ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської кваліфікаційної роботи

на ступінь вищої освіти бакалавр

на тему: « Розробка web-додатку для керування процесом читання з використанням бібліотеки React»

Виконала: студентка 4 курсу, група ПД-42

спеціальності

121 Інженерія програмного забезпечення

(шифр і назва спеціальності)

Федоренко А.А

(прізвище та ініціали)

Керівник Трінтіна Н.А

(прізвище та ініціали)

Рецензент

(прізвище та ініціали)

Нормоконтроль

(прізвище та ініціали)

Київ - 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти – «Бакалавр»

Спеціальність – 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного

забезпечення

_____О.В Негоденко

«____»_____2023 року

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Федоренко Анна Андріївна

(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка web-додатку для керування процесом читання з використанням бібліотеки React»

Керівник роботи к.т.н., доцент Трінтіна Н.А.

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджено наказом вищого навчального закладу від – «24» лютого 2023 року № 26

2. Строки подання студентом роботи «1» червня 2023 року
3. Вхідні данні до роботи:

   1.1 Науково-технічна література пов'язана з розробкою

   1.2 Середовище розробки Visual Studio Code

   1.3 Офіційна документація React

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
    4.1 Аналіз предметної галузі
    4.2 Вибір технологій та середовища розробки
    4.3 Розробка структури web-додатку для керування процесом читання з використанням бібліотеки React
    4.4 Програмна реалізація додатку
5 Перелік демонстраційного матеріалу
    5.1 Титульний слайд
    5.2 Об’єкт, мета та предмет дослідження
    5.3 Задачі дипломної роботи
    5.4 Аналіз аналогів
    5.5 Переваги та недоліки аналогів
    5.6 Вимоги до додатку
    5.7 Програмні засоби реалізації
    5.8 Діаграма прецедентів
    5.9 Діаграма діяльності
    5.10 Діаграма класів
    5.11 Схема бази даних
    5.12 Екрані форми додатку
    5.13 Апробація результатів дослідження
    5.14 Висновки
6. Дата видачі завдання «25» лютого 2023 року

КАЛЕДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строки виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	01.04.2023-02.04.2023	Виконано
2	Аналіз існуючих аналогів та актуальності додатку	03.04.2023-05.04.2023	Виконано
3	Вибір інструментів для розробки додатку	05.04.2023-10.04.2023	Виконано
4	Проектування веб-додатку	10.04.2023-23.04.2023	Виконано
5	Розробка функціоналу для веб-додатку	23.04.2023-10.05.023	Виконано
6	Вступ, висновки, реферат	10.05.2023-18.05.2023	Виконано
7	Попередній захист роботи	19.05.2023	Виконано

Студент _____ Федоренко А.А.

Керівник роботи _____ Трінтіна Н.А.

РЕФЕРАТ

Текстова частина бакалаврської роботи 61 с., рис. 29, 25 джерел.

Об'єкт дослідження – керування процесом читання.

Предмет дослідження- програмне забезпечення для керування процесом читання.

Мета роботи – спрощення керування процесу читання за рахунок використання розробленого web-додатку за допомогою бібліотеки React.

Проведено аналіз предметної галузі, існуючих додатків для керування процесом читання. В результаті аналізу було визначено основні недоліки додатків аналогів та потреби користувача.

Проаналізовано можливості мови програмування JavaScript, мови розмітки HTML, SCSS, бібліотеки React, платформу Firebase зокрема її модуля Realtime Database, бібліотеку компонентів інтерфейсу користувача MUI, середовище розробки Visual Studio Code.

Виконано проектування додатку з використанням діаграм UML, розроблено діаграму прецедентів та діаграму класів.

Галузь використання – web-додаток для керування процесом читання може бути корисним для будь-якої галузі, де присутнє вивчення та акумуляція інформації.

ЗМІСТ

ВСТУП

Актуальність теми. В сучасному світі люди все більше звертають увагу на своє освітнє та професійне зростання, що вимагає від них постійного навчання та саморозвитку. Тому розробка web-додатку, який допоможе керувати процесом читання, є дуже актуальною. Такий додаток допоможе відслідковувати свій прогрес читання, зберігати та організовувати знайдену інформацію для подальшого використання, може забезпечувати можливість відзначення важливих місць в тексті, створення списків для читання. Таким чином, веб-додаток для керування процесом читання може бути актуальним для широкого кола користувачів, які бажають організовувати свої книги та отримати функції для поліпшення процесу читання та збільшення свого рівня освіти.

Об'єкт дослідження –керування процесом читання.

Предмет дослідження- програмне забезпечення для керування процесом читання.

Мета роботи – спрощення керування процесом читання за рахунок використання розробленого web-додатку за допомогою бібліотеки React.

Щоб досягнути поставленої мети необхідно виконати наступні завдання:

1. Проаналізувати існуючі аналоги і визначити переваги та недоліки.
2. Розробити вимоги до web-додатку на основі аналізу існуючих аналогів та потреб користувача.
3. Проаналізувати технічні засоби, які використовуються для розробки, та обрати необхідні для створення додатку.
4. Розробити додаток на основі проведеного аналізу потреб користувача.

Галузь використання – додаток може відкрити кожен користувач будь-якої операційної системи через такі браузери як Chrome, Safari, Mozilla Firefox. Додаток допоможе систематизувати книги користувача та дасть можливість читати та завантажувати книги з пристрою користувача. Виділяти та зберігати важливу інформацію в книзі.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Поняття «web-додаток»

Web-додаток – це програма, яка використовує веб-браузер для виконання певних функцій. Такі додатки мають незліченні переваги. Зокрема їх можна запустити у таких веб-браузерах, як Mozilla Firefox, Safari, та Google Chrome, що робить їх більш доступними.

Одною з переваг є оновлення web-додатку. Воно здійснюється на серверному боці, що дозволяє оновлювати функціонал додатку без необхідності встановлення оновлення на кожен пристрій окремо. Це гарантує, що користувачі матимуть найактуальнішу версію додатку та забезпечує однаковий функціонал для всіх.

Такі програми не потрібно встановлювати на жорсткий диск, оскільки до них можна отримати доступ повністю онлайн. Крім того, веб-додатки можна запустити на кількох платформах за умови сумісності браузерів.

Веб-додатки бувають трьох видів PWA (Progressive Web Application), SPA (Single Page Application) та MPA (Multi-Page Application).

1.1.1 PWA (Progressive Web Application)

PWA (Progressive Web Application) – це тип веб-додатку, який працює на будь якому пристрою як веб-сторінка так і мобільна програма. Ці додатки мають зовнішній вигляд та поведінку, що схожа на веб-сторінку, проте вони також надають функціональні можливості, які ідентичні мобільним додаткам: можуть функціонувати в режимі офлайн, відправляти push-повідомлення та використовувати функції пристрою користувача.

Основні переваги використання веб-додатків PWA включають:

- Кросплатформеність: підтримують усі популярні браузери та операційні системи;
- Легкий доступ та встановлення: додаток можна встановлювати безпосередньо з веб-сайту, без необхідності відвідувати магазин додатків;

- Швидка робота: PWA додатки можуть працювати швидше ніж звичайні веб-сайти, оскільки вони можуть зберігати дані локально, що дає змогу їм працювати онлайн та забезпечує швидку відповідь на дії користувача;
- Невеликий розмір: такі додатки порівняно з мобільними мають невеликий розмір, що дозволяє швидше завантаження та використовувати менше ресурсів пристрою;

Однак, не зважаючи на всі плюси PWA додатків є й недоліки, а саме:

- Роботу таких додатків можуть обмежувати деякі браузери та операційні системи;
- Офлайн робота обмежена;

1.1.2 SPA (Single Page Application)

SPA (Single Page Application) – веб-сайти або веб-додатки, які працюють без перезавантаження.

Така архітектура має специфічну особливість, коли всі компоненти додатку знаходяться на одній сторінці та завантажуються під час ініціалізації та додатково після запиту користувача. При завантажуванні нових модулів у SPA, оновлюється лише частина контенту, що дозволяє швидше реагувати та економити обсяг передачі даних.

Переваги SPA додатків:

- Додаток можна використовувати з будь-якого пристрою, який має доступ до інтернету;
- Розмір даних та додатка не обмежений пам’яттю пристрою;
- Односторінкові сайти або додатки підвищує продуктивність роботи та заощаджує час на повторне завантаження даних;

Недоліки SPA додатків:

- Необхідний доступ до мережі для використання софту.
- Ускладнює процес індексації пошуковими системами усіх модулів додатка;

- Якщо у користувача відключенні відображення JS-елементів у браузері, то SPA додаток може не працювати;

1.1.3 MPA (Multi-Page Application)

MPA (Multi-Page Application) – програма з багатьма сторінками, яка складається з веб-сторінок, що завантажуються при переході користувачів до різних розділів веб-сайту. При запиті користувача сервер надає всі необхідні ресурси (HTML, CSS, та JavaScript) для створення нової сторінки.

Переваги таких додатків наступні:

- Така архітектура дозволяє легко оптимізувати кожну сторінку для пошукових систем;
- Розробка багатосторінкової програми вимагає меншого набору технологій;

Недоліки MPA:

- Важко відокремити інтерфейс і бек-енд, такими чином це ускладнює роботу розробників;

1.2 Актуальність веб-додатків для керування процесом читання

В сучасному світі люди все більше звертають увагу на своє освітнє та професійне зростання, що вимагає від них постійного навчання та саморозвитку. Більшість людей швидко переглядають книги, та часто забувають подробиці сюжету та героїв протягом кількох місяців, якщо не днів. Додатки для керування процесом читання мають допомагати зберігати та організовувати знайдену інформацію для подальшого використання. Вони допомагають ефективніше використовувати свій час та підвищать ефективність навчання і розвитку.

Такі додатки мають легко зберігати та організовувати книги. Мати можливість швидко переглянути книги на поличках та оновлювати їх ключову інформацію. Наявність списків для книг одна з ключових функцій у таких додатках. Адже за рахунок упорядкованих списків для читання, користувачеві буде досить

легко запам’ятати що він читав у минулому, що читає на даний момент і що планує читати у майбутньому.

Зазвичай такі додатки стають ефективним інструментом для підвищення мотивації до читання та розвитку навичок самоорганізації. Крім того додатки для керування процесом читання можуть бути корисними для студентів, дослідників та інших людей, які часто працюють з великою кількістю літератури та потребують зручного способу її організації.

Наявність функції електронної книги, наддасть можливість читати власні книги у веб-додатку, без потреби завантажувати додатковий софт на свій пристрій.

Одна з переваг таких додатків, місце де користувачі записують свої враження від прочитаних книг, відзначають, що сподобалося або не сподобалося в кожній книзі, висловлювати власні думки, роздуми. Це допомагає ще глибше проаналізувати свої враження від літератури та розвивати критичне мислення.

1.3 Огляд існуючих веб-додатків для керування процесом читання

1.3.1 Goodreads

Goodreads – це веб-сайт призначений для книголюбів, які можуть відстежувати книги, які вони читають, переглядати книги та ділитись рекомендаціями з іншими користувачами. Після реєстрації ви зможете створювати полички для своїх книг, що дозволяють зручно їх сортувати.

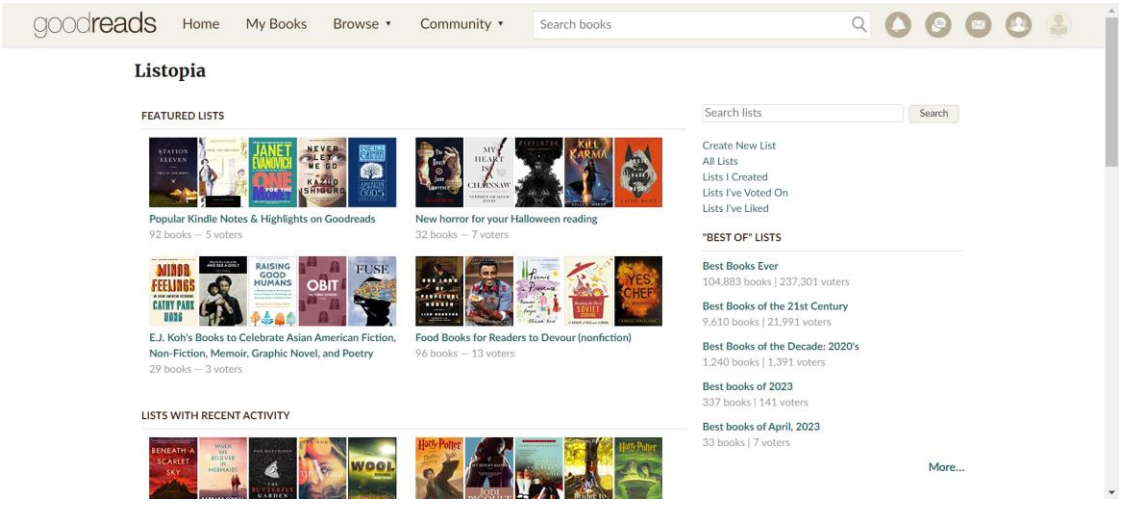


Рисунок 1.1 – Приклад інтерфейсу веб-сайту Goodreads

Платформа Goodreads досить популярна серед книголюбів, та має ряд переваг і недоліків.

Переваги

- Goodreads дозволяє залишати відгуки на книги, оцінювати та отримувати рекомендації на основі вподобань користувача;
- Дозволяє користувачам відслідковувати процес читання, додавати книги до списку «Прочитано», «Читаю зараз», «Бажаю прочитати»;
- Goodreads дозволяє користувачам взаємодіяти з іншими книголюбами, ділитися своїм прогресом читання, обговорювати книги;

Недоліки

- Рейтинг та рецензії на Goodreads можуть бути суб'єктивними, оскілки вони базуються на відгуках та оцінках інших користувачів. Це може вплинути на вибір книги для читання та створювати певний тиск на користувача при виборі книги;
- Відсутня інтеграція з Google Book;
- Відсутня функція електронної книги на веб-сайті;

1.3.2 Scribd

Scribd – ефективний та надійний сервіс, надає своїм користувачам доступ до книг, документів, журналів та наукових статей.

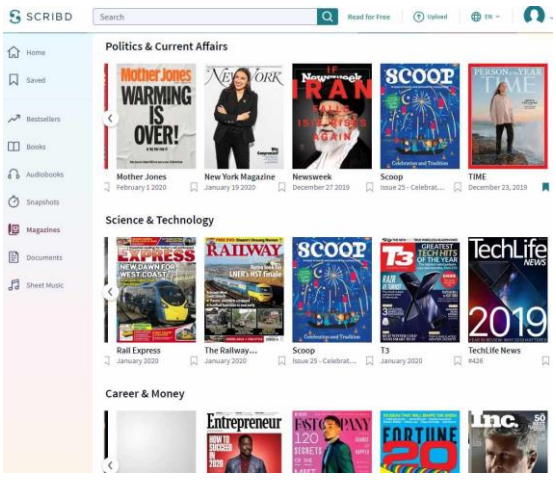


Рисунок 1.2 – Приклад інтерфейсу Scribd [11].

Переваги Scribd:

- Надає доступ до книг, журналів, наукових статей, аудіокниг;
- Є можливість читати книги у додатку;
- Має досить простий дизайн та просту навігацію;
- Можливість читати офлайн;

Недоліки

- Відсутня інтеграція з Google Book;
- Не можливо завантажувати книги у додаток;

1.3.3 LibraryThing

LibraryThing –веб-застосунок для відстежування книг, досить функціональний та простий у використанні.

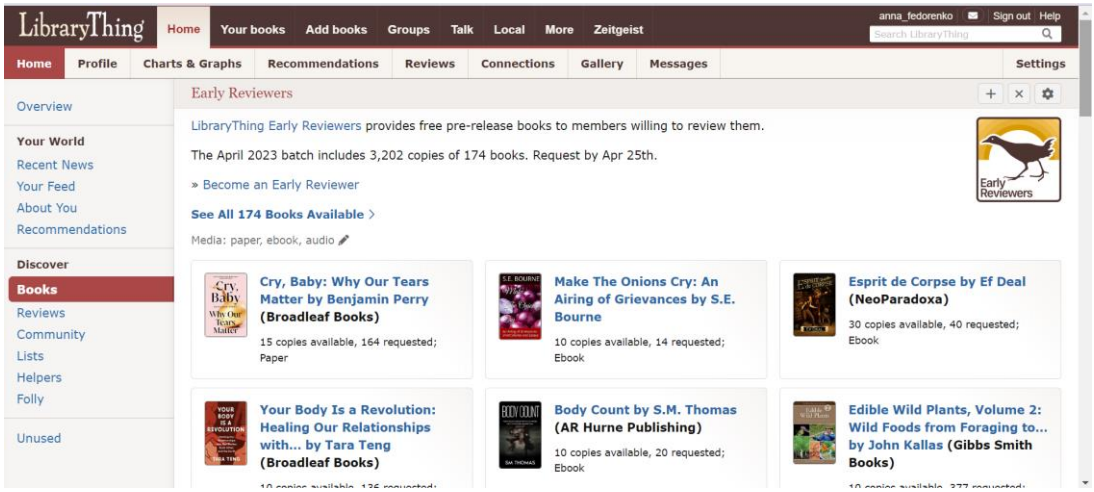


Рисунок 1.3 – Приклад інтерфейсу LibraryThing.

Переваги LibraryThing

- Можливість імпортувати данні з понад 2000 бібліотек;
- Пропонує багато різноманітних функцій, таких як можливість додавання книг, організовувати бібліотеку за різними категоріями, можливість додавати відгуки та оцінки;

Недоліки

- Відсутня підтримка української мови;

- Відсутня інтеграція з Google Book;
- Має обмеження на кількість книг, які можна додати до бібліотеки;

1.4 Формування вимог для веб-додатку для керування процесом читання

Виходячи з результатів аналізу додатків аналогів, формуємо наступні вимоги до розроблюваного додатку:

1. Додаток має підтримувати українську мову, адже на ринку майже немає схожих веб-додатків українською мовою.
2. Додаток має мати функцію електронної книги. Це наддасть можливість користувачам зручно отримувати доступ до своїх книг безпосередньо у веб-браузері без наявності окремих додатків на пристрої.
3. Додаток має мати зручний та зрозумілий інтерфейс, оскільки він забезпечить позитивний досвід взаємодії користувача і додатку. Зрозумілий інтерфейс допоможе уникнути помилок та неправильних дій користувача.

1.5 Висновки до першого розділу

У першому розділі були розглянуті наступні теми: поняття «web-додаток» та які є види веб-додатків, актуальність розробки веб-додатку для керування процесом читання, огляд існуючих аналогів додатків для керування процесом читання.

На основі аналізу отримуємо наступні результати. Розробка веб-додатку для керування процесом читання є досить актуальною, адже багато людей потребують організувати свої книг для зручності відстежування свого процесу читання.

Жоден з розглянутих аналогів не мав таких переваг, як підтримка української мови та тільки один з аналогів має повністю безкоштовну версію.

2. ВИБІР ТЕХНОЛОГІЙ ТА СЕРЕДОВИЩА РОЗРОБКИ

2.1 Середовище розробки

2.1.1 Visual Studio Code

Visual Studio Code – це потужний редактор вихідного коду, розроблений компанією Microsoft для Windows, MacOS та Linux.

Вперше редактор був представлений на конференції Build у квітні 2015 році.

Visual Studio Code має ряд переваг перед іншими редакторами, а саме:

- Підтримує кілька мов програмування, він легко виявляє будь-яку помилку чи посилання на іншу мову;
- Функція IntelliSense виявляє, якщо якийсь фрагмент коду не завершений, крім того створюється автоматично синтаксис змінних та їх оголошення;
- У разі якщо користувач захоче використовувати мову програмування яка не підтримується редактором, він містить досить багато розширень які легко можна завантажити;
- Visual Studio Code підключений до Git, це забезпечить вчасне збереження коду. Також редактор можна підключити до будь-якого іншого сховища;
- Має вбудовану веб-підтримку для веб-додатків;
- Вбудована підтримка розробки Node.js, JavaScript, TypeScript, JSX/React, HTML, CSS, SCSS, Less та JSON;
- Має відкритий вихідний код що дає можливість додати свій внесок до спільноти на GitHub;
- Інтеграція з інструментами збірки та сценарії для виконання завдань;

На рисунку 2.1 зображено приклад інтерфейсу Visual Studio Code

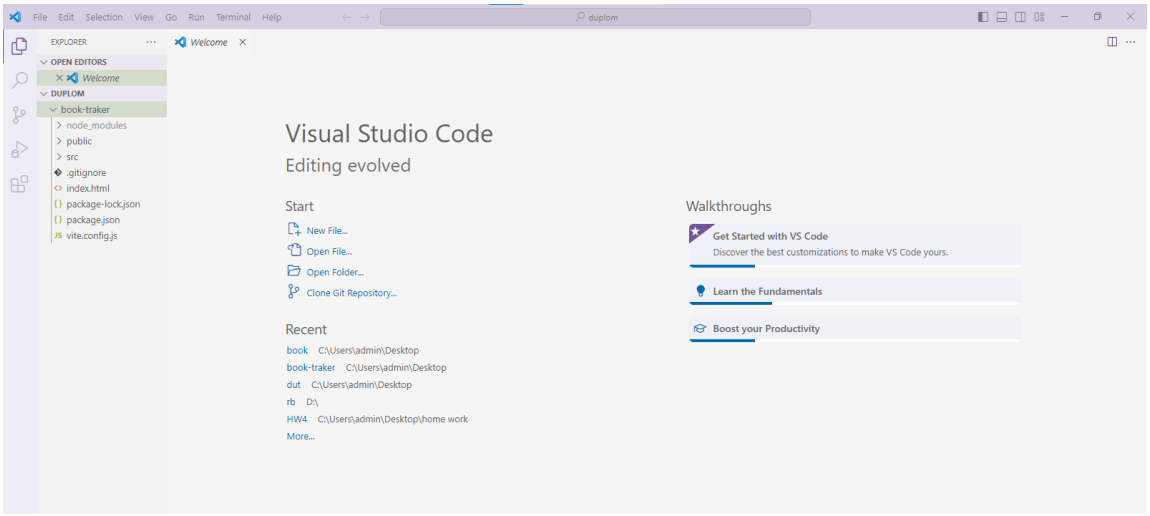


Рисунок 2.1 – Інтерфейс Visual Studio Code.

2.2 Обрані технології розробки

2.2.1 Безсерверна модель розробки

Безсерверна модель розробки – дозволяє створювати та запускати додатки без необхідності керувати сервером. Хмарний провайдер динамічно розподіляє ресурси для виконання та масштабування програми виходячи з вхідних даних. Ключовою особливістю безсерверної архітектури є автоматичне надання та масштабування середовища виконання, та також керування потужністю. Основна мета створення – це спростити розгортання коду у виробництві.

Безсерверна модель поєднує в собі дві ідеї, функція-як-сервіс (FaaS) та бекенд-як-сервіс (BaaS). Основна увага приділяється FaaS, що дозволяє хмарним службам виконувати код без необхідності в повністю налаштованих екземплярах.

Така модель розробки має ряд переваг:

- за допомогою FaaS розробники можуть створювати прості функції, які виконують одні мету, наприклад виклик API;
- безсерверна архітектура скоротить час виходу додатку на ринок. Так як немає необхідності витрачати час на підготовку серверу ;
- За рахунок використання вбудованих функцій безпеки на платформі така модель розробки є більш безпечною;

- завдання пов’язані з серверами виконуються в фоновому режимі;

Безсерверна модель розробки чудово підходить для веб-додатків, значно скорочує час розробки, так як дозволяє уникнути розробки власного сервісу. Та дає можливість приділяти більше уваги для написання бізнес-логіки додатку.

2.2.2 HTML

HTML (HyperText Markup Language) – це мова розмітки для створення веб-сторінок. HTML вважається одним з трьох основних інструментів для створення веб-сайтів. Він допомагає браузерам зрозуміти структуру та стиль документа для перегляду в інтернеті.

Основними компонентами HTML документа є елементи та теги. Ці елементи повідомляють браузеру як відображувати вміст документу. Базова структура HTML сторінки складається:

- Декларація типу документу – позначається як <!DOCTYPE html>, відображається в самому верху документа. Цей елемент повідомляє браузер яка версія HTML була використана;
- Тег <html> - корінь документа HTML. Він є контейнером для всіх інших елементів HTML. В ньому можна вказувати мову документа;
- Тег <head> - містить метадані, які описують інформацію про сторінку, зазвичай це назва документу, стилі, скрипти та іншу метаінформацію;
- Тег <body> - це основна частина документа, вона містить інформацію яку браузер відображає на екрані. Наприклад заголовки, зображення, таблиці, списки, абзаци, гіперпосилання;

На Рисунку 2.2 зображено приклад структури HTML документа.

Рисунок 2.2 – Структура HTML документа.

Файли HTML зберігаються з розширенням .html, потім ці файли можуть бути прочитанні веб-браузером та відображатись у вигляді веб-сторінки.

Переваги та недоліки використання HTML:

- запускається в кожному браузері;
- Є можливість інтегрувати з серверними мовами програмування, наприклад такими як PHP;
- Має чистий і послідовний вихідний код;

Недоліки:

- В основному використовується для статичних веб-сторінок;
- Через несумісність старших браузерів з новими функціями поведінка браузера може бути не передбачувана;

2.2.3 SCSS

SCSS (Sassy Cascading Style Sheets) – призначений для спрощення створення CSS-коду. SCSS дає можливість розширити CSS новими можливостями такими як створення змінних, вкладеність. За рахунок додаткових функцій написання SCSS набагато швидше і легше, ніж написання звичайного CSS.

Для того щоб використовувати SCSS в браузері, його потрібно скомпілювати в CSS, так як більшість браузерів не розуміють SCSS. Він використовує той самий синтаксис що і CSS.

На Рисунку 2.3 зображено приклад синтаксису SCSS.

```scss
#main {
  width: 97%;

  p, div {
    font-size: 2em;

    a {
      font-weight: bold;
    }
  }

  pre {
    font-size: 3em;
  }
}
```

Рисунок 2.3 – Синтаксис SCSS.

Переваги використання SCSS наступні:

- Допомагає писати чистий та менший CSS-код;
- Пропонує вкладеність та важливі функції, такі як маніпуляція кольорами, математичні функції;
- Складається зі змінних, що допомагає повторно використовувати значення;
- Сумісний з усіма версіями CSS;

2.2.4 Мова програмування JavaScript

JavaScript – це мова програмування, яка дозволяє впроваджувати складні функції на веб-сторінках, створює динамічно оновлюваний вміст, дозволяє керувати мультимедіа, анімувати зображення.

JavaScript – мова програмування яку інтерпретує веб-браузер з вихідного коду у текстовий вигляд. Однак, більшість сучасних інтерпретаторів використовує своєчасну компіляцію, щоб поліпшити продуктивність. Під час виконання сценарію JavaScript компілюється в швидкий двійковий формат, щоб забезпечити швидке виконання коду. Це дозволяє отримувати високу продуктивність, але JavaScript все ще рахується інтерпретованою мовою програмування, тому що компіляція відбувається під час виконання.

На рисунку 2.2 зображено приклад коду JavaScript

```
export const getDataBooks = (data) => {
  let result = data.map((element) => {
    return {
      img: element.volumeInfo.imageLinks,
      id: element.id,
      title: element.volumeInfo.title,
      authors: element.volumeInfo.authors,
      categories: element.volumeInfo.categories,
    };
  });
  return result;
};
```

Рисунок 2.4 – фрагмент коду JavaScript

Переваги JavaScript:

- Синтаксис мови досить легкий, що дає можливість швидко її вивчити.
- Покращення інтерфейсу користувача за рахунок використання JavaScript.
- JavaScript може виконувати паралельно кілька різних наборів інструкцій.

2.2.5 Бібліотека React

React – це JavaScript бібліотека з відкритим кодом, для розробки інтерфейсів користувача.

Бібліотека базується на компонентному підході, що дозволяє створювати незалежні компоненти, які легко можна перевикористовувати. Це дозволить створити компоненти для керування різними аспектами додатку, такі як

відображення книжок, список вподобань. На рисунку 2.3 зображено приклад компоненту React.

```
import { Outlet } from "react-router-dom";
import { Header } from "../Header/Header";
export const Layot = () => {
  return (
    <main>
      <Header />
      <Outlet></Outlet>
    </main>
  );
};
```

Рисунок 2.5 – Приклад компоненту React.

React використовує віртуальний DOM, що дозволяє оптимізувати оновлення відображення даних на сторінці. Віртуальний DOM порівнює попередні стани компонента і оновлює лише елементи в реальному DOM, які були змінені. Це особливо важливо для веб-додатків для керування процесом читання, де зміни в даних можуть бути частими, при додаванні нових книжок, оновлення стану читання тощо.

Бібліотека React має багато розширень, які можна використовувати для повноцінних програм інтерфейсу користувача. React можна використовувати з різними бібліотеками та фреймворками, такими як Redux, MUI, React Router. Це дозволяє використовувати вже існуючі рішення та екосистему, що спрощує розробку веб-додатку та забезпечує більшу функціональність та гнучкість.

Дані в React передаються тільки в одному напрямку, тобто дані передаються зверху вниз, від батьківських компонентів до дочірніх. Це спрощує керування станом додатку та допомагає уникати непередбачених змін в даних. Така можливість допоможу відслідковувати стан книги та інші взаємодії користувача з додатком.

2.2.6 API Google Book

API Google Book – дозволяє використовувати функції Google книги у вашому веб-сайті або додатку. Він надає вам доступ до багатьох функцій, а саме: пошук книг відповідно до заданого запиту, переглядати інформацію про книгу, можливість керувати книжковими поличками. Також за допомогою Embedded Viewer API є можливість додати попередні перегляд книги.

Переваги використання API Google Book:

- Надає доступ до великої кількості книжок, журналів та інших літературних джерел, що дозволяє отримувати контент для додатку
- Дозоляє здійснювати розширений пошук за різними критеріями, такі як заголовок, автор, жанри та ключовими словами
- Підтримує різноманітність медіа-форматів, таких як текст, зображення, аудіо та відео
- Є можливість отримувати деталі про книгу, отримувати список рекомендованої літератури

2.2.7 EpubJS

EpubJS – це бібліотека JavaScript яка дозволяє відтворювати документи з розширенням epub на будь-якій веб-сторінці з сучасним браузером. Вона містить гнучкий механізм візуалізації та має простий інтерфейс для функцій електронних книг, наприклад розбиття на сторінки, стиль і збереження без спеціального плагіна чи програми.

Бібліотека надає можливість використовувати два різні методи візуалізації: стандартний та безпереривний. За замовчуванням використовується стандартний режим, що передбачає відображення лише одного розділу на екрані. Але можна також використовувати безперервний режим, який показує декілька розділів одночасно, щоб заповнювати екран та попередньо завантажити розділ поза екраном. Цей режим дозволяє плавно перегортати сторінки на різних пристроях.

Також EpubJs реалізує події, до яких легко підключитися. Це дозволяє маніпулювати змістом книги та взаємодіяти з ним. Також бібліотека включає в себе наступні функції:

- Дозволяє шукати певний текст у книзі
- Дозволяє завантажувати книгу з файлу
- За допомогою цієї бібліотеки користувачі можуть легко гортати сторінки на сенсорних екранах

2.2.8 Material UI

Material UI – це набір інструментів інтерфейсу користувача, які можна використовувати для розробки програми на React. Крім того, вона постачається з вбудованою системою оформлення тем, що дозволяє створювати спеціальні теми для додатку. Використання цієї бібліотеки значно прискорює процес розробки React додатків, оскільки компоненти, які імпортуються в проект, є перевіреними та готовими до використання. Крім того, це сприяє швидкому тестуванню зручності використання інтерфейсу користувача.

Одна з переваг Material UI, це її взаємодія з різними системами стилю. Це дає можливість використовувати бібліотеку з різними CSS фреймворками або з ванільним CSS.

Бібліотека добре задокументована та досить проста у використані, це робить її чудовим вибором для розробників будь-якого рівня досвіду. До бібліотеки постійно додається новий функціонал, що робить її найкращим вибором для розробників React на сьогоднішній день.

2.2.9 Firebase

Firebase – це платформа розробки мобільних для створення мобільних та веб-додатків, яка працює на основі моделі Backend як послуга (BaaS). Ця платформа надає широкий спектр інструментів для розробки, в тому числі управління програмами.

Realtime Database – це основна функція Firebase, яка є хмарною базою даних типу NoSQL і працює в режимі реального часу. Дані зберігаються у форматі JSON і можуть бути оновленні та синхронізовані автоматично. База даних працює як веб-середовище, так і в автономному режимі, використовуючи локальний кеш на пристроях для зберігання змін.

Firebase також надає можливість інтеграції з Google Analytics, що дозволяє отримувати дані про поведінку користувачів. Ці дані є корисними для збільшення залученості користувачів до програмного забезпечення.

Також у Firebase присутній модуль Firebase Cloud Storage, який надає користувачам програми можливість зберігати свої файли в хмарі. Хмара має корисні функції, такі як автоматичне призупинення та відновлення завантаження файлів.

Одна з великих переваг Firebase є його швидкі та безпечні послуги хостингу. Крім того, постачальник хостингу забезпечує високу безпеку вмісту за допомогою SSL без конфігурації. SSL запобігає витоку даних і захищає домен від зовнішніх атак.

2.3 Висновки до другого розділу

У другому розділі було розглянуто середовище розробки системи. Також було розглянуто технології які слід використовувати для розробки веб-додатку, мову програмування та її бібліотеку, інтерфейсну бібліотеку та безсервесні обчислення.

Для реалізації веб-додатку для керування процесом читання будуть застосовуватись такі технології:

- Мова програмування JavaScript
- Мову розмітки HTML
- Перепроцесор SCSS
- Бібліотека React
- Для реалізації функції електронної книги буде застосовуватись бібліотека EpubJs
- Для отримання даних про книги буде використовуватись API Google Book

- Для інтерфейсу буде використана бібліотека Material UI
- Дані користувачів будуть зберігатись в Firebase Realtime database

3. РОЗРОБКА СТРУКТУРИ WEB-ДОДАТКУ ДЛЯ КЕРУВАННЯ ПРОЦЕСОМ ЧИТАННЯ З ВИКОРИСТАННЯМ БІБЛІОТЕКИ REACT

3.1 Завдання web-додатку для керування процесом читання

Відслідковувати свої книги досить важливо адже більшість людей швидко переглядають книги, та часто забувають подробиці сюжету, героїв протягом кількох місяців. Це приводить до того що купуючи нову книгу або беручи в бібліотеці та прочитавши близько 10 сторінок, може виявитись що таку книгу людина вже читали. Додаток для керування процесом читання допоможе вирішити цю проблему.

Такий додаток допомагає покращити розуміння прочитаного за рахунок написання резюме на книгу або короткого огляду. Також використовуючи додаток для керування процесом читання для відстеження своїх книг допоможе краще зосереджуватись та концентруватись на книзі. Роблячи нотатки або пишучи короткий виклад прочитаного покращить навички конспектування. Додаток показує як змінювались та розвивались вподобання щодо жанрів книг.

Основна перевага таких додатків це організація книг користувача, допомагають ефективніше використовувати свій час та підвищують ефективність навчання і розвитку.

Розроблений додаток дозволить виконувати такі задачі, пов’язані з керуванням процесом читання:

- Пошук книг за назвою або жанром;
- Додавання книг на полички;
- Нотувати моменти з книги
- Написання резюме на книгу або короткий огляд
- Завантаження книг у додаток для подальшого читання
- Виділення важливих моментів у книзі

Мета роботи – розробити web-додаток для керування процесом читання з використанням бібліотеки React.

3.2 Моделювання об’єкту проектування

3.2.1 Діаграма прецедентів системи

Діаграма прецедентів – це UML діаграма, яка зображає відношення між актором та прецедентом в системі.

Суть цієї діаграми полягає в тому, що система яка проектується подається у вигляді множин сутностей та акторів, які взаємодіють з системою за допомогою варіантів використання. Кожен варіант використання описує набір дій, що виконує система під час діалогу з користувачем.

Основні елементи діаграми прецедентів:

- Актор – сутність яка взаємодіє з системою ззовні. Стандартним графічним позначенням актора є фігурка «чоловічка», під якою вказується ім’я актора. Актор може бути не тільки людина, але і будь-яка інша система
- Прецедент - набір послідовних дій, що виконуються системою, для того щоб актор отримав певний результат. Стандартним графічним позначенням прецеденту є еліпс, всередині якого міститься назва прецеденту

Між актором та прецедентом існують різні відносини, які описують взаємодію між ними. Є кілька видів таких відносин між актором та прецедентом:

- Ставлення асоціації (association) – визначає зв'язок між актором та прецедентом. Позначається суцільною лінією
- Ставлення розширення (extend) – визначає взаємозв’язок одного прецеденту з іншим більш загальним. Зазвичай позначається пунктирною лінією, рух напрямку від прецеденту який є розширенням до вихідного прецеденту, та позначається ключовим словом «extend»
- Ставлення включення (include) – вказує, на те що поведінка одного прецеденту є складовою поведінки іншого прецеденту. Позначається

пунктирною лінією, рух напрямку від базового прецеденту до того що включається, підписується ключовим словом «include»

Веб-додаток для керування процесом читання має одного актора який взаємодіє з системою. Для користуванням повним функціоналом додатку користувачеві потрібно авторизуватися. Без авторизації буде доступний тільки пошук книг за назвою або жанром. Після успішної авторизації стануть доступні методи додавання книг на полички користувача, перегляд поличок з книгами, додавання цитат з книг, написання відгуку на книгу, можливість змінювати статус книг на «читаю» або «прочитано» та завантаження з пристрою користувача книги у додаток для читання. На рисунку 3.1 зображена діаграма прецедентів системи.

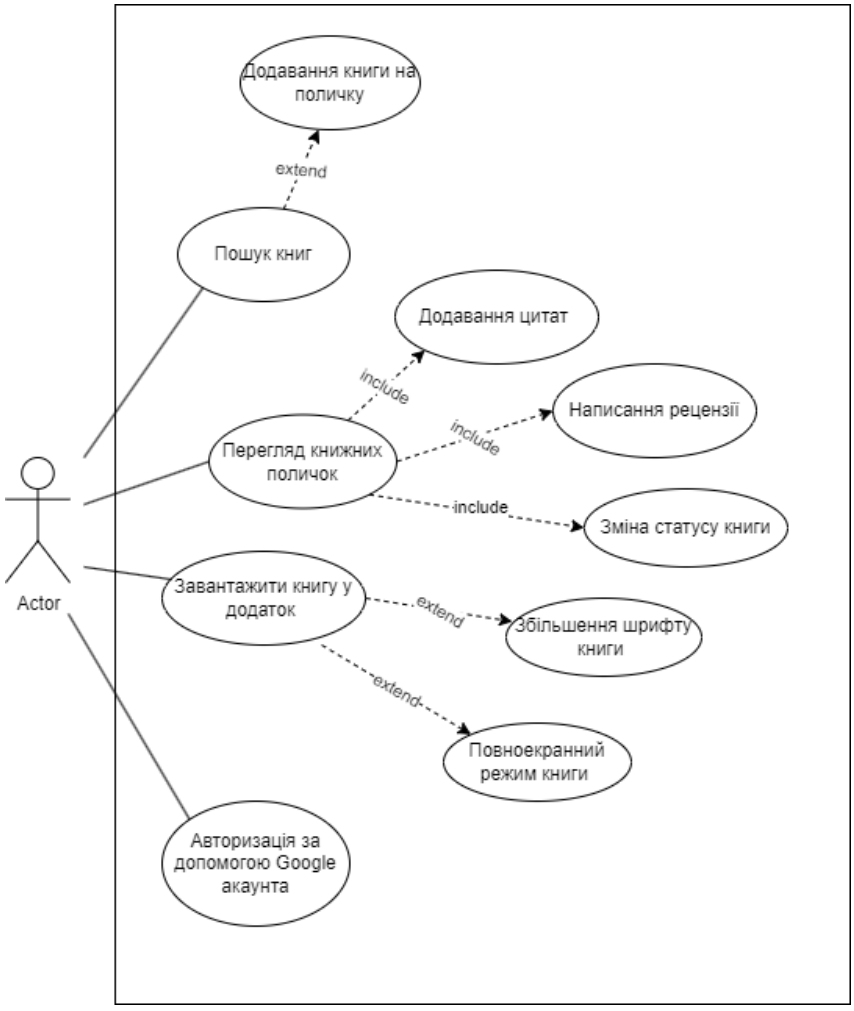


Рисунок 3.1 – UML діаграма прецедентів системи.

3.2.2 Діаграма діяльності

Діаграма діяльності – це UML діаграма яка показує дії що виконує система. Основні складові діаграми діяльності:

- Вузли: етапи на якому користувач або система виконує певну дію, позначається прямокутником з заокругленими кутами;
- Розвилки: розгалуження вибору, позначається ромбом;
- Початок та кінець процесу: показують початок та кінець процесу, позначаються у вигляді кружечків;
- Потоки керування: з’єднують вузли та показують порядок виконання дій, позначаються у вигляді лінії зі стрілками та вказують напрямок;

За допомогою діаграми діяльності можна продемонструвати логіку алгоритму, спростить процес розуміння як працює система та прояснить складні випадки використання, можна використовувати для модулювання елементів архітектури програми.

На Рисунку 3.2 зображено UML діаграма діяльності додатку.

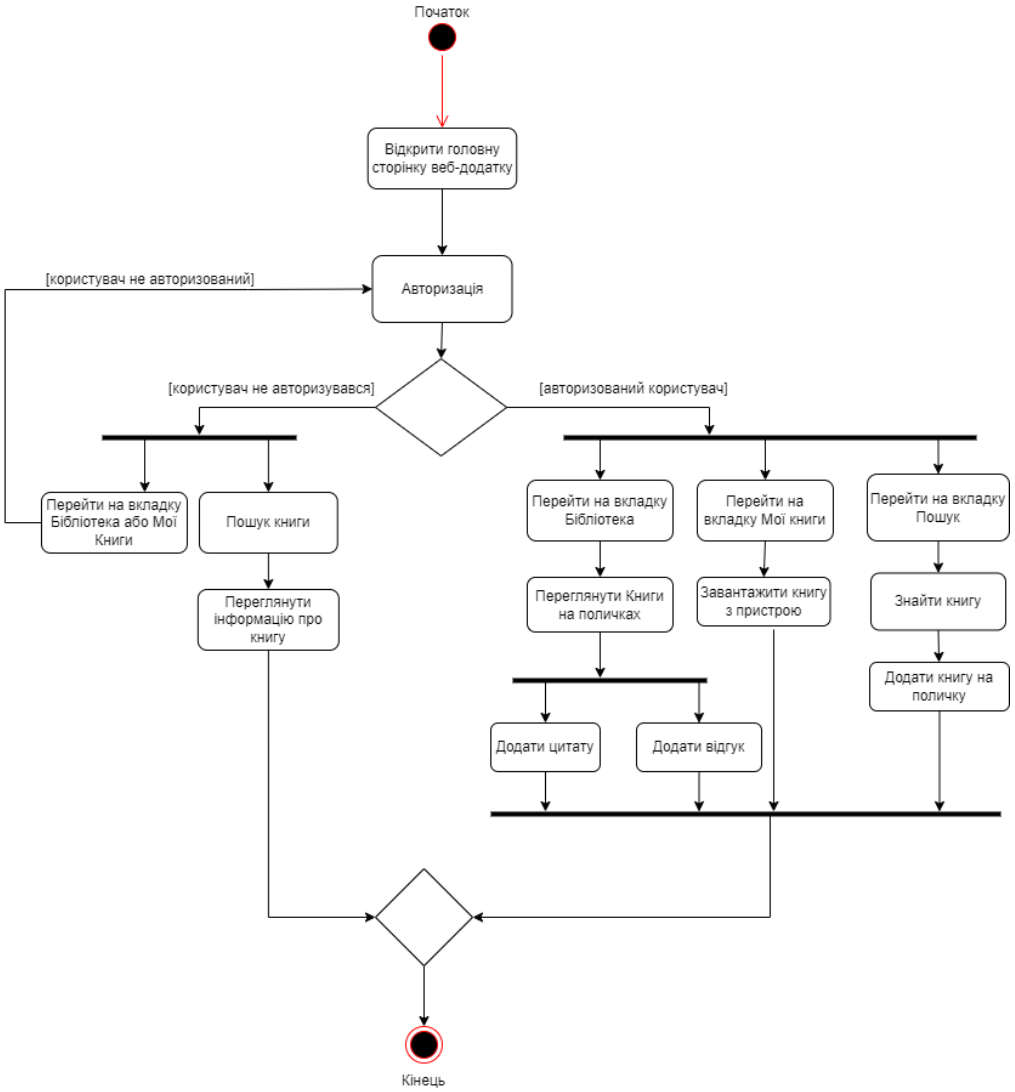


Рисунок 3.2 –UML діаграма діяльності.

3.2.3 Структура даних у додатку

У веб-додатку використовується Realtime database від платформи Firebase. База даних Firebase Realtime являється базою даних NoSQL, це надає можливість синхронізувати та зберігати дані між користувачами в режимі реального часу. Структура бази даних зображена на Рисунку 3.2.

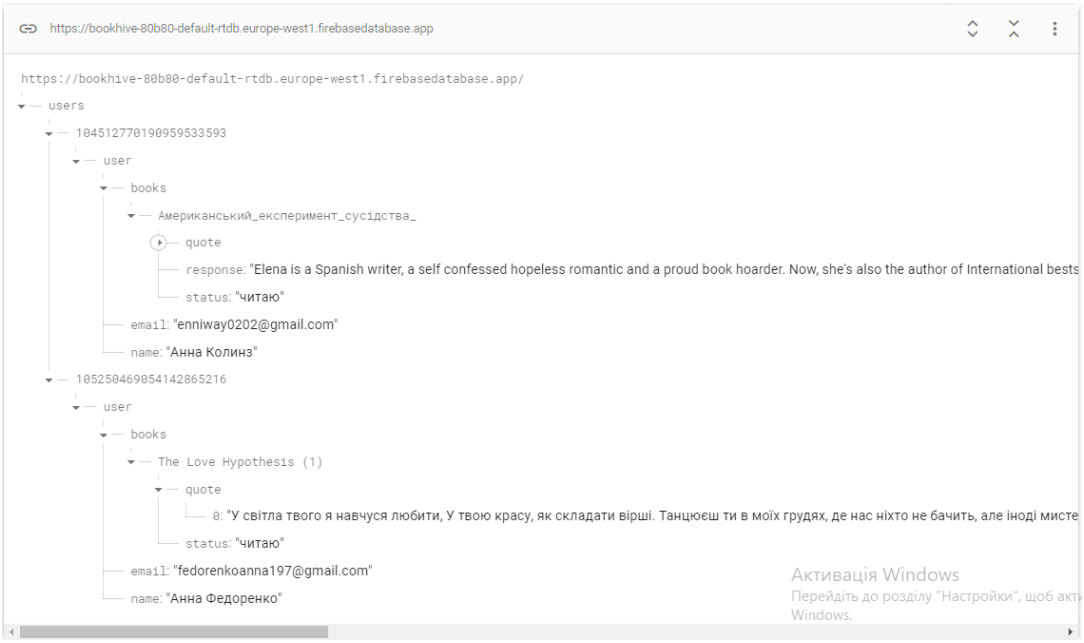


Рисунок 3.2– Структура бази даних.

Нижче наведені об’єкти є основними сутностями структури.

User – головний об’єкт (Рис.3.2.3) містить в собі дані користувача та об’єкт з масивом книг.

Поле id – має тип number та містить унікальний ідентифікатор користувача.

Поле books – містить в собі масив об’єктів Book.

Поле name – має тип string, та містить в собі ім’я користувача.

Поле email – має тип string, та містить в собі електрону пошту користувача.

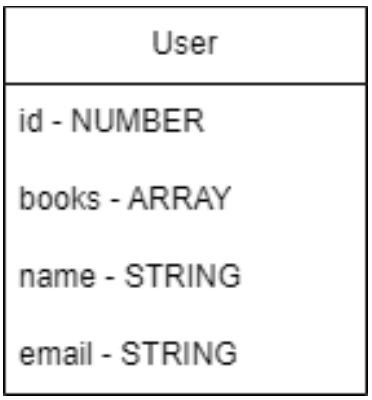


Рисунок 3.3 – Структура об’єкту User .

Book – об’єкт із книгою користувача (Рис 3.2.4), знаходиться в масиві об’єктів Books які належать об’єкту User.

Поле title – має тип string, містить назву книги.

Поле quote – має тип array, містить масив цитат з книг користувача.

Поле review – має тип string, містить відгук на книгу або короткий опис.

Поле status – має тип string, містить статус книги.

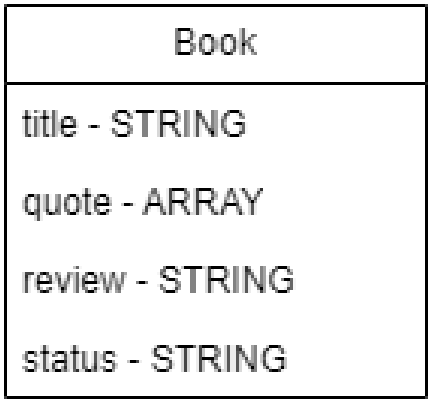


Рисунок 3.4 – Структура об’єкту Book .

3.3 Структура системи

3.3.1 Структура клієнтської частини

Клієнтська частина додатку розроблена з використанням бібліотеки React, яка базується на компонентному підході. Це наддасть змогу створювати компоненти які можна легко перевикористовувати. Бібліотека React має велику кількість інтерфейсних бібліотек які мають вже готові компоненти. Для побудови UI була обрана одна з таких бібліотек, а саме MUI.

Структура клієнтської частини має 5 основних класи що зображенні на Рисунку 3.3.1.

Клас User зберігає в собі інформацію про користувача та масив об’єктів Shelf, при вході користувача у додаток викликається метод useGoogle.

Клас Shelf зберігає в собі ідентифікатор книжної полички, назва полички, поле books містить в собі масив об’єктів Book які знаходяться на поличці. Поле displayBook містить об’єкт DisplayBook який відповідає за книгу з книжної полички користувача яка перша відображається на сторінці. Клас містить методи для роботи з книжними поличками користувача.

Клас Book містить в собі інформацію про книгу заголовок, автора, зображення обкладинки, ідентифікатор книги, опис та кількість сторінок. За допомогою методу addVolume книга додається на книжну поличку.

Клас Books містить в собі масив об’єктів Book. Метод getBook здійснює пошук книг за ключовим словом або жанром. При натисканні на книгу спрацьовує метод setOpen та відображається детальна інформація про обрану книгу.

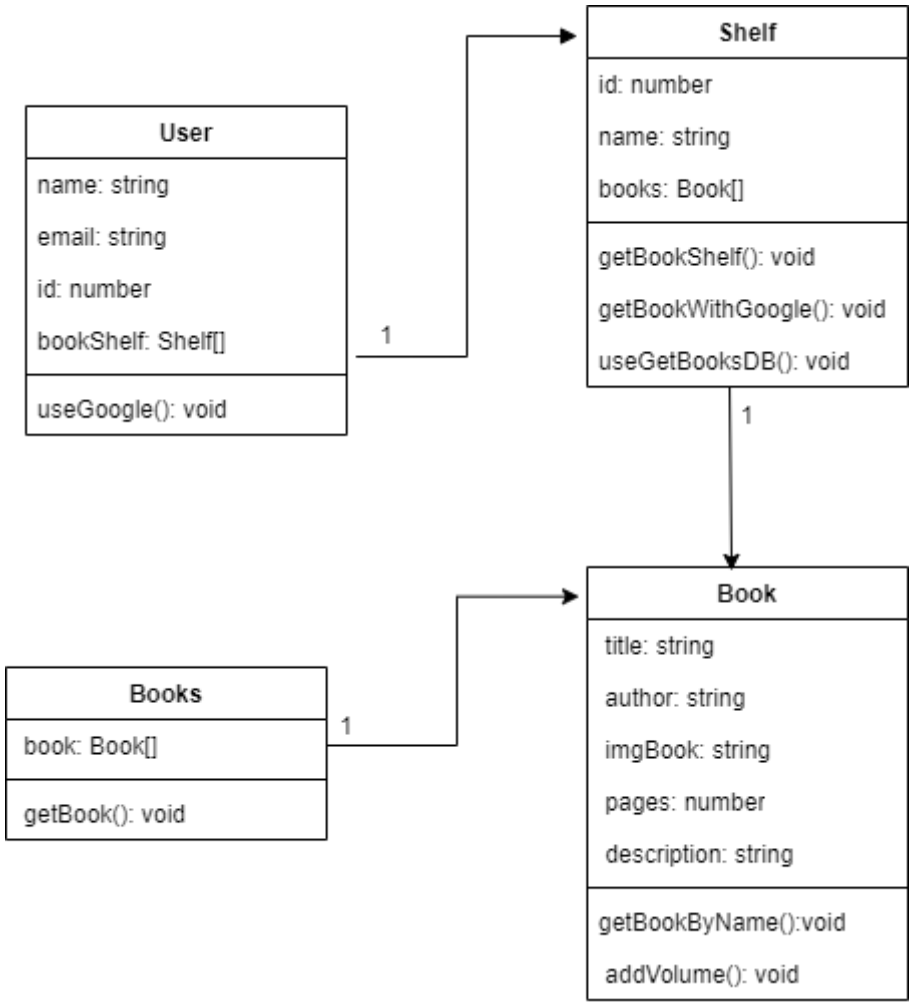


Рисунок 3.5– Структура клієнтської частини додатку.

3.3.2 Серверна структура додатку

Роль сервера у додатку виконує Firebase, він надає доступ до модуля realtime database.За кожен об’єкт у таблиці відповідає свій роутер.

Авторизація реалізована за допомогою npm-пакету @react-oauth/google який використовує SDK Google Identity Service для входу в google акаунт користувача.

Після авторизації та отримання ідентифікатора користувача, надаються права на редагування даних.

3.4 Висновки до третього розділу

В третьому розділі було розглянуто структуру web-додатку для керування процесом читання. Визначено завдання системи що розробляється, а саме система повинна мати:

- Зручний та зрозумілий інтерфейс;
- Можливість пошуку книг за жанром або назвою;
- Можливість нотувати цитати з книги;
- Можливість залишати відгук на книгу або писати короткий опис книги;
- Мати функцію електронної книги;

Створено діаграму прецедентів, яка показує взаємодію системи з користувачем. Розглянуто структуру даних у додатку, структуру клієнтської частини та її класи, структуру серверної частини.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ДОДАТКУ

4.1 Оцінка та планування розробки проекту

Під час розробки програмного забезпечення перш за все визначають задачі та визначають функціонал що потрібно розробити. Щоб кожна людина, яка приймає участь у створенні програмного забезпечення, могла бачити які завдання потрібно виконати та які вже зроблені, використовують сайти або додатки для керування проектами в режимі реального часу. Одним з таких додатків є Trello.

Trello – це інструмент для управління проектами, який допомагає організовувати свою роботу та підвищувати продуктивність.

Головними перевагами Trello є:

- Він дозволяє легко організовувати проекти та завдання;
- Trello має інтеграції з багатьма іншими інструментами, наприклад такими як Google Drive, Slack, Zapier;
- Дає змогу встановлювати терміни для виконання завдань та переглядати прогрес;
- Дозволяє визначати пріоритети завдань;
- Має простий інтуїтивний інтерфейс;

Для організації завдань було створено три дошки «Потрібно зробити», «В процесі» та «Готово».

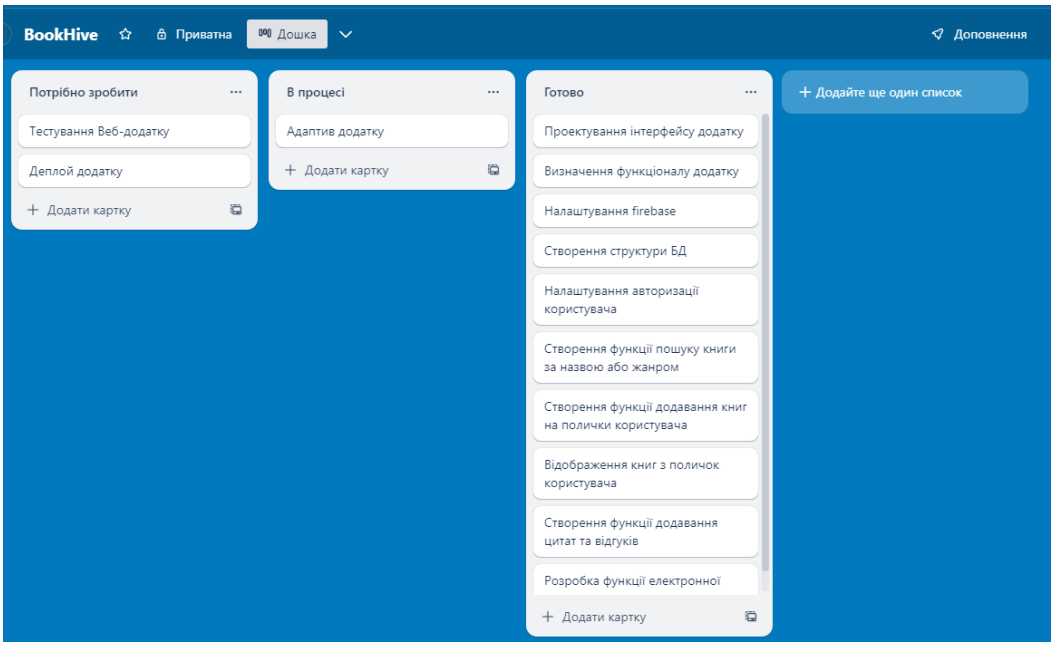


Рисунок 4.1 – Задачі для створення додатку на сайті Trello.

Як можна побачити на Рис 4.1 більшість завдань для створення веб-додатку вже виконано. Всі задачі виконує одна людина яка є і менеджером проекту, дизайнером та програмістом.

4.2 Функціонал клієнтської частини

Клієнтська частина була виконана з використанням мови програмування JavaScript та її бібліотеки React.

Зв'язок клієнтської частини з базою даних відбувається за допомогою HTTP запитів. Інтерфейс додатку мінімалістичний та інтуїтивно зрозумілий.

Для навігації між сторінками веб-додатку був використаний React Router, це дозволить змінювати відображення контенту на сторінці без повної перезавантаження сторінки. На Рисунку 4.2 зображено маршрути веб-додатку.

```
export const router = createBrowserRouter([
  {
    path: "/",
    element: <Layot />,
    children: [
      {
        index: true,
        element: <MainPage />,
      },
      {
        path: "/home",
        element: <Home />,
        children: [
          {
            index: true,
            element: <ReadingNow />,
          },
          {
            path: "/home/:id/:title",
            element: <GoogleBookShelf />,
            loader: getBookWithGoogle,
          },
        ],
      },
      {
        path: "/my-book",
        element: <Books />,
      },
    ],
  },
]);
```

Рисунок 4.2 – Маршрути веб-додатку.

Не авторизований користувач може тільки здійснювати пошук книг за назвою або жанром. Для того щоб користуватись повним функціоналом додатку, потрібно авторизуватись за допомогою акаунта Google.

Так як для використання Google API Books потрібен ідентифікатор користувача та маркер доступу для отримання інформації про книжні полички користувача та мати змогу їх редагувати, був обраний спосіб авторизації за допомогою Google API. Авторизація реалізована за допомогою npm-пакета @react-

oauth/google який використовує SDK Google Identity Service для входу в google акаунт. Фрагмент коду який відповідає за авторизацію зображений на Рисунку 4.2.

```
export const useGoogle = () => {
  const [user, setUser] = useState(null);
  const googleLogin = useGoogleLogin({
    onSuccess: async (tokenResponse) => {
      const userInfo = await axios
        .get("https://www.googleapis.com/oauth2/v3/userinfo", {
          headers: { Authorization: `Bearer ${tokenResponse.access_token}` },
        })
        .then((res) => res.data);
      setUser({
        name: userInfo.name,
        userImg: userInfo.picture,
        token: tokenResponse.access_token,
        id: userInfo.sub,
      });
      addUser(userInfo.sub, userInfo.name, userInfo.email)
    },
    scope: "https://www.googleapis.com/auth/books",
  });
  useEffect(() => {
    if (user) {
      const dataJson = JSON.stringify(user);
      localStorage.setItem("user", dataJson);
    }
  }, [user]);
  useEffect(() => {
    if (localStorage.getItem("user")) {
      setUser(JSON.parse(localStorage.getItem("user")));
    }
  }, []);
  return [user, setUser, googleLogin];
};
```

Рисунок 4.3 – Фрагмент коду авторизації.

На Рисунку 4.3 зображено екран згоди, яку надає користувач для веб-додатку.

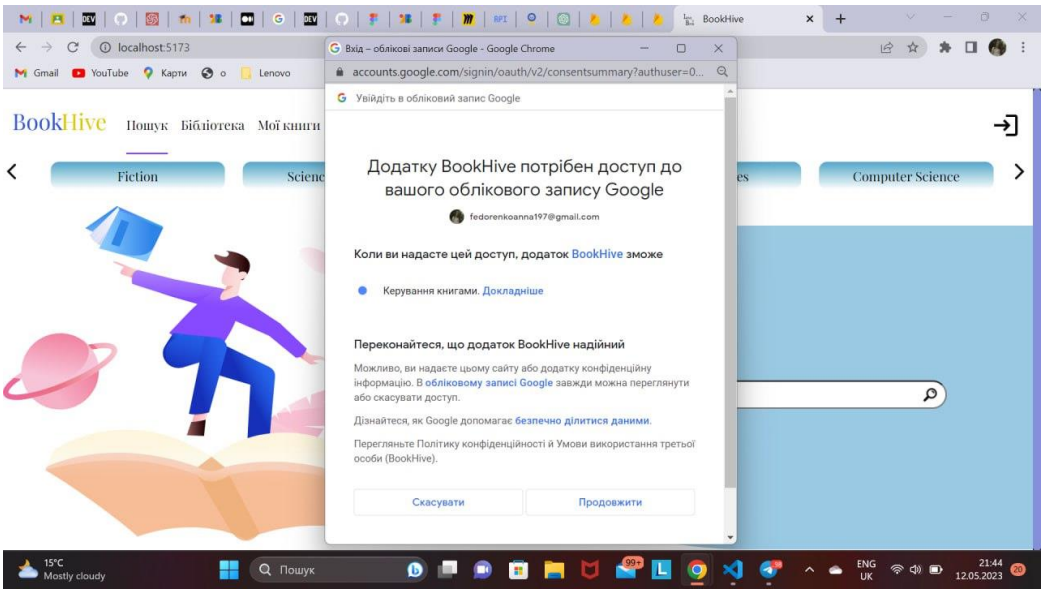


Рисунок 4.4 – Екран згоди.

Після авторизації на вкладці «Пошук» користувач може здійснити пошук книги, після введення в полі пошуку назви книги або ім’я автора та натискання на кнопку Enter надсилається запит на Google API Book та на сторінці відображається результат запиту. Також пошук можна здійснити за жанрами, які знаходяться зверху у вигляді каруселі. При натисканні на жанр здійснюється пошук та на сторінці відображається результат. Книги відображаються у вигляді карток на яких зображено обкладинку книги, назва, автор та жанр. Якщо натиснути на автора то здійсниться пошук книг цього автора. На рисунку 4.4 зображено результату пошуку книги «Чорна Рада».

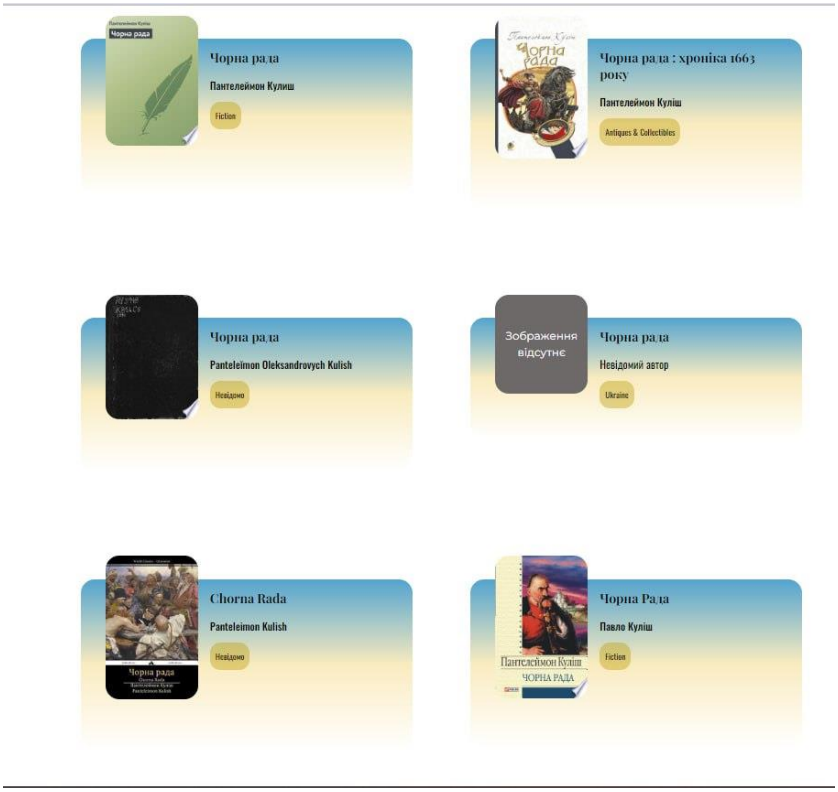


Рисунок 4.5 – Результат пошуку книги «Чорна Рада».

При натисканні на обкладинку книги відкривається модальне вікно на якому відображається зображення книги, опис, кількість сторінок та кнопка «Додати книгу», яка відповідає за додавання обраної книги на книжні полички. На Рисунку 4.5 зображено приклад модального вікна.

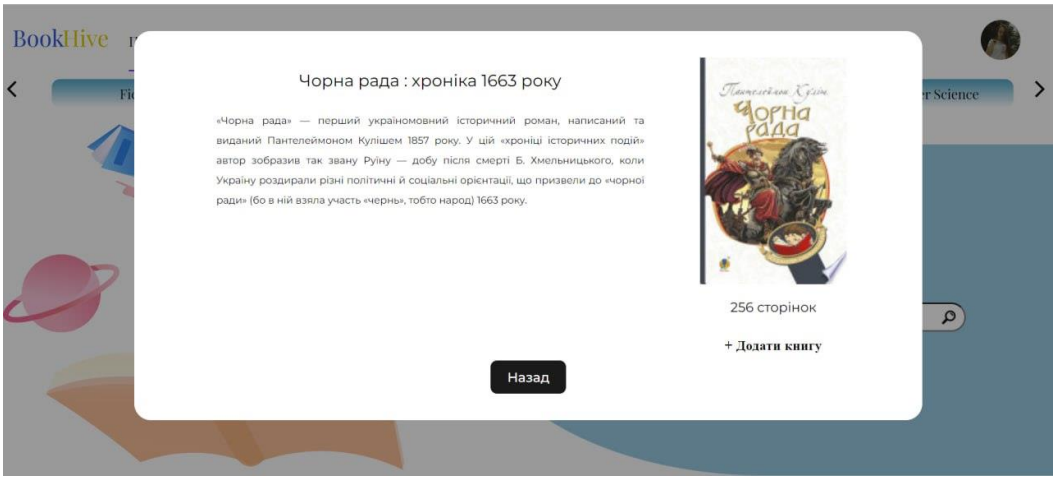


Рисунок 4.6 – Модальне вікно.

При переході на вкладку «Бібліотека» відображаються всі книжкові полички користувача. При переході на книжкові полички відображаються книги які там знаходяться. На Рисунку 4.6 зображено поличка «To read».

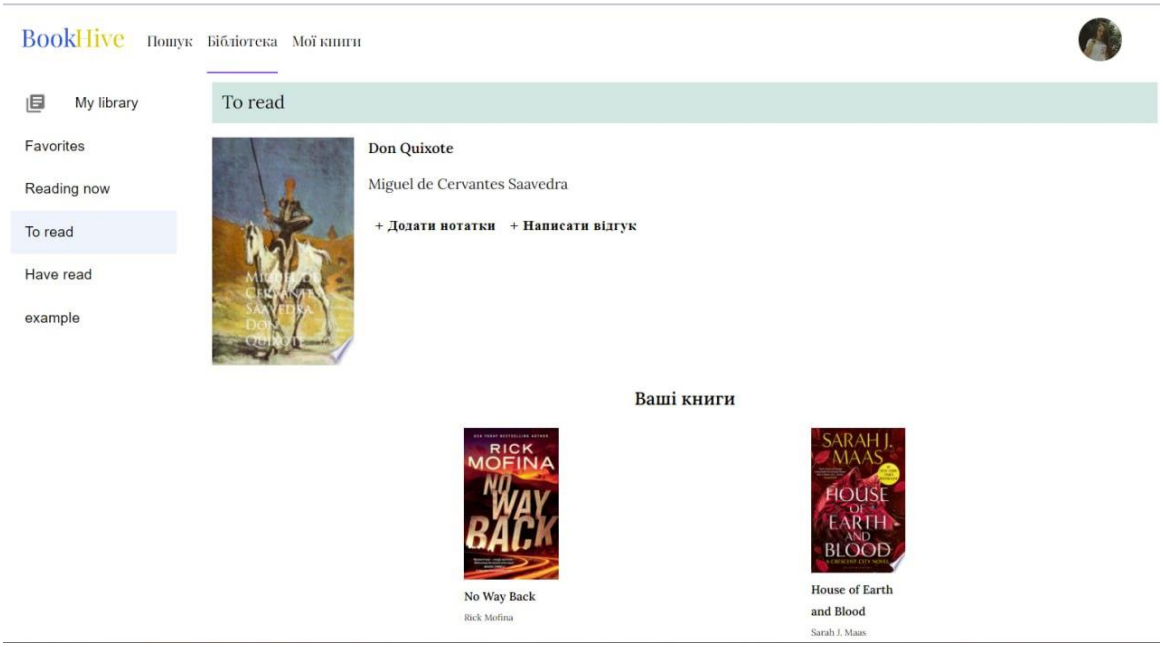


Рисунок 4.7 – Поличка «To read».

Біля книги є дві кнопки які відповідають за додавання цитати та написання відгука. На Рисунку 4.7 зображено приклад інтерфейсу книги з відгуком та цитатою.

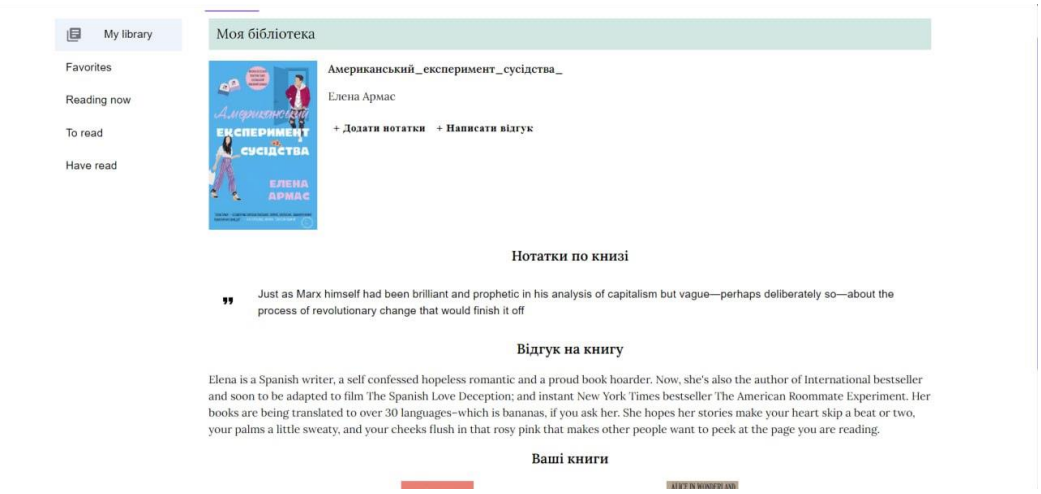


Рисунок 4.8 – приклад інтерфейсу книги з відгуком та цитатою.

На вкладці «Мої Книги» користувач має можливість завантажити книгу з пристрою, для того щоб прочитати її. На Рисунку 4.8 зображено приклад інтерфейсу вкладки «Мої Книги».

Рисунок 4.9 – Приклад інтерфейсу вкладки «Мої Книги».

При натисканні на кнопку Завантажити відкривається Файловий провідник та користувач може завантажити файл з розширенням epub. Після чого книга відкривається у додатку. На Рисунку 4.8 зображено приклад інтерфейсу книги користувача.

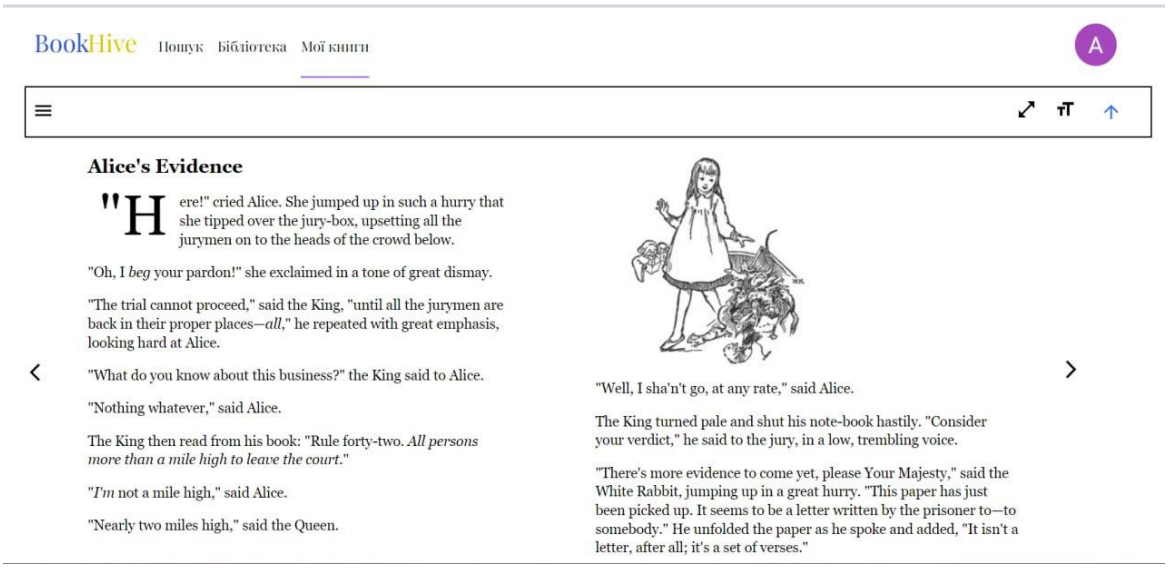


Рисунок 4.10 – Приклад інтерфейсу книги користувача.

Користувач має можливість збільшити розмір шрифту, відкрити книгу у повноекранному режимі та використовувати навігацію по книзі, переходячи по розділам. Також виділяти текст в книзі жовтим кольором. При повторному вході у додаток, всі книги які були завантажені у додаток зберігаються та є можливість продовжити читати з моменту на якому користувач зупинився. Всі виділені фрази у книзі також зберігаються.

4.3 Функціонал серверу

Головною задачею сервера є взаємодія з користувачем та базою даних. Після авторизації користувач отримує ідентифікатор. Профіль користувача в базі даних знаходиться за шляхом users/userID/user/books/titleBook, де використовується ідентифікатор отриманий при авторизації.

Для взаємодії з Realtime Database було створено функції які відповідають за додавання юзера у базу даних, додавання відгуків та цитат. На Рисунку 4.9 зображено приклад коду для додавання користувача у базу даних.

```
export const addUser = (userId,name,email) =>{
  const userRef = ref(database,`users/${userId}/user`)
  get(userRef)
  .then((snapshot) => {
    if (snapshot.exists()) {
      return
    } else {
      set(userRef, {
        name: name,
        email: email
      });
    }
  })
  .catch((error) => {
    console.log(error);
  });
}
```

Рисунок 4.11 – Функція що відповідає за додавання користувача у базу даних.

За аналогією працюють решта функцій які відповідають за взаємодію з базою даних.

4.4 Тестування

Функціональне тестування – це вид тестування яке спрямоване на перевірку функціональних вимог програмного забезпечення.

На основі діаграми прецедентів було створено такі тест-кейси:

- Тестування авторизації;
- Тестування функції додавання книг на поличку;
- Тестування функції пошуку книг;
- Тестування функції додавання цитат та відгуків на книгу;
- Тестування функції електронної книги;

На рисунку 4.12 зображено приклад тест-кейсів для тестування системи.

Test Case ID	Test Priority	Test Case Title	Step	Test Steps	Test Data	Expected Result	Status
TC-1	Високий	Перевірка авторизації	1	Відкрити додаток	http://localhost:4173/	Додаток успішно відкрився	pass
			2	Натиснути кнопку входу в акаунт		Відкрито вікно входу в акаунт	pass
			3	Обрати акаунт Google	fedorenkoanna1997@gmail.com	Вібувся вхід в акаунт	pass
TC-2	Середній	Тестування функції пошуку книг	1	Перейти на вкладку Пошук		Перехід на вкладку Пошук	pass
			2	Ввести назву книги в полі пошуку	Лісова пісня	Назва книги відображається в полі пошуку	pass
			3	Натиснути Enter	Enter	Результат пошуку відобразився на сторінці	pass
TC-3	Середній	Тестування функції додавання книг на поличку	1	Натиснути на обкладенку книги Лісова пісня		Відкрилось модальне вікно з інформацією про книгу	pass
			2	Натиснути на кнопку Додати книгу		Відкрився список книжних поличок	pass
TC-4	Середній	Тестування функції додавання цитат	3	Обрати поличку	Улюблене	Повідомлення про успішно додану книгу	pass
			1	Перейти на вкладку Бібліотека		Успішний перехід на вкладку Бібліотека	pass
			2	Обрати поличку на якій знаходяться книги	Улблене	Відображаються книги з полички Улюблене	pass
			3	Натиснути кнопку Додати нотатки		Відкрилось текстове поле	pass
			4	Ввести цитату з книги	для тебе щастя - тінь, ти нежива.	Повідомлення про успішно додану цитату	pass
TC-5	Середній	Тестування функції електроної книги	1	Перейти на вкладку Мої книги		Вкладка Мої книги успішно відкрилась	pass
			2	Натиснути на кнопку Завантажити		Відкрився файловий провідник	pass
			3	Обрати файл з розширенням epub	alice.epub	Книга відобразилась на сторінці	pass

Рисунок 4.12 – Приклад тест-кейсів для тестування системи.

Тестування інтерфейсу – це тестування інтерфейсу користувача на графічні та графічно-функціональні помилки.

При тестуванні було виправленні всі помилки інтерфейсу які з’являлись під час адаптації додатку під мобільні пристрої.

На Рисунку 4.13 та 4.14 зображено тестування адаптації додатку.

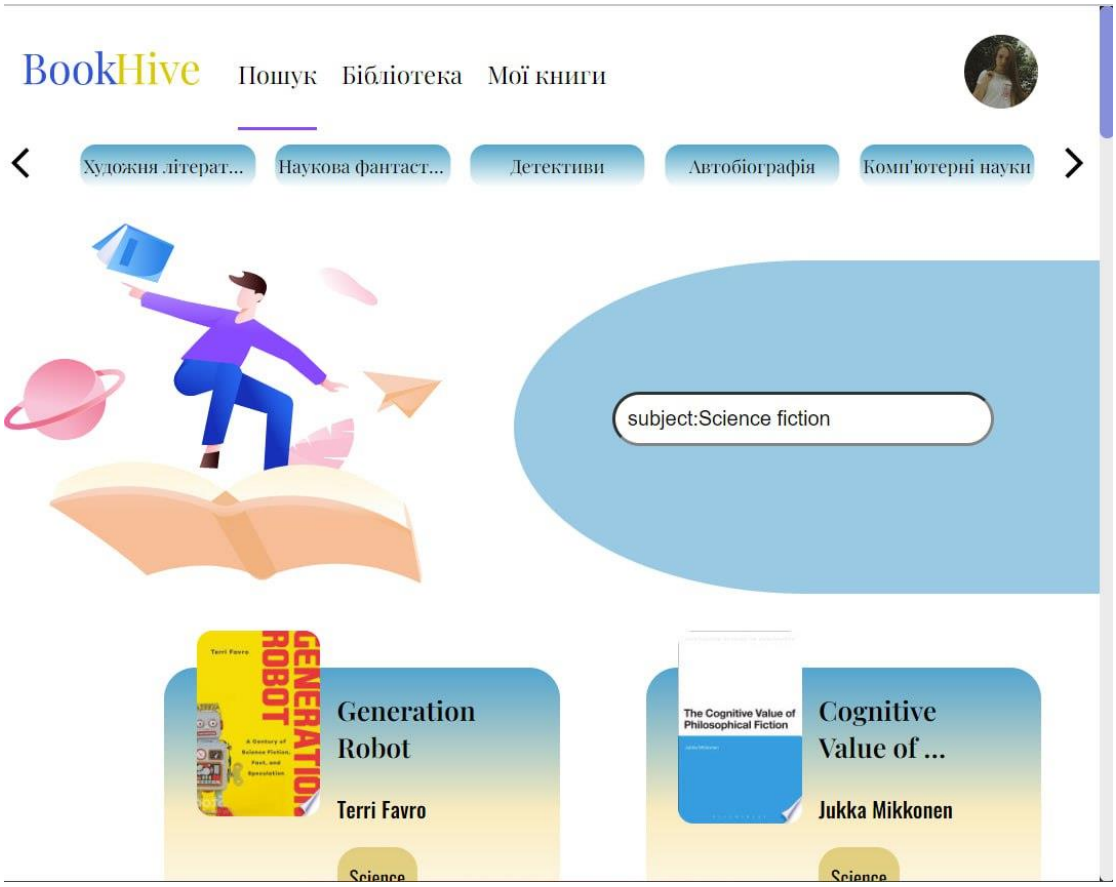


Рисунок 4.13 – Тестування адаптації під планшети.

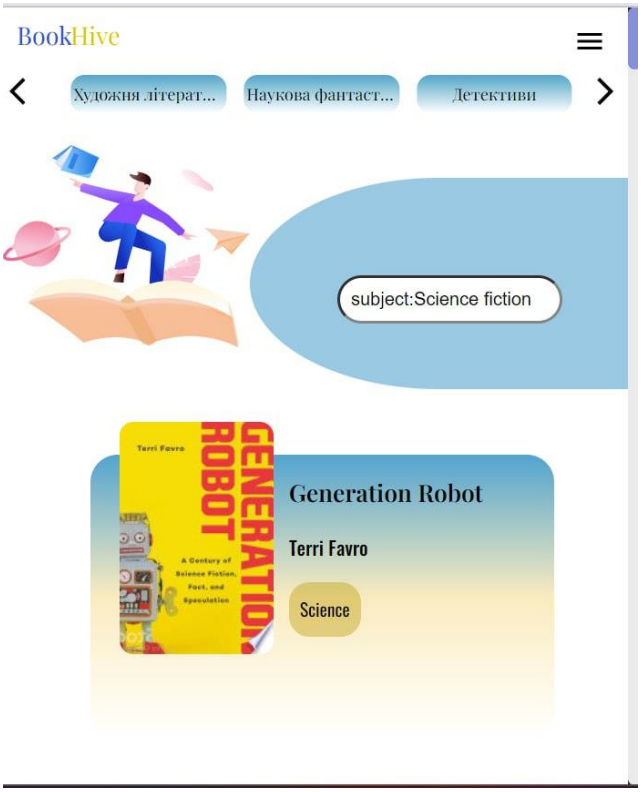


Рисунок 4.13 – Тестування адаптації під мобільні пристрої.

4.5 Висновки до четвертого розділу

В четвертому розділі було розглянуто програмну реалізацію додатка. Розглянуто функціонал додатку.

По завершенню була закінчена розробка веб-додатку та його тестування.

База даних використовується від платформи Firebase, а саме модуль Realtime Database.

Клієнтська частина була розроблена з використанням React, HTML,SCSS.

ВИСНОВКИ

У результаті виконання даної дипломної роботи було спроектовано та розроблено web-додаток, який допоможе керувати процесом читання.

Проаналізовано існуючі аналоги з метою визначення їх недоліків та переваг. Основними недоліками існуючих систем для керування процесом читання є відсутність інтеграції з Google Book та можливість завантажувати книги у додаток. Розглянуто актуальність та визначено потреби користувача при роботі з додатками для керування процесом читання.

Проведено аналіз середовищ розробки та інструментів для реалізації web-додатків.

Було обрано такі програмні засоби для реалізації додатку: React, HTML, SCSS, JavaScript, Firebase, MUI. Visual Studio Code було обрано як середовище розробки.

Виконано проектування додатку: розроблено діаграму прецедентів, діаграму діяльності, діаграму класів.

Розроблено клієнтську частину додатку з використанням бібліотеки React. Використано компонентний підхід бібліотеки, що дає можливість перевикористовувати компоненти. Також використано маршрутизацію бібліотеки, що забезпечує оновлення вмісту сторінки без її перезавантаження.

Серверну частину розроблено на основі сервісу Firebase. Зокрема модуль realtime database використано для збереження даних користувача.

Проведено функціональне тестування та тестування інтерфейсу користувача. При функціональному тестуванні були виправлені дефекти та помилки. При тестуванні інтерфейсу усі елементи інтерфейсу користувача коректно відображаються на різних розширеннях екрану та в різних веб-браузерах.

В ході розробки системи реалізовано такі функції як пошук книг за назвою або жанром, додавання книг на полички, написання цитат та відгуків до книг, функція електронної книги.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Діаграма Прецедентів [Електронний ресурс] – Режим доступу до ресурсу: http://ni.biz.ua/9/9_11/9_115925_diagramma-pretsedentov.html.

2. Розуміння безсерверних обчислень для початківців [Електронний ресурс] –Режим доступу до ресурсу: https://techukraine.net/розуміння-безсерверних-обчислень-дл.

3. Що таке SPA - особливості та приклади односторінкових додатків для бізнесу. [Електронний ресурс] – Режим доступу до ресурсу: https://wezom.com.ua/ua/blog/chto-takoe-spa-prilozheniya.

4. EPUB.js vs readium.js - A detailed comparison of 2 epub readers [Електронний ресурс] – Режим доступу до ресурсу: https://kitaboo.com/epub-js-vs-readium-js-comparison-of-epub-readers/.

5. Goodreads [Електронний ресурс] – Режим доступу до ресурсу: https://www.goodreads.com/.

6. Html [Електронний ресурс] – Режим доступу до ресурсу: https://influencermarketinghub.com/glossary/html/.

7. Is Material-UI the best choice for React in 2023? [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/codex/is-material-ui-the-best-choice-for-react-in-2023-b1b85f92db98.

8. LibraryThing [Електронний ресурс] – Режим доступу до ресурсу: https://www.librarything.com/.

9. Overview | google books apis | google for developers. [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/books/docs/overview.

10. PWA додатки Web Progressive App: як вони працюють і чим корисні для бізнесу [Електронний ресурс] – Режим доступу до ресурсу: https://wezom.com.ua/ua/blog/pwa-prilozheniya-web-progressive-app.

11. Scribd ui [Електронний ресурс] – Режим доступу до ресурсу: https://www.greatworklife.com/wp-content/uploads/2020/01/scribd-review-magazines-ebooks.jpg.

12. Scribd [Електронний ресурс] – Режим доступу до ресурсу: https://www.scribd.com/.

13. SINGLE PAGE APPLICATION (SPA) VS MULTI PAGE APPLICATION (MPA): PROS AND CONS. [Електронний ресурс] – Режим доступу до ресурсу: https://merehead.com/blog/single-page-application-vs-multi-page-application/.

14. SPA vs. MPA: pros, cons & how to make final choice [Електронний ресурс] – Режим доступу до ресурсу: https://www.simicart.com/blog/spa-vs-mpa/.

15. The best guide to know what is react [Електронний ресурс] – Режим доступу до ресурсу: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs.

16. UML activity diagram [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/uml-activity-diagram.

17. What is a web application? [Електронний ресурс] – Режим доступу до ресурсу: https://www.indeed.com/career-advice/career-development/what-is-web-application.

18. What is HTML and how does hypertext markup language work? [Електронний ресурс] – Режим доступу до ресурсу: https://www.theserverside.com/definition/HTML-Hypertext-Markup-Language.

19. What is JavaScript? [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript.

20. What is PWA? Progressive web apps explained [Електронний ресурс] – Режим доступу до ресурсу: https://vuestorefront.io/pwa.

21. What is serverless computing and top 5 benefits [Електронний ресурс] – Режим доступу до ресурсу: https://www.ancoris.com/blog/what-serverless-computing-top-5-benefits.

22. What is the difference between CSS and SCSS? [Електронний ресурс] – Режим доступу до ресурсу: https://www.scaler.com/topics/difference-between-css-and-scss/.

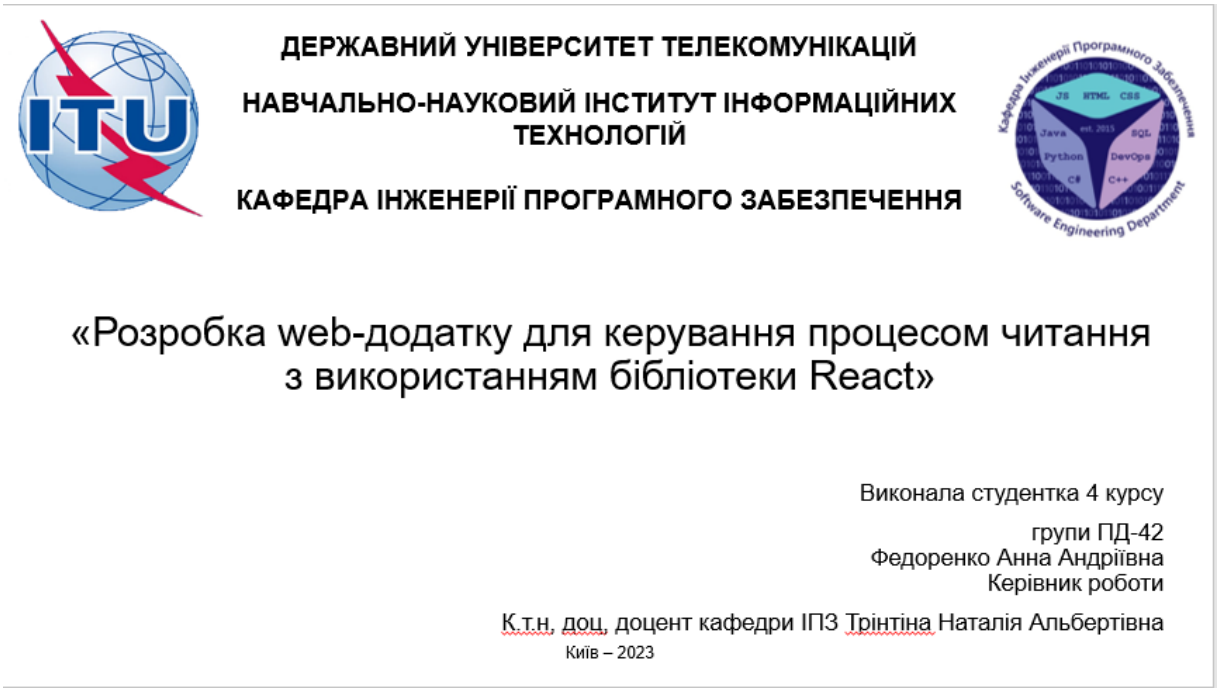
23. What is trello? Uses and benefits [Електронний ресурс] – Режим доступу до ресурсу: https://intellipaat.com/blog/what-is-trello/?US#no6.

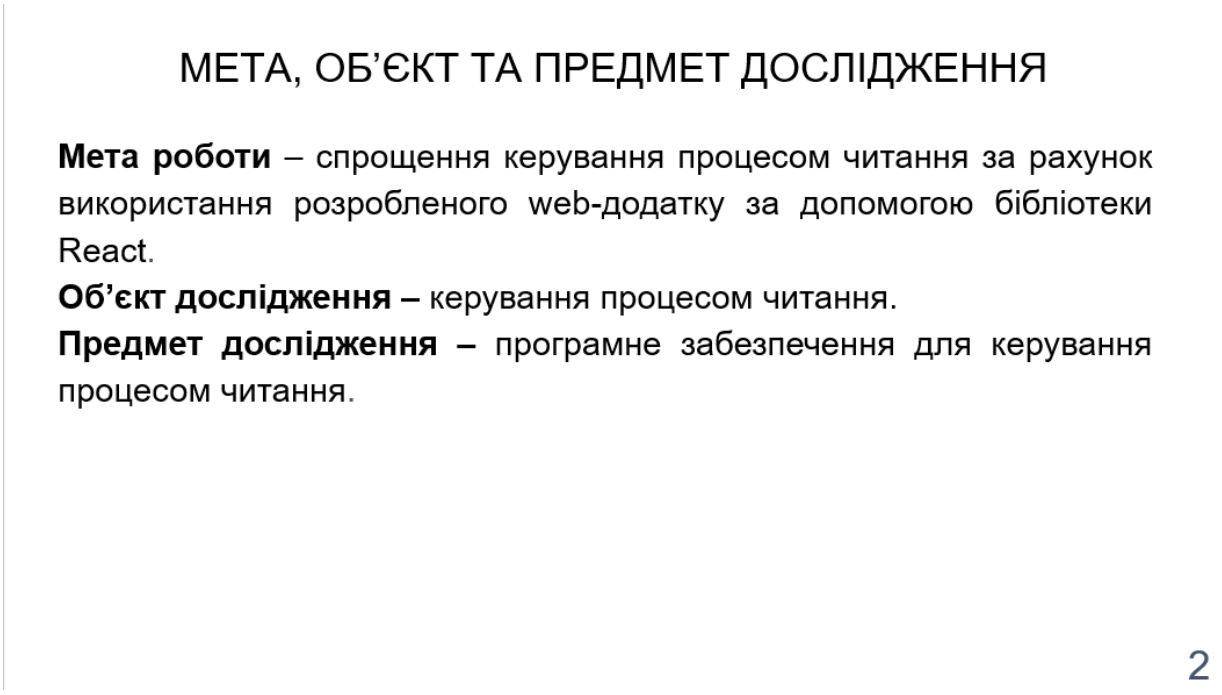
24. What is visual studio code? [Електронний ресурс] – Режим доступу до ресурсу: https://www.educba.com/what-is-visual-studio-code/.

25. Why is firebase an ideal pick for mobile app development?. [Електронний ресурс] – Режим доступу до ресурсу: https://citrusbug.com/blog/advantages-of-firebase-mobile-app-development.

ДОДАТОК А

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ





ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати аналоги та визначити їх переваги та недоліки.
2. Розглянути інструменти які будуть використані при розробці веб-додатку.
3. Спроектувати архітектуру web-додатку.
4. Розробити клієнтську частину додатку з використанням бібліотеки React.
5. Розробити серверну частину додатку на основі сервісу Firebase.
6. Протестувати розроблений додаток.

3

АНАЛІЗ АНАЛОГІВ

goodreads

LibraryThing

SCRIBD

4

ПЕРЕВАГИ ТА НЕДОЛІКИ АНАЛОГІВ

	Goodreads	Scribd	LibraryThing	BookHive
Пошук книг	+	+	+	+
Інтеграція з Google Book	-	-	-	+
Написання рецензії до книги	+	+	+	+
Додавання книг до списків для читання	+	+	+	+
Завантажувати книги у додаток	-	-	-	+

5

ВИМОГИ ДО ДОДАТКУ

1. Авторизація у додатку.
2. Додавання книги до списків для читання.
3. Здійснювання пошуку книг за назвою або жанром.
4. Переглянути детальну інформацію про книгу.
5. Виписування цитат з книг та написання рецензії.
6. Можливість завантажувати книги та читати їх у додатку.

6

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

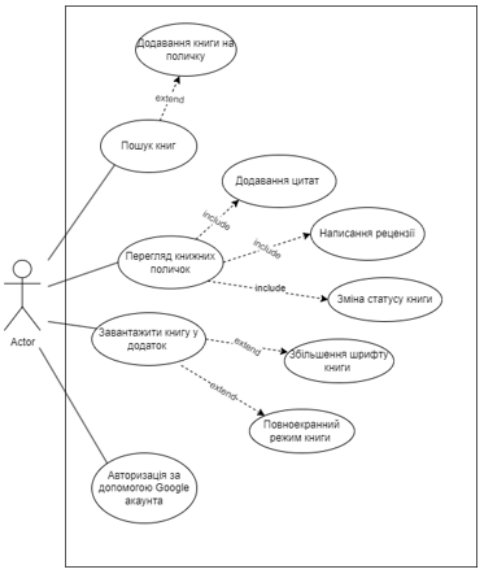
JavaScript

SASS

Material UI

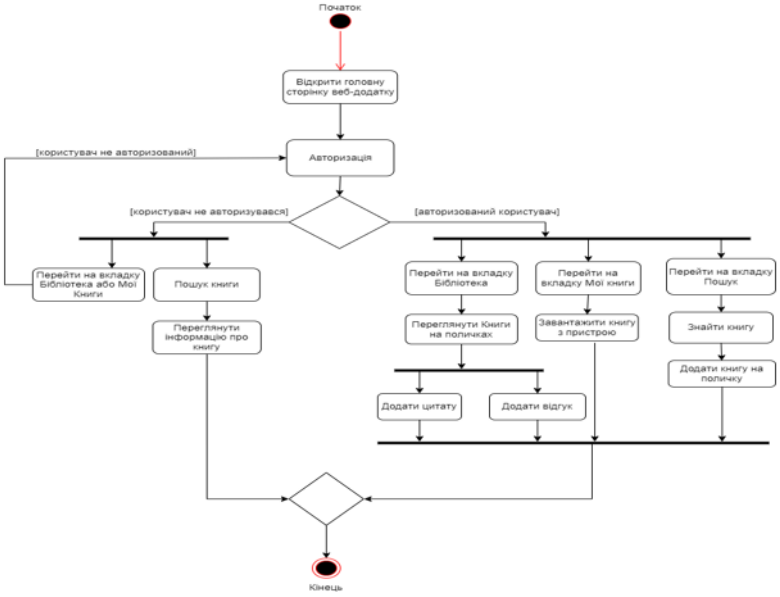
7

ДІАГРАМА ПРЕЦЕДЕНТІВ



8

ДІАГРАМА ДІЯЛЬНОСТІ ПРОГРАМИ



ДІАГРАМА КЛАСІВ

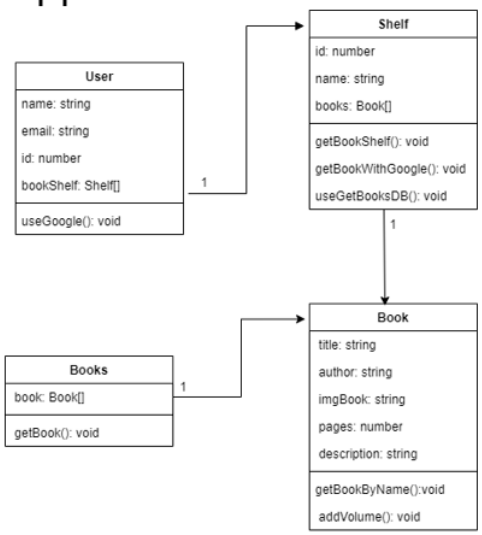
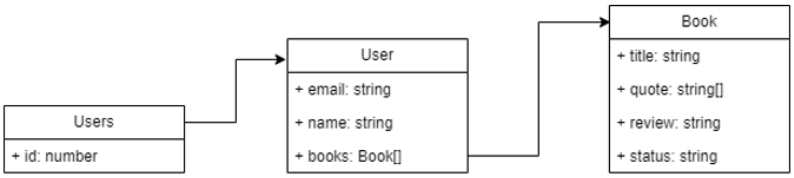


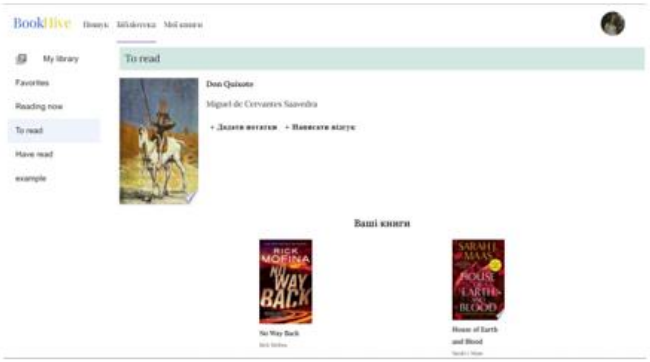
СХЕМА БАЗИ ДАНИХ



11

ЕКРАННІ ФОРМИ ДОДАТКУ

Головний екран додатку



Вкладка додатку «Бібліотека»

10

ЕКРАННІ ФОРМИ ДОДАТКУ

Вкладка «Мої книги»

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Федоренко А.А.// Переваги бібліотеки React для розробки web-додатків/ Трінтіна Н.А., Федоренко А.А.// Застосування програмного забезпечення в ІКТ: Матеріали всеукраїнської науково-технічної конференції. Збірник тез. 20.04.2023, ДУТ м. Київ – К.: ДУТ, 2023 – С. 59.
2. Федоренко А.А.//Проектування та розробка web-додатку для керування процесом читання/Трінтіна Н.А., Федоренко А.А.// Сучасні аспекти діджиталізації та інформатизації в програмній і комп'ютерній інженерії: Матеріали міжнародної науково-практичної конференції. Подано до друку.

ВИСНОВКИ

1. Проведено аналіз аналогів та визначено їх переваги та недоліки. Основним недоліком існуючих систем для керування процесом читання є відсутність інтеграції з Google Book та можливість завантажувати книги у додаток.
2. Розглянуто інструменти, які були використані під час розробки додатку, а саме: React, JS, SASS, MUI, Firebase, HTML.
3. Виконано проектування додатку: розроблено діаграму прецедентів, діаграму діяльності, діаграму класів та розроблено схему бази даних.
4. Розроблено клієнтську частину на основі бібліотеки React. Використано компонентний підхід бібліотеки який дозволяє перивикористовувати компоненти. Також використано маршрутизацію бібліотеки, що забезпечує оновлення вмісту без повного перевантаження сторінки.
5. Розроблено серверну частину додатку на основі сервісу Firebase. Зокрема використано модуль realtime database для збереження даних користувача.
6. Проведено функціональне тестування та тестування інтерфейсу. При функціональному тестуванні були виправлені дефекти та помилки. При тестуванні інтерфейсу усі елементи інтерфейсу користувача коректно відображаються на різних розширеннях екрану та в різних веб-браузерах.

15

ДЯКУЮ ЗА УВАГУ!