книг відображається прапорець (checkbox), який сповіщає, що цю книгу вже прочитано. В правому блоці змінюється кількість книжок, які залишилось прочитати. А також можна побачити зміни в графіку. Графік порівнює планові та фактичні дані, показуючи кількість сторінок, які користувач планував прочитати (позначено чорним кольором), і кількість сторінок, які були фактично прочитані (позначено помаранчевим кольором). Це допомагає відстежувати прогрес читання і бачити, як користувач дотримується свого плану (рис 3.12).

Після початку тренування, якщо повернутися на сторінку бібліотеки, список книг буде розділено на три категорії: «Прочитано», «Читаю» та «Маю намір прочитати». Для книг, що належать до категорії «Прочитано», з'являться нове поле, яке відображає оцінку книги та кнопка «Резюме» (рис. 3.13).

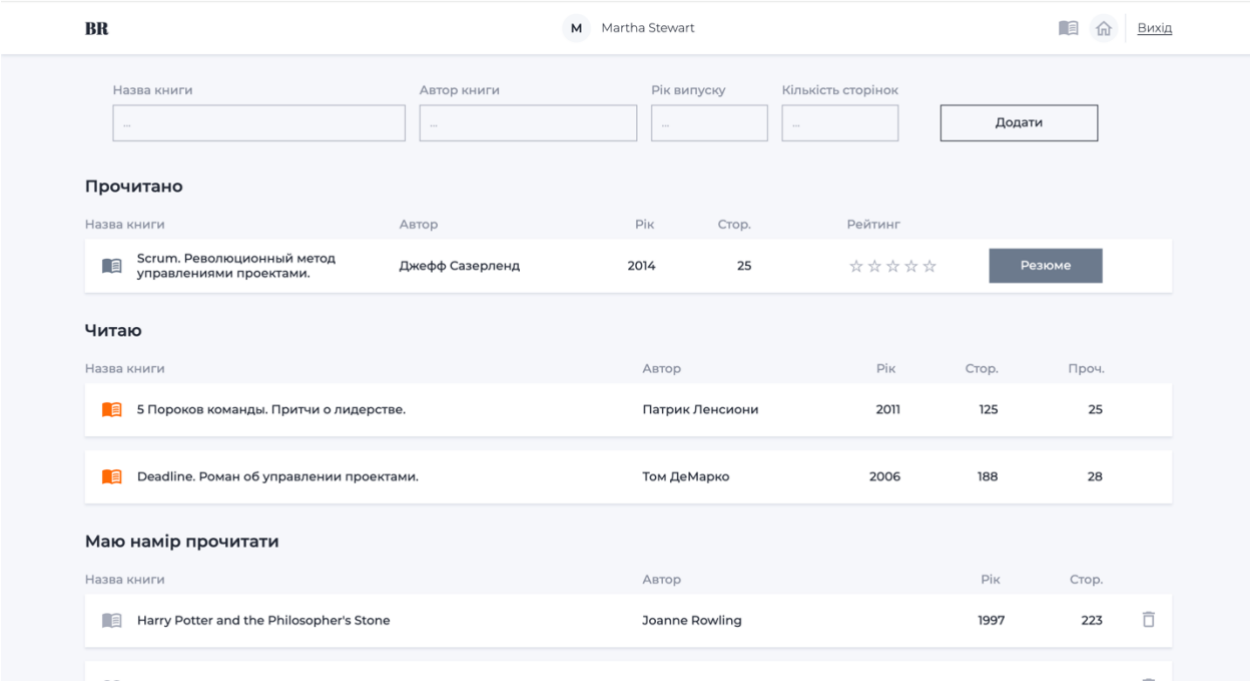

Рисунок 3.13 – Відображається категорій для списків книг

Після натискання кнопки «Резюме» відкривається модальне вікно, де користувач може обрати рейтинг і написати резюме до книги (рис. 3.14).

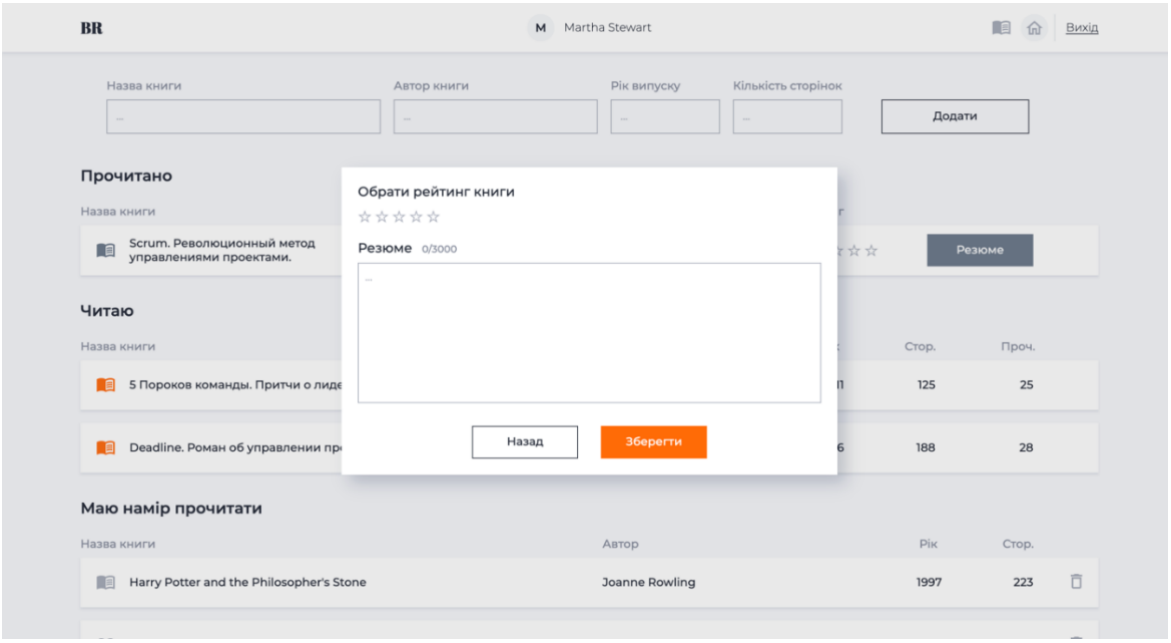

Рисунок 3.14 – Відображається модальне вікно оцінки книги

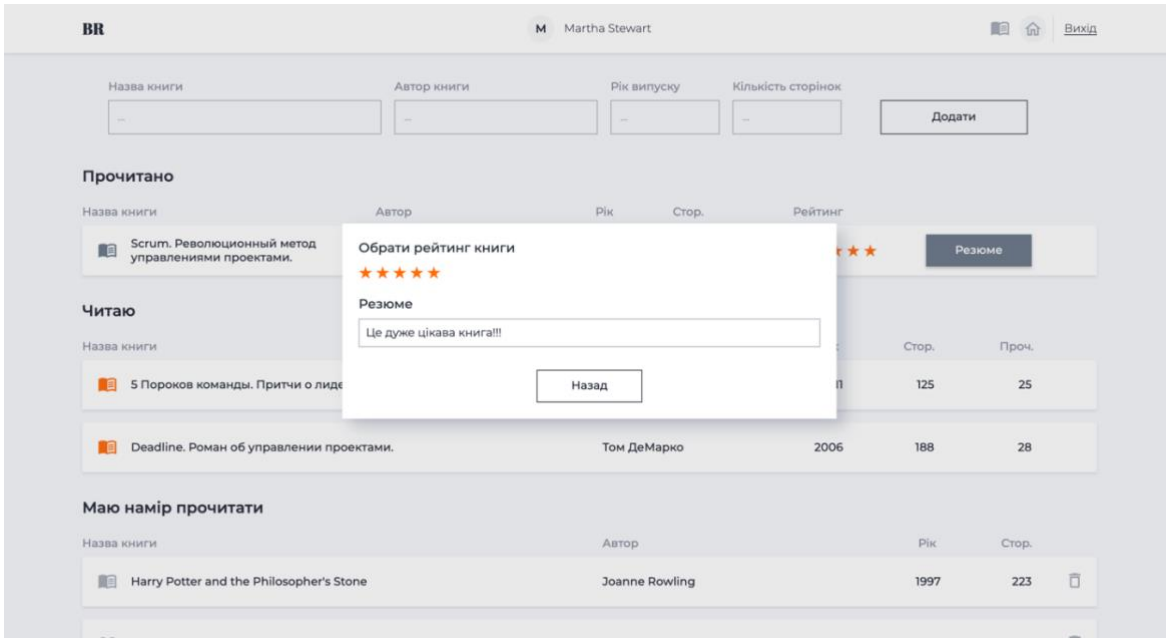

Рисунок 3.15 – Модальне вікно для книги, яка вже була оцінена

Якщо користувач бажає вийти зі свого облікового запису, він може скористатися кнопкою «Вихід» на верхній панелі. Після натискання цієї кнопки відкривається модальне вікно з двома опціями: «Відміна» і «Вийти». Натискання кнопки «Відміна» закриває модальне вікно без виходу з облікового запису, а натискання кнопки «Вийти» завершує сесію користувача та виходить з облікового запису (рис 3.16).

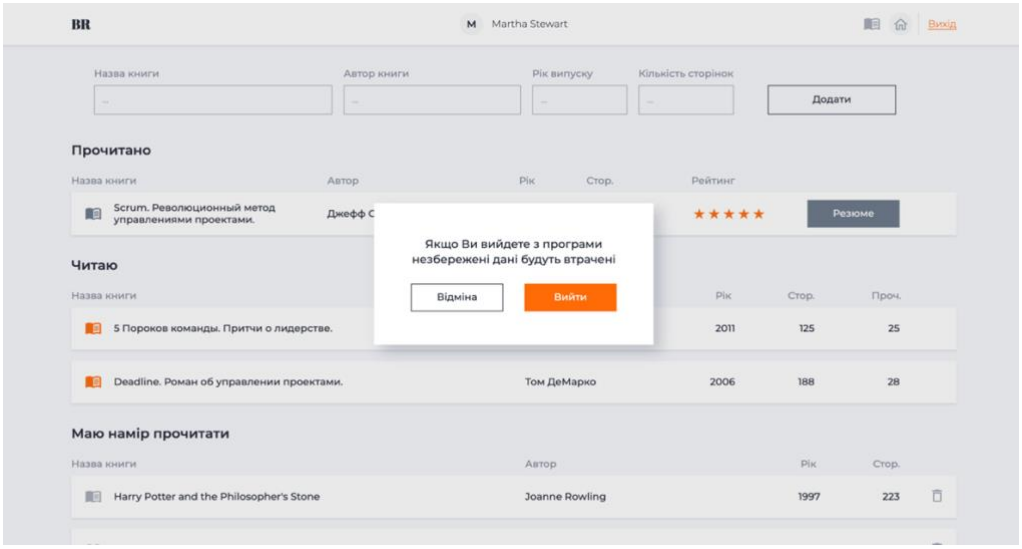

Рисунок 3.16 – Модальне вікно для підтвердження виходу с облікового запису

Варто відзначити, що вебзастосунок також оптимізований для мобільних пристроїв. На рисунках 3.17-3.19 показано, як виглядає інтерфейс застосунку «Персональна бібліотека» для каталогізації і управління прочитаними книгами на екранах різного розміру.

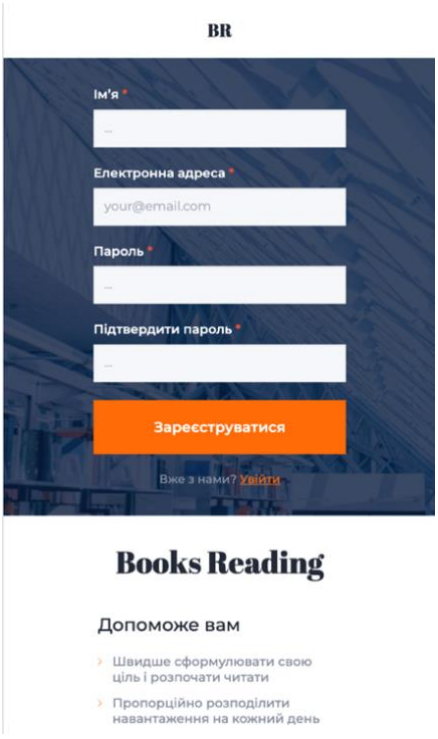

Рисунок 3.17 – Сторінка реєстрації вебзастосунку. Вигляд з роздільної здатності мобільного пристрою

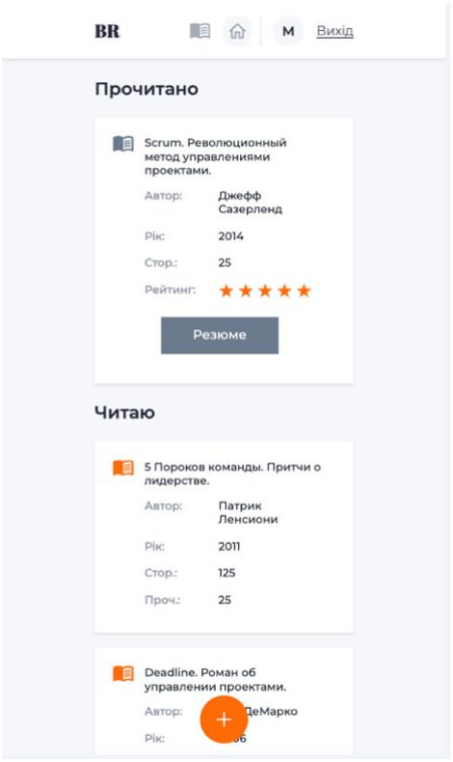

Рисунок 3.18 – Сторінка персональної бібліотеки. Вигляд з роздільної здатності мобільного пристрою

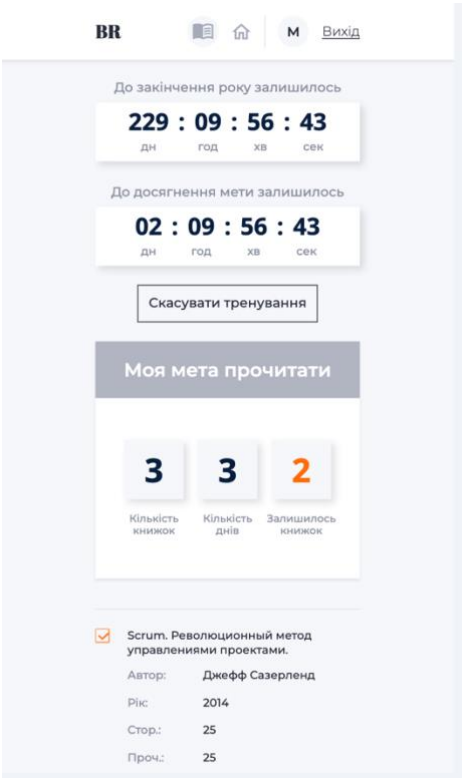

Рисунок 3.19 – Сторінка тренування (секція тренування). Вигляд з роздільної здатності мобільного пристрою

3.3 Заходи щодо поліпшення вебзастосунку

Застосунок «Персональна бібліотека» для каталогізації і управління прочитаними книгами має великий потенціал до удосконалення. Наприклад, у майбутньому можна вдосконалити такі аспекти проєкту:

– темна тема: додати можливість перемикання між світлою та темною темами. Це забезпечить користувачам зручне використання вебзастосунку в різних умовах освітлення, зменшить навантаження на очі у темний час доби та покращить загальний досвід користування;

– пошук та фільтрація: впровадити потужні інструменти пошуку та фільтрації, які дозволять користувачам швидко знаходити необхідні книги. Це включає пошук за автором, назвою, роком випуску, кількістю сторінок, а також можливість застосування кількох фільтрів одночасно для точнішого результату. Такі інструменти значно полегшать навігацію по великій бібліотеці та зекономить час користувачів;

– додати функціональність сортування книг за різними критеріями, такими як: сортування за кількістю сторінок від більшого до меншого і навпаки, що дозволить швидко знаходити найдовші або найкоротші книги у бібліотеці, сортування по алфавіту за назвою та автором, що забезпечить зручний спосіб знаходити книги певного автора або за конкретною назвою, сортування за датою додавання книги, що дозволить переглядати нещодавно додані книги або знаходити найстаріші записи, сортування за роком випуску від найновіших до найстаріших і навпаки, що дасть змогу легко знаходити найновіші видання або переглядати книги за певний період;

– великі списки книг за жанрами: запровадити функцію, яка буде робити запити для отримання великих списків книг, класифікованих за жанрами. Це дозволить користувачам легко знаходити книги за їхніми літературними уподобаннями. Кожен список буде містити книги певного жанру, таких як фантастика, детективи, історичні романи, наукова література та інше. Користувачі зможуть переглядати ці списки, ознайомлюватися з

різними книгами і швидко додавати їх до свого списку запланованих для прочитання книг. Цей процес буде автоматизованим, тобто користувачам не потрібно буде вручну вводити дані про книги. Достатньо буде обрати книгу зі списку та натиснути кнопку «Додати», після чого книга автоматично з’явиться у списку запланованих для читання;

– соціальна взаємодія: впровадження функцій соціальної взаємодії дозволить користувачам активно спілкуватися та обмінюватися досвідом читання. Додати можливість створювати та приєднуватися до спільнот за інтересами, де учасники зможуть обговорювати книги, ділитися новинами та обмінюватися думками. Це зробить процес читання більш інтерактивним та цікавим.

Отже, вебзастосунок має широкий спектр можливостей для майбутніх покращень, що дозволить значно розширити його функціонал та підвищити зручність користування.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений застосунок «Персональна бібліотека» для каталогізації і управління прочитаними книгами.

Основною метою роботи було створення зручного та ефективного інструменту для користувачів, який дозволяє легко організовувати свою особисту бібліотеку, планувати читання, відстежувати прогрес та зберігати враження від прочитаних книг.

Під час розробки були використані сучасні технології та бібліотеки. Для реалізації клієнтської частини застосунку використовувалися JavaScript, React, Redux-Toolkit, TypeScript, Styled Components. Для розробки серверної частини застосунку використовувалися такі технології: Node.js, Express.

Базу даних застосунку створено відповідно до всіх вимог, використовуючи MongoDB.

Робочу версію вебзастосунку успішно розгорнуто і вона функціонує відповідно до запланованих вимог. Після детального аналізу виявлено можливості для подальшого вдосконалення застосунку. Ці покращення охоплюють як технічні аспекти, так і покращення користувацького досвіду, що дозволить зробити застосунок ще більш ефективним та зручним для користувачів.

Результатом роботи є функціональний інструмент, який використовується для каталогізації та управління прочитаними книгами.

Результати роботи апробовано у вигляді тез доповідей під час XXVIII Міжнародного молодіжного форуму «Радіоелектроніка і молодь у XXI столітті», онлайн конференції «Комп’ютерний зір, системний аналіз та математичне моделювання» [33].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What is a web application? URL: https://groupeshift.ca/en/blog/what-is-a-web-app/ (дата звернення 26.04.2024).

2. Що таке веб додаток? Різниця між сайтом, вебдодатком, SPA I PWA. URL: https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy (дата звернення 26.04.2024).

3. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, International Journal of Academic and Applied Research, 7(9), pp. 57-70.

4. Monolith vs Microservices Architecture: Which Is the Best Choice for Your Business? URL: https://gowombat.team/blog/posts/monolith-vs-microservices-architecture-which-is-the-best-choice-for-your-business (дата звернення 27.04.2024).

5. Tvoroshenko, I., Ahmad, M. A., Mustafa, S. K., Lyashenko, V., & Alharbi, A. R. (2020). Modification of models intensive development ontologies by fuzzy logic.

6. Ahmad, M. A., Tvoroshenko, I., Baker, J. H., & Lyashenko, V. (2019). Modeling the structure of intellectual means of decision-making using a systemoriented nfo approach.

7. Tvoroshenko, I. S., & Tabashnyk, V. A. (2018). Development of a spatial model of geoinformation support for people with disabilities in wheelchairs in Kharkiv. Collection of scientific works of KhNUPS, 1(55), 122-128.

8. Кобилін, О.А., & Творошенко, І.С. (2021). Методи цифрової обробки зображень: навч. посібник. *Харків: ХНУРЕ*.

9. Goodreads. URL: https://www.goodreads.com/ (дата звернення 27.04.2024).

10. Libib. URL: https://www.libib.com/ (дата звернення 27.04.2024).

11. Book Catalogue. URL: https://book-catalogue.com/ (дата звернення 27.04.2024).

12. Гороховатський, В.О., & Творошенко, І.С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник.

13. How Does React Work? URL: https://www.uxpin.com/studio/blog/how-react-works/ (дата звернення 28.04.2024).

14. Тітов С.В., Тітова О.В. Аналіз якості інформаційно-технологічного забезпечення обробки документів у хмарному сервісі Microsoft OneDrive / С. В. Тітов, О. В. Тітова // Вісник ХДАК / Зб. наук. праць. – Х: ХДАК., 2018. – Вип. 52. – С. 142-148.

15. Styled Components: https://styled-components.com/ (дата звернення 28.04.2024).

16. Творошенко, І.С. (2021). Технології прийняття рішень в інформаційних системах: навч. посібник. *Харків: ХНУРЕ*.

17. Тітов, С. В., & Тітова, О. В. (2015). Оцінка юзабіліті освітніх сайтів: методи і технології. Вісник Харківської державної академії культури. Серія: Соціальні комунікації, (47), 127-134.

18. Руденко, Д. О., & Бондар, В. О. (2020, November). Огляд можливостей використання стратегій об'єктно-орієнтованого маппінгу для зіставлення сутностей при розробці web додатків. In The 3 rd International scientific and practical conference—Priority directions of science and technology development‖(November 22-24, 2020) SPC—Sci-conf. com. ua‖, Kyiv, Ukraine. 2020. 1488 p. (p. 377).

19. Sitnikov D., Titova O., Minukhin S., Kovalenko A., Titov S. (2018). Informativity of Association Rules from the Viewpoint of Information Theory. Conference: 2018 IEEE International Scientific-Practical Conference Problems of Infocommunications. Science and Technology.

20. Kobylin O., Gorokhovatskyi V., Tvoroshenko I., and Peredrii O. (2020) The application of non-parametric statistics methods in image classifiers based on

structural description components, *Telecommunications and Radio Engineering*, 79(10), pp. 855-863.

21. Шафроненко, А. Ю., Бодянський, Є. В., & Руденко, Д. О. (2023). Модифікований рекурентний метод достовірної нечіткої кластеризації з використанням оптимізаційної процедури на основі косяків риб. Системи обробки інформації, (1 (172)), 92-96.

22. Бодянський, Є. В., Шафроненко, А. Ю., & Климова, І. М. (2021). Метод адаптивної достовірної нечіткої кластеризації даних на основі еволюційного алгоритму. Збірник наукових праць Харківського національного університету Повітряних Сил, (2 (68)), 80-83.

23. O. Chorna, P. Didyk, S. Titov, O. Titova. Usage of Clustering Algorithms for Automating Route Planning in Transportation Routing Tasks // Системи обробки інформації. №2(173), 2024, с. 58-62.

24. Learn to code with Visual Studio Code. URL: https://code.visualstudio.com/docs/editor/whyvscode (дата звернення 05.05.2024).

25. Mongoose. URL: https://monib-bormon.medium.com/what-is-mongoose-c1bc3031cc08 (дата звернення 09.05.2024).

26. Valeria, B., Hryhorii, K., Diana, R., & Vyacheslav, L. (2023). Phase Portrait Models as a Tool for Analyzing Banking Activities.

27. Lyashenko, V., & Rudenko, D. (2021). Modeling Deformation of Spur Gear.

28. Bcrypt. URL: https://medium.com/@valevpn/what-is-bcrypt-and-how-does-it-work-bef43ee8762d (дата звернення 09.05.2024).

29. Introduction to JSON web tokens. URL: https://jwt.io/introduction (дата звернення 10.05.2024)

30. Андрєєва, А. Ю., & Руденко, Д. О. (2022, October). Як блокчейн токенізація змінює світ. In The 2 nd International scientific and practical conference “Science and innovation of modern world”(October 26-28, 2022) Cognum Publishing House, London, United Kingdom. 2022. 948 p. (p. 233).

31. Gorokhovatskyi, V., Tvoroshenko, I., Kobylin, O., & Vlasenko, N. (2023). Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19-27.

32. Kovalenko, A., Titov, S., Titova, E., Cherna, O. Estimation of requirements to signal parameters at V-shaped frequency distribution in mathematical model of multi-position transmitter system. Radiotekhnika, 2(209), 2022. P. 178–184.

33. Уткін Є. І. Розробка застосунку «Персональна бібліотека» для каталогізації і управління прочитаними книгами. Радіоелектроніка та молодь у XXI столітті: тези доповідей 28-го Міжнародного молодіжного форуму (Харків, 16–18 квітня 2024 р.). Харків: ХНУРЕ, 2024. Т. 7. С. 130-131.