

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Застосунок для контролю читання книг»

Виконав студент групи ЗІПЗк-21-1
КРОСНІК Олександра Дмитрівна

Керівник роботи:
ЧИЖМОТРЯ Олексій Володимирович

Рецензент:
ЛОКТИКОВА Тамара Миколаївна

Житомир – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Завідувач кафедри інженерії

програмного забезпечення

Тетяна ВАКАЛЮК

«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КРОСНІК Олександра Дмитрівна**

Керівник роботи: **ЧИЖМОТРЕ Олексій Володимирович**

Тема роботи: «Застосунок для контролю читання книг»,

затверджена Наказом закладу вищої освіти від «23» лютого 2024 р., №75/с

Вихідні дані для роботи: **розробка застосунку для контролю читання книг**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів,
що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.В.	27.02.2024р.	27.02.2024р.
2	Чижмотря О.В.	27.02.2024р.	27.02.2024р.
3	Чижмотря О.В.	27.02.2024р.	27.02.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з програмного комплексу веб-сайту для застосунку для контролю читання книг та пояснювальної записки. Пояснювальна записка до випускної роботи містить текстову частину, викладену на 64 сторінках друкованого тексту, має 10 сторінок додатків. Список використаних джерел містить 30 найменувань і займає 3 сторінки. В роботі наведено 22 рисунки та 7 таблиць. Загальний обсяг роботи становить 82 сторінки.

Метою роботи є розробка застосунку для контролю читання книг, який надасть можливість користувачам зручно зберігати інформацію про книги, бачити свій прогрес читання та ставити цілі.

У роботі було визначено вимоги користувача та обґрунтовано створення концептуальної моделі. Також було проаналізовано аналоги програмних продуктів та обрано засоби реалізації, проведено розробку структури бази даних, а також розроблено алгоритм роботи, обрано архітектуру та розроблено об'єктно-орієнтовану структуру системи.

КЛЮЧОВІ СЛОВА: БАЗА ДАНИХ, ІНТЕРНЕТ, КНИГИ, ЖАНРИ, АВТОРИ, КОРИСТУВАЧІ, ЗАСТОСУНОК, ТРЕНУВАННЯ.

					ІПЗ.КР.Б – 121 – 24 – ПЗ					
Змн.	Арк.	№ докум.	Підпис	Дата						
Розробник	Кроснік О.Д.				Застосунок для контролю читання книг			Літ.	Арк.	Аркушів
Керівник	Чижмотря О.В.								3	82
								Житомирська політехніка, група ЗІПЗк-21-1		
Рецензент	Локтікова Т.М.									
Зав. каф.	Вакалюк Т.А.									

ABSTRACT

The Bachelor's thesis consists of a software complex for a website application for book reading control and an explanatory note. The explanatory note contains a textual part, presented on 64 pages of printed text, with 10 pages of appendices. The list of used sources includes 30 titles and occupies 3 pages. The thesis includes 22 figures and 7 tables. The total volume of the work is 83 pages.

The aim of the thesis is to develop an application for book reading control that will allow users to conveniently store information about books, track their reading progress, and set goals.

The thesis identifies user requirements and justifies the creation of a conceptual model. Analogous software products were analyzed, implementation tools were selected, the structure of the database was developed, and an algorithm of operation was devised. Additionally, an architecture was chosen, and an object-oriented system structure was developed.

KEYWORDS: DATABASE, INTERNET, BOOKS, GENRES, AUTHORS, USERS, APPLICATION, TRAINING.

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT.....	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ЗАСТОСУНКУ КОНТРОЛЮ ЧИТАННЯ КНИГ.....	9
1.1 Постановка задачі.....	9
1.2 Аналіз аналогів програмного продукту	11
1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.....	19
Висновки до першого розділу.....	24
РОЗДІЛ 2. ПРОЕКТУВАННЯ ЗАСТОСУНКУ ДЛЯ КОНТРОЛЮ ЧИТАННЯ КНИГ.....	26
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	26
2.2 Розробка бази даних системи.....	38
2.3 Проектування та реалізація алгоритмів роботи системи	45
2.4 Реалізація функціоналу застосунку для контролю читання книг	48
Висновки до другого розділу.....	53
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ЗАСТОСУНКОМ ДЛЯ КОНТРОЛЮ ЧИТАННЯ КНИГ	55
3.1 Порядок встановлення та налаштування параметрів сайту.....	55
3.2 Структура інтерфейсу сайту з контролю читання книг.....	56
3.3 Тестування сайту.....	66
Висновки до третього розділу	66
ВИСНОВКИ.....	68
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	69
ДОДАТКИ.....	71

Перелік умовних позначень

JS (JavaScript) – мова програмування, яка використовується для створення інтерактивних та динамічних веб-сторінок.

HTML (HyperText Markup Language) – мова розмітки, яка використовується для створення та структурування веб-сторінок.

CSS (Cascading Style Sheets) – мова стилів, яка використовується для опису зовнішнього вигляду та форматування веб-документів.

JWT (JSON Web Token) - це стандарт токена автентифікації, який використовується для передачі інформації між сторонами у вигляді JSON об'єкта. Він складається з трьох частин: заголовка, тіла та підпису, які розділені крапкою.

bcryptjs - це бібліотека для JavaScript, яка використовується для хешування паролів і перевірки їхніх хешів. Хешування паролів є важливим елементом забезпечення безпеки в веб-додатках, оскільки воно дозволяє зберігати паролі в зашифрованому вигляді, захищаючи їх від злоумисників у разі витоку бази даних.

ВСТУП

В епоху сучасного світу, який насичений технологіями і інформацією, постає питання ефективного управління часом та контролю власної діяльності. Читання книг та відслідковування його процесу є однією з важливих сфер, що потребують уваги. Незважаючи на появу великого потоку цифрової інформації, книги залишаються незамінними для поглиблення свої знань, відпочинку, розвитку критичного мислення та самовдосконалення. В умовах постійної зайнятості та швидкоплинності часу, багато людей стикаються з проблемою організації свого читання.

Актуальність теми бакалаврської роботи полягає в тому, щоб розробити застосунок для контролю читання книг, який дозволить користувачам ефективно планувати, відслідковувати та аналізувати свій процес читання.

Мета полягає у створенні інструменту, що допоможе користувачам отримувати більше задоволення від читання, дотримуватись поставлених цілей та підвищити продуктивність.

Встановлена мета обумовлює наступні завдання:

- Провести аналіз сучасного ринку онлайн застосунків для контролю читання книг. Визначити їх переваги та недоліки.

- Визначити основні проблеми, пов'язані з організацією читання книг та обґрунтувати актуальність розробки спеціалізованого застосунку для їх вирішення

- Визначення технологій для реалізації задачі;

- Визначення функціональних вимог предметної області дослідження;

- Розробити інтерфейс платформи, який би був інтуїтивно зрозумілим та забезпечував зручний пошук, перегляд, сортування книг та поставлення цілей для читання;

- Провести тестування та оцінку ефективності;

Об'єктом дослідження є процес читання книг користувачами та методи їх контролю та організації за допомогою сучасних технологій

Предметом дослідження є розробка для впровадження застосунку для контролю читання книг, який включає функції реєстрації користувача, додавання книг та відстеження статусу (планую прочитати, читаю, прочитані) та прогресу читання книг, створення та управління цілями читання, а також ведення прогресу читання.

Очікувані результати

Очікуваними результатами роботи є підвищення ефективності читання. Користувачі зможуть більш ефективно використовувати свій час, завдяки можливості створення цілей та планування читання. Також це дозволить розподіляти навантаження слідкуючи за прогресом, що сприятиме мотивації до читання та підвищенню продуктивності. Також результатом роботи є поліпшення організації читання. Застосунок дозволить впорядковувати свої книги за статусами та колекціями, що забезпечить чітке розуміння поточного стану їхньої бібліотеки. Застосунок спонукатиме користувачів розвивати навички самодисципліни та тайм-менеджменту, оскільки виконання поставлених цілей та регулярне дотримання графіка читання потребує відповідальності та організації. Також результати досліджень та відгуки користувачів допоможуть ідентифікувати нові потреби та можливості для вдосконалення застосунку.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ЗАСТОСУНКУ КОНТРОЛЮ ЧИТАННЯ КНИГ

1.1 Постановка задачі

Кваліфікаційна робота передбачає створення інтернет-додатку для моніторингу читання книг. Основна мета завдання – розробити зручний і ефективний інструмент, який дозволить користувачам організовувати процес читання, ставити цілі, відстежувати прогрес і керувати своєю бібліотекою.

Додаток покликаний вирішити проблему відсутності мотивації та тайм-менеджменту для читання книг. Завдяки впровадженню цього інструменту користувачі зможуть краще планувати свій час, відстежувати прогрес і досягати своїх цілей щодо читання.

Веб-застосунок для контролю читання книг має наступні ключові функції:

- **Реєстрація та авторизація:** Забезпечує створення облікового запису користувача для збереження персональних даних та прогресу.
- **Додавання та управління книгами:** Користувач може додавати нові книги до своєї бібліотеки з можливістю вказати, чи була книга вже прочитана раніше, та скільки разів.
- **Категоризація книг:** Книги розподіляються на категорії: "планую прочитати", "читаю" та "прочитані", що спрощує навігацію по бібліотеці.
- **Редагування інформації про книги:** Можливість змінювати дані про книги, такі як назва, автор, кількість сторінок та інші параметри.
- **Постановка цілей читання:** Користувач може створювати цілі з читання, вказуючи дати початку та завершення, а також список книг, які планується прочитати в цей період.
- **Планування та відстеження прогресу:** Автоматичне формування графіка читання, який розподіляє кількість сторінок, що потрібно прочитати щодня для досягнення цілі.

- **Запис прогресу читання:** Форма для заповнення кількості прочитаних сторінок за певний день, що дозволяє користувачу відслідковувати свій щоденний прогрес.
- **Категоризація тренувань:** Розподіл читання на "заплановані", "активні" та "завершені", що допомагає користувачам краще організовувати свій час та слідкувати за виконанням поставлених цілей.

Завдяки впровадженню цього веб-застосунку рівень організації читання якісно підвищується, зростає мотивація користувачів та ефективність використання часу. Він забезпечує наступні переваги:

- Спрощення введення та пошуку інформації про книги;
- Прискорення обробки та передачі інформації;
- Зручність оперування базою даних та точність введених даних;
- Підвищення мотивації до досягнення поставлених цілей у читанні.

З огляду на перелічені переваги, було визначено перелік задач, які потрібно вирішити для досягнення даного результату:

- Реєстрація та авторизація користувачів;
- Створення та управління каталогом книг;
- Редагування інформації про книги;
- Видалення інформації про книги;
- Постановка та управління цілями читання;
- Формування графіка читання;
- Запис та відстеження прогресу читання;
- Категоризація тренувань.

Для реалізації описаних задач у системі необхідні наступні об'єкти:

1. Користувач

- 1.1. Ідентифікатор
- 1.2. Ім'я
- 1.3. Електронна пошта

2. Книга

- 2.1. Ідентифікатор
- 2.2. Назва

- 2.3. Автор
- 2.4. Дата публікації
- 2.5. Кількість сторінок
- 2.6. Статус
- 2.7. Чи знаходиться книга в тренуванні
- 2.8. Обкладенка книги
- 2.9. Кількість прочитаних разів
- 2.10. Оцінка книги
- 2.11. Відгук про книгу
- 3. Тренування
- 3.1. Ідентифікатор
- 3.2. Дата початку
- 3.3. Дата кінця
- 3.4. Назва
- 3.5. Статус

Цей веб-застосунок допоможе користувачам не лише більш організовано підходити до процесу читання, але й зробить цей процес більш захоплюючим та мотивуючим.

1.2 Аналіз аналогів програмного продукту

Для успішного створення ефективного веб-додатку для моніторингу читання книг необхідно проаналізувати існуюче програмне забезпечення, яке використовується в Україні та за кордоном. Це дозволить виявити їхні сильні та слабкі сторони, а також визначити найкращі практики та технології, які можна включити в нову систему.

Популярне на ринку програмне забезпечення для моніторингу читання книг

1. Goodreads

Опис: Goodreads - одна з найбільших у світі соціальних мереж для книголюбів, що дозволяє користувачам відстежувати, оцінювати та рецензувати книги, які вони читають. як показано на рис. 1.1.

Переваги:

- Велика спільнота користувачів;

					ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
						11

- Можливість створення власних книжкових полиць;
- Рекомендаційна система на основі вподобань користувача.

Недоліки:

- Інтерфейс може бути перевантажений функціональністю;
- Відсутність деяких локалізацій, зокрема української мови;
- Висока соціальна спрямованість, що не завжди необхідно.

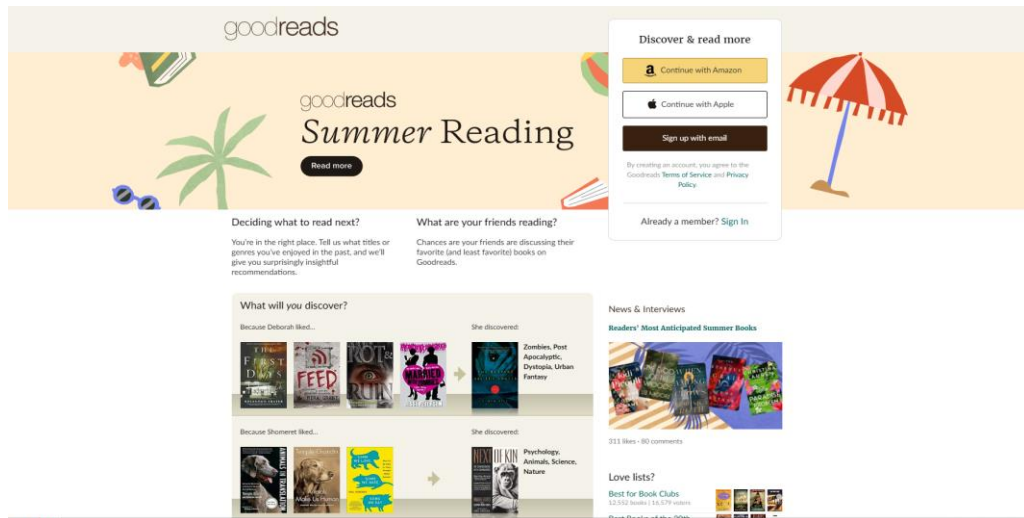


Рисунок 1.1. Онлайн магазин книг **Goodreads** [1]

2. LibraryThing

Опис: LibraryThing — це служба онлайн-каталогізації книг, яка дозволяє користувачам створювати власні бібліотеки та ділитися ними з іншими користувачами.

Переваги:

- Можливість каталогізації великих бібліотек;
- Інтеграція з іншими сервісами для отримання додаткової інформації про книги

Недоліки:

- Обмежений набір функцій для відстеження прогресу читання;
- Інтерфейс може виглядати застарілим;
- Немає особливостей соціальної взаємодії

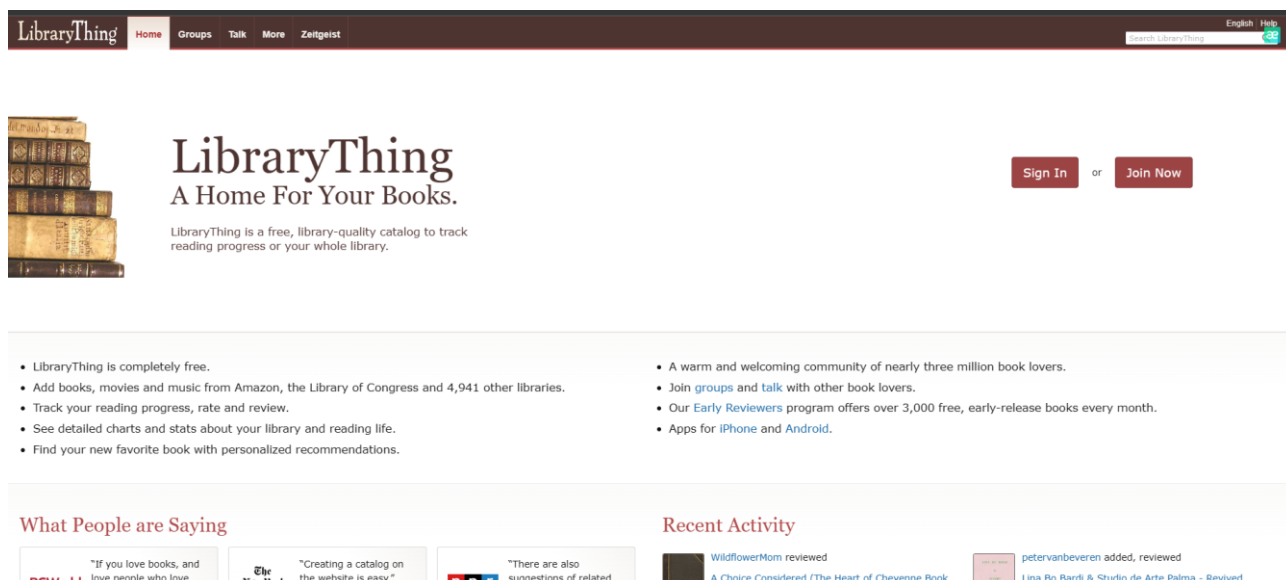


Рисунок 1.2. Онлайн магазин книг **LibraryThing** [2]

3. Bookworm

Опис: Bookworm — це веб-програма, яка дозволяє користувачам відстежувати книги, які вони читають, ставити цілі та записувати свій досвід. (рис. 1.3).

Переваги:

- Простота використання;
- Простий інтерфейс для введення даних книги;
- Уміння ставити цілі читання.

Недоліки:

- Обмежений функціонал статистики;
- Відсутність соціальних особливостей;
- Немає підтримки експорту та імпорту даних.

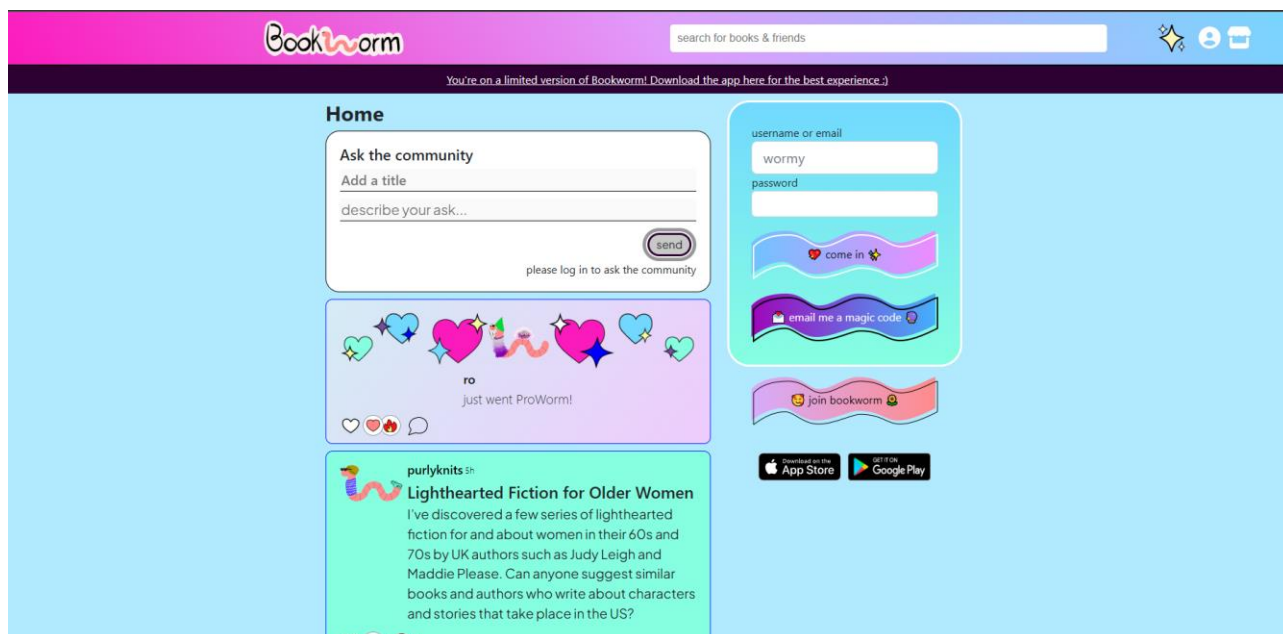


Рисунок 1.3. Онлайн магазин книг **Bookworm** [3]

4. Book Track

Опис: Book Track — мобільний додаток, який дозволяє користувачам відстежувати прочитані книги та впорядковувати свою бібліотеку (рис. 1.4).

Переваги:

- Простота використання;
- Можливість створення списків книг;
- Інтуїтивно зрозумілий інтерфейс.

Недоліки:

- Відсутність соціальних особливостей;
- Немає розширеної статистики читання;
- Немає функцій експорту та імпорту даних

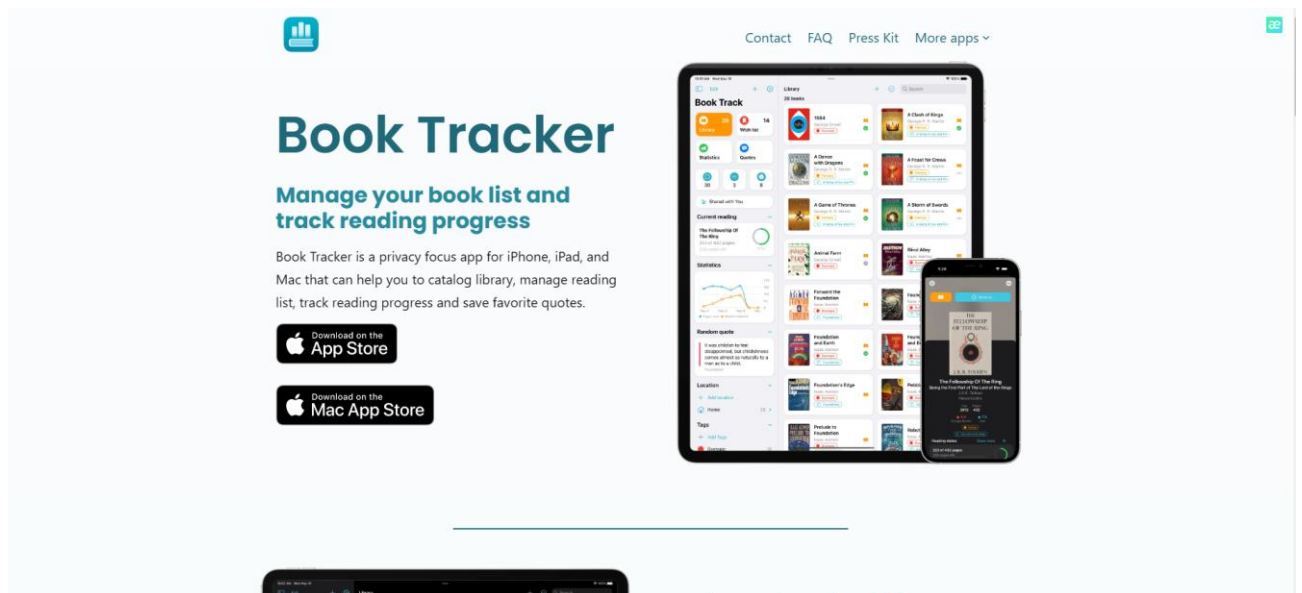


Рисунок 1.4. Онлайн магазин книг **Book Track** [4]

5. Rork

Опис: Rork — український веб-додаток для відстеження прогресу читання книг. Користувачі можуть додавати книги до своєї бібліотеки, відзначати свій прогрес у читанні та ставити цілі. (рис. 1.5).

Переваги:

- Інтуїтивно зрозумілий і зручний інтерфейс;
- Можливість відстежувати прогрес читання;
- Постановка цілей і система нагадувань.

Недоліки:

- Відсутність розширеної статистики та аналізу;
- Обмежені соціальні особливості;
- Немає можливості експортувати та імпортувати дані.

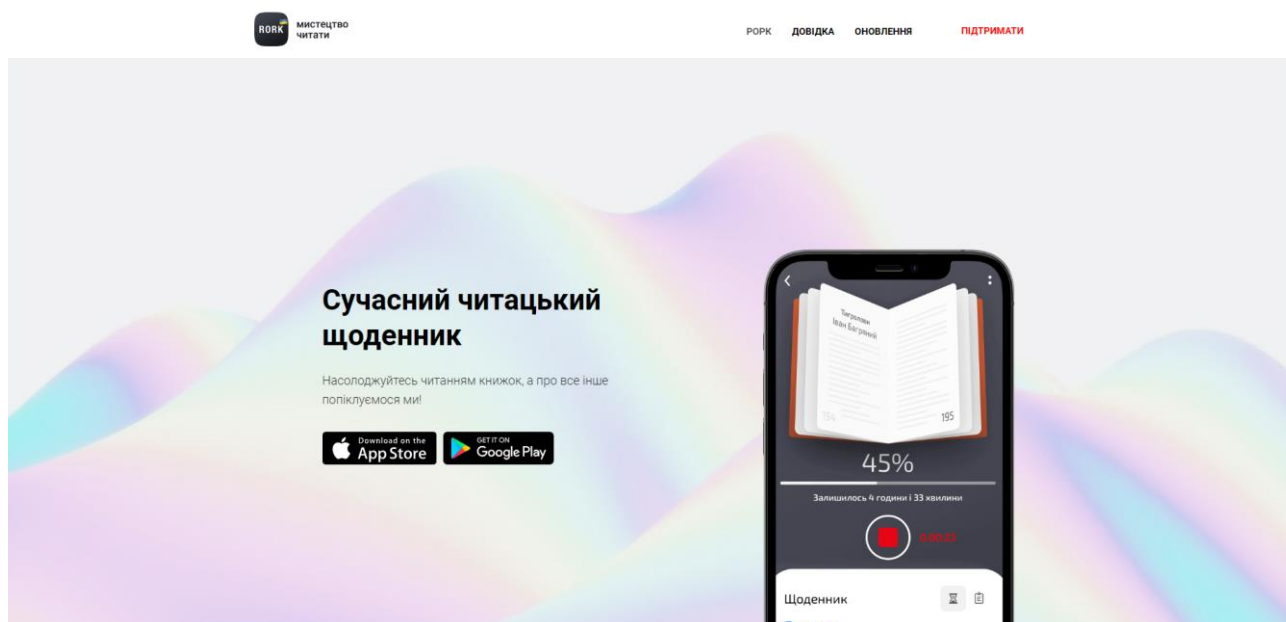


Рисунок 1.5. Онлайн магазин книг **Rork** [5]

Аналізуючи ці програмні продукти, можна виділити ряд важливих аспектів розробки веб-додатку для моніторингу читання книг:

- Інтуїтивно зрозумілий і зручний інтерфейс;
- Можливість відстежувати прогрес читання та ставити цілі;
- Проста та ефективна організація бібліотеки користувача;
- Зосередьтеся на основній функції без непотрібних соціальних функцій.

Ці аспекти будуть враховані при розробці нового веб-додатку, що дозволить створити ефективний інструмент для користувачів, який відповідає їхнім потребам та очікуванням.

Порівняння характеристик: **Goodreads**, **LibraryThing**, **Bookworm**, **Book Track**, **Rork** кваліфікаційної роботи наведено в таблиці 1.1.

Таблиця 1.1

Порівняння характеристик застосунків

Характеристика	Goodreads	LibraryThing	Bookworm	Book Track	Rork
Інтерфейс сайту	Сучасний та зручний для навігації, з зручними	Простий та зрозумілий інтерфейс, легкий для	Модерний дизайн, зручна навігація та	Привабливий та зрозумілий інтерфейс, з	Сучасний дизайн, легкий у використанні,

	фільтрами та категоріями	користуванн я	швидкий пошук книг	функціональ ним пошуком	зручний пошук та фільтри
Відстеженн я прогресу читання	Можливість відстеження прогресу читання, встановлення цілей, щоденні оновлення	Обмежена функціональ ність для відстеження прогресу читання	Простий інтерфейс для введення даних про прочитані сторінки, базове відстеження	Детальна статистика читання, нагадування для досягнення цілей	Відстеження прочитаних сторінок, розподіл на заплановані, активні та завершені книги
Організаці я бібліотеки	Можливість створення власних книжкових полиць, сортування за категоріями	Широкі можливості для каталогізації великих бібліотек, інтеграція з іншими сервісами	Простий інтерфейс для організації особистої бібліотеки	Можливість створення списків книг, організація бібліотеки	Простий інтерфейс для додавання та редагування книг, можливість розподілу книг на планую прочитати, читаю та прочитані
Постановка цілей	Можливість встановлення річних цілей, рекомендації на основі цілей	Відсутність функцій для постановки цілей	Можливість встановленн я цілей читання, щоденний прогрес	Можливість встановленн я цілей, нагадування для досягнення цілей	Можливість створення цілей з визначеними термінами та розбивкою на дні, вказівка на кількість сторінок для читання на день

Кожен із перерахованих сервісів для моніторингу читання книг має свій унікальний інтерфейс, розроблений для певної аудиторії. Наприклад, Goodreads і Book Track пропонують сучасні та прості в навігації інтерфейси. Сучасний дизайн, продумане розташування елементів і зручні фільтри допомагають користувачам швидко знаходити потрібні книги і відстежувати прогрес у читанні. Такий підхід робить використання сервісу приємним процесом, що особливо важливо для молоді та активних користувачів Інтернету.

LibraryThing, навпаки, має простий і зрозумілий інтерфейс, який особливо цінують досвідчені користувачі або ті, хто не звик до складних технологій. Чітка структура сторінки, відсутність зайвих елементів і простий формат додавання книг роблять цей сервіс привабливим для більш традиційної аудиторії. Такий підхід допомагає уникнути непорозумінь і полегшує процес організації особистої бібліотеки.

Відгуки користувачів також впливають на вибір сервісу, і тут Goodreads виділяється великою кількістю відгуків і можливістю обговорення книг. Розділ відгуків дозволяє користувачам ділитися своїми враженнями від прочитаних книг, що створює відчуття спільності та надає додаткову інформацію потенційним користувачам. Можливість обговорювати книги безпосередньо на сайті створює інтерактивне середовище, в якому читачі можуть спілкуватися, рекомендувати книги один одному та отримувати рекомендації від інших користувачів.

Bookworm і Rork також мають свої унікальні переваги. Завдяки сучасному дизайну та простоті використання Bookworm привертає увагу тих, хто любить швидкий доступ до функцій і зручну організацію своїх бібліотек. Rork пропонує сучасний дизайн, простоту використання та зручні фільтри, що робить його привабливим для користувачів, які цінують естетику та ефективність.

Тому кожен із цих сервісів має свої особливості, які є визначальними для різних категорій користувачів, забезпечуючи їм зручний та ефективний спосіб відстеження читання та організації своїх бібліотек..

1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Вибір інструментальних засобів

Для розробки веб-застосунку для контролю читання книг було обрано такі інструменти: React, Redux, Node.js, Express, MongoDB, Ant Design, Styled-Components та Cloudinary.

Кожен з цих інструментів має свої унікальні переваги, які сприяють ефективній розробці, зручності в підтримці та масштабованості застосунку. Нижче наведено детальний опис кожного інструменту та причини їх вибору.

React

Опис: React - це бібліотека JavaScript для побудови користувацьких інтерфейсів, розроблена Facebook. Вона дозволяє створювати компоненти, які можна повторно використовувати, що сприяє розширюваності та підтримці коду.

Переваги:

- **Компонентний підхід:** React дозволяє розділити застосунок на незалежні компоненти, що спрощує розробку та тестування.
- **Віртуальний DOM:** впровадження віртуального DOM забезпечує високу продуктивність завдяки мінімізації змін у реальному DOM.
- **Односпрямований потік даних:** це робить керування станом більш передбачуваним і легшим для налагодження.
- **Широке використання:** React активно використовується багатьма великими компаніями, надаючи велику спільноту та багато ресурсів підтримки.

Redux

Опис: Redux — це бібліотека для керування станом програми. Його часто використовують у поєднанні з React для централізованого зберігання стану та забезпечення передбачуваної поведінки програми.

Переваги:

- **Централізоване керування станом:** усі дані зберігаються в єдиному об'єкті стану, що спрощує керування та налагодження.

- **Передбачуваність:** суворі правила щодо змін стану полегшують відстеження змін і уникнення помилок.
- **Розширюваність:** Легко розширюється за допомогою середовищ (middleware), що дозволяє додавати додаткову функціональність.
- **Інтеграція DevTools:** Redux DevTools надає потужні інструменти для налагодження та моніторингу стану додатків.
- **Активна спільнота:** велика спільнота користувачів і розробників, що надає багато ресурсів і підтримки.

Node.js

Опис: Node.js — це середовище виконання JavaScript, яке дозволяє запускати код JavaScript на сервері. Він побудований на движку Google V8 і забезпечує високу продуктивність.

Переваги:

- **Асинхронна обробка:** підтримка асинхронних операцій дозволяє обробляти велику кількість запитів одночасно.
- **Однорідність коду:** Використання JavaScript як на клієнтській, так і на серверній стороні забезпечує узгодженість та спрощує процес розробки.
- **Велика екосистема:** широкий вибір бібліотек і модулів, доступних через npm (Node Package Manager).
- **Масштабованість:** Node.js можна легко масштабувати для обробки великих навантажень.
- **Швидкодія:** Завдяки рушію V8 та асинхронній природі, Node.js забезпечує високу продуктивність та швидкість роботи.

Express

Опис: Express - це мінімалістичний фреймворк для веб-додатків, побудований на Node.js. Він забезпечує набір інструментів для побудови веб-серверів та API.

- **Простота використання:** простий у налаштуванні та використанні, забезпечує простий спосіб створення серверних програм.

- **Модульність:** Легко розширюється за допомогою middleware, що дозволяє додавати різноманітні функціональності.
- **Широко використовується:** Express є одним із найпопулярніших фреймворків для Node.js, що забезпечує велику спільноту та багато ресурсів підтримки.
- **Гнучкість:** підтримує різні методи HTTP та маршрутизації, що дозволяє створювати ефективні та гнучкі API.
- **Інтеграція з іншими бібліотеками:** проста інтеграція з іншими бібліотеками та фреймворками, що розширює функціональність.

MongoDB

Опис: MongoDB — це нереляційна база даних, яка зберігає дані як документи у JSON-подібному форматі (BSON). Він ідеально підходить для роботи з великими обсягами даних, що постійно змінюються.

- **Гнучкість схеми:** можливість динамічно змінювати структуру даних без необхідності перебудови всієї бази даних.
- **Масштабованість:** підтримує горизонтальне масштабування та розподіл даних між кількома серверами.
- **Висока продуктивність:** ефективна обробка великих обсягів даних і швидкі операції читання та запису.
- **Інтеграція JavaScript:** можливість використовувати JavaScript для запитів і обробки даних, що спрощує програмування.
- **Резервне копіювання та відновлення:** підтримує вбудовані механізми резервного копіювання та відновлення.

Ant Design

Опис: Ant Design - це набір компонентів для React, розроблений компанією Ant Financial. Він забезпечує готові до використання компоненти для створення професійних інтерфейсів.

- **Повний комплект компонентів:** містить усі компоненти, необхідні для створення сучасної веб-програми.

- **Послідовність дизайну:** усі компоненти дотримуються одного стилю, що забезпечує узгоджений інтерфейс.
- **Висока персоналізація:** легко адаптуватися до потреб конкретного проекту.
- **Документація та спільнота:** широка документація та активна спільнота користувачів спрощують вивчення та використання.
- **Модульність:** можна використовувати лише необхідні компоненти, зменшуючи обсяг завантажуваних даних.

Styled-Components

Опис: Styled-Components — це бібліотека для написання CSS у JavaScript. Це дозволяє створювати стилізовані компоненти з ізольованими стилями, що підвищує модульність коду та зручність обслуговування.

- **Модульність:** Стили прив'язані до компонентів, що забезпечує ізоляцію та уникнення конфліктів.
- **Динамічні стилі:** Можливість створювати стилі на основі пропсів компонентів, що підвищує гнучкість.
- **Підтримка CSS:** підтримка всіх можливостей CSS, включаючи псевдокласи та медіа-запити.
- **Зменшені залежності:** немає потреби у великих файлах CSS, що зменшує складність дизайну.
- **Продуктивність:** оптимізовано продуктивність за допомогою CSS-in-JS.

Cloudinary

Опис: Cloudinary — сервіс для зберігання та обробки фотографій і відео в хмарі. Він надає зручні інструменти для завантаження, зберігання, обробки та доставки мультимедійного вмісту.

- **Оптимізація мультимедійного вмісту:** Автоматична оптимізація зображень та відео для покращення продуктивності веб-додатків.
- **Проста інтеграція:** простий API для інтеграції з веб-додатками.
- **Масштабованість:** підтримка великої кількості мультимедійного вмісту при збереженні високої доступності та продуктивності.

- **Безпека:** підтримує різні рівні безпеки для захисту вашого мультимедійного вмісту.
- **Гнучкість обробки:** можливість виконувати різноманітні операції над зображеннями та відео, як-от зміна розміру, форматування та застосування фільтрів.

Вимоги до апаратного забезпечення

Щоб розробити та запустити веб-додаток за допомогою наведених вище інструментів, необхідне апаратне забезпечення має відповідати таким вимогам:

Серверні вимоги:

- **Процесор:** Багатоядерний процесор з частотою не менше 2,5 ГГц.
- **Оперативна пам'ять:** мінімум 8 ГБ оперативної пам'яті, для зручності роботи рекомендується 16 ГБ.
- **Жорсткий диск:** SSD об'ємом не менше 256 ГБ для швидкої роботи з даними.
- **Операційна система:** Windows 10, macOS або будь-який сучасний дистрибутив Linux.
- **Мережа:** стабільне підключення для роботи з хмарними сервісами та бібліотеками.

Вимоги до робочих станцій розробників:

- **Процесор:** Багатоядерний процесор (не менше 4 ядер) з тактовою частотою не менше 2.5 ГГц. Рекомендовано Intel Core i7 або AMD Ryzen 7 для забезпечення високої продуктивності.
- **Оперативна пам'ять:** Мінімум 8 ГБ ОЗП, рекомендується 16 ГБ для комфортної роботи з середовищами розробки та одночасного запуску декількох додатків.
- **Жорсткий диск:** Мінімум 256 ГБ SSD для швидкого доступу до файлів проекту. Рекомендовано 512 ГБ для зберігання великих обсягів даних і резервних копій.

- **Операційна система:** Сучасна версія Windows 10 або 11, macOS 10.15 Catalina або новіша, або дистрибуція Linux, така як Ubuntu 20.04 або новіша, яка сумісна з використовуваними інструментами розробки.
- **Додаткове програмне забезпечення:** середовище розробки (IDE) Visual Studio Code, WebStorm або інше, що підтримує розробку на JavaScript; система керування версіями Git для управління версіями коду; платформи для розробки Node.js для серверної частини з пакетним менеджером npm; бази даних MongoDB для зберігання даних застосунку; інструменти для управління проєктами та тестування Postman для тестування API; фреймворки та бібліотеки React для фронтенд-розробки, Redux для управління станом застосунку, Express для серверної частини; інструменти для стилізації Ant Design для компонентів інтерфейсу, Styled-Components для налаштування стилів; обробка медіа Cloudinary для зберігання та обробки зображень і відео.

Таке апаратне та програмне забезпечення забезпечує високу продуктивність, ефективність та комфорт розробки веб-застосунку для контролю читання книг.

Висновки до першого розділу

У першому розділі дипломного проєкту була проведена детальна постановка завдання та аналіз вимог до розробки застосунку для контролю читання книг.

Визначено основні функції та можливості, якими повинен володіти цей додаток для задоволення потреб користувачів.

Аналіз аналогічних програмних рішень дозволив виявити їх переваги та недоліки, що стало основою для формулювання вимог до розробки власного додатку. Вибрані інструменти та технології, такі як React, Redux, Node.js, Express та MongoDB, відповідають потребам у забезпеченні швидкості, ефективності та зручного користування.

Отже, обрані інструменти та технології мають на меті забезпечити реалізацію функціонального, надійного та зручного застосунку для контролю читання книг, що відповідає сучасним вимогам та потребам користувачів.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ЗАСТОСУНКУ ДЛЯ КОНТРОЛЮ ЧИТАННЯ КНИГ

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи

Для забезпечення неперервного виконання необхідних функцій, наведено високорівневі вимоги, які повинна задовольняти веб-застосунок для контролю читання книг. Ці вимоги допомагають забезпечити безперебійну роботу системи.

Бізнес вимоги

1. Основні цілі: проект створюється з метою відслідковування процесу читання.
2. Представлення проекту: проект буде реалізовано у вигляді сайту

Вимоги користувачів

1. Можливість реєстрації та авторизації
2. Можливість реєстрації та авторизації через обліковий запис в Google
3. Можливість додавання книг та редагування введених даних.
4. Можливість оцінки книги та написання відгуку до неї
5. Можливість відслідковувати процес читання кожної книги
6. Можливість ставити цілі читання та слідкувати за прогресом.
7. Можливість фіксувати проміжні етапи досягнення цілі читання.
8. Можливість перегляду історій цілей читання.

Функціональні вимоги:

1. Авторизація у системі: в системі повинна бути представлена можливість реєстрації користувача
2. Можливість збереження інформації: система повинна зберігати інформацію і надавати можливість користувачу керувати нею.
3. Можливість входу в Google обліковий запис.

Нефункціональні вимоги:

1. Сприйняття користувачем:
 - Середній час навчання звичайних користувачів повинен бути в межах 3-4 годин, а просунутих користувачів - 1 година.

- Максимальний час очікування відповіді для типових завдань - не більше 15 секунд, для складних завдань - не більше 30 секунд.

- Інтерфейс повинен бути інтуїтивно зрозумілим і містити відповідні підказки.

2. Надійність та доступність:

- Час, який витрачається на обслуговування системи, не повинен перевищувати 6% від загального часу її роботи.
- Середній час безвідмовної роботи повинен становити 15 робочих днів.
- Максимальна норма помилок та дефектів в роботі системи - не більше 1 помилки на кожні десять тисяч рядків коду.

3. Продуктивність:

- Система повинна підтримувати одночасну роботу принаймні 40 користувачів, які працюють з загальною базою даних.
- Можливість масштабування:
- Система повинна бути здатною збільшувати свої ресурси та продуктивність зі збільшенням кількості користувачів, не негативно впливаючи на її працездатність.

Вивчення потреб користувачів дало можливість визначити можливі сценарії використання застосунку для контролю читання книг. На рисунку 2.1 наведена діаграма, що відображає різні способи взаємодії з програмною системою.

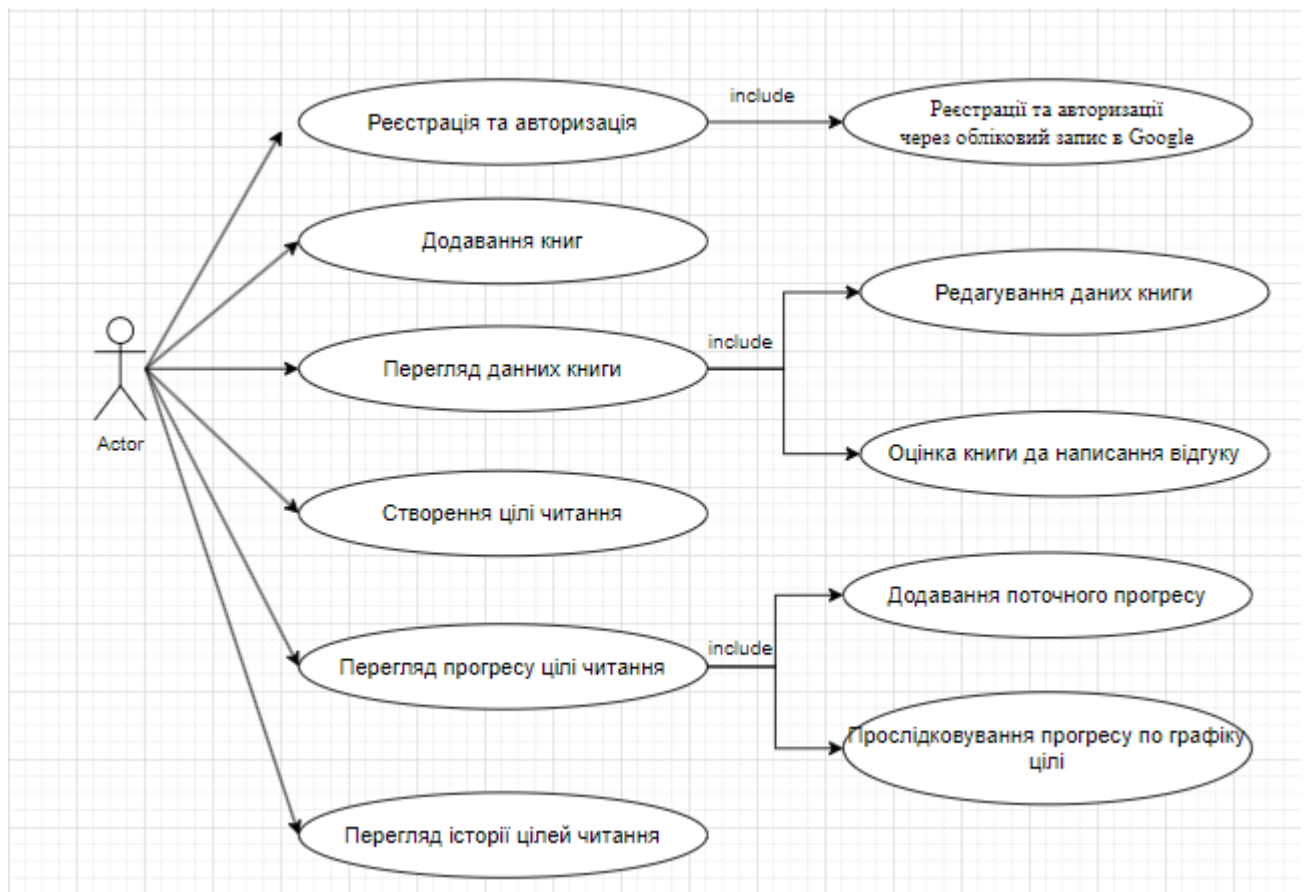


Рисунок 2.1. Варіанти використання застосунку

Відомості про користувача

У системі існують окремі користувачі. Кожен з відвідувачів має можливість зареєструватись в системі використовуючи свою пошту або обліковий запис Google. Користувач може додавати книги, які він планує прочитати. Книги діляться на три категорії: «прочитані», «читаю», «планую прочитати». У користувача є можливість створювати план читання книг. Він може обрати період і книги, які він хоче прочитати за цей період. Також прослідковувати прогрес планування, вводячи інформацію про те, скільки сторінок якої книги він прочитав у вибраний день.

Призначене для користувача середовище

Даний продукт є web-додатком. Відкрити додаток можна в будь-якому доступному веб-браузері.

Виявлені варіанти використання зібрані в Таблиці 2.1.

Виявлені варіанти використання

Автор	Найменування	Формулювання
Користувач	Реєстрація	Новий користувач може зареєструватись використовуючи свою пошту або обліковий запис в Google
Користувач	Вхід	Щоб авторизуватись в системі потрібно ввести логін і пароль або використати обліковий запис в Google
Користувач	Додавання нової книги	Користувач вводить інформацію про книгу: назва книги, автор, кількість сторінок, рік видання і вказує чи книга була прочитана чи ні. Якщо книга була прочитана, то у користувача буде можливість вказати скільки разів.
Користувач	Перегляд книг	Книги діляться на три списки по статусу: «прочитані», «читаю», «планую прочитати»
Користувач	Перегляд і редагування книги	Користувач може відкоригувати інформацію про книгу, а також переглянути чи видалити книгу, натиснувши на неї
Користувач	Планування читання	Користувач може обрати період та книжки, як він хоче прочитати за цей період. Після старту планування, користувач кожного дня може вводити скільки сторінок він прочитав певної книги. Планування закінчується, якщо вийшов час з статусом «не успішно», або якщо всі книги прочитані з статусом «успішно»
Користувач	Перегляд історії читання	Користувач може переглянути вже закінченні планування.

Реєстр варіантів використання

Результуючий список варіантів використання показаний в таблиці 2.2.

Таблиця 2.2.

Реєстр варіантів використання

Код	Основний актор	Найменування	Формулювання
K1	Користувач	Реєстрація в системі	Користувач може зареєструватись напрямку в додатку або використовуючи обліковий запис Google
K2	Користувач	Вхід у систему	Користувач може увійти в додаток використовуючи логін та пароль або обліковий запис Google
K3	Користувач	Додавання книги	Аутентифікований користувач може додавати книги в бібліотеку
K4	Користувач	Редагування книги	Аутентифікований користувач може відредагувати книги в бібліотеці
K5	Користувач	Видалення книги	Аутентифікований користувач може видалити книгу з бібліотеки
K6	Користувач	Створення планування	Аутентифікований користувач може створювати планування свого читання на основі наявних книг в бібліотеці
K7	Користувач	Фіксація результатів	Коли планування створено, користувач може фіксувати результати виконання плану, вводячи дату, книгу і кількість сторінок, які були прочитані
K8	Користувач	Завершення планування	Планування завершується, якщо вийшов час або прочитані всі книги.
K9	Користувач	Вихід з системи	Користувач може вийти з додатку.

Конкретизація варіантів використання

1. Реєстрація в системі

Основна діюча особа: Користувач.

Інші учасники прецеденту: відсутні

Зв'язки з іншими варіантами використання: відсутні

Короткий опис.

- Користувач має право зареєструватись в системі. Є два варіанти реєстрації: на пряму в системі, через обліковий запис в Google. Перший варіант передбачає ведення ім'я користувача, електронної адреси та пароллю. Щоб пройти верифікацію необхідно повторно ввести пароль.
- Щоб зареєструватись через Google на формі реєстрації присутня кнопка, яка перенаправить користувача на відповідно форму Google, де йому буде запропоновано увійти в систему, використовуючи один з доступних облікових записів.
- Після реєстрації користувач буде автоматично перенаправлений на сторінку бібліотеки.

2. Вхід у систему

Основна діюча особа: Користувач.

Інші учасники прецеденту: відсутні

Зв'язки з іншими варіантами використання: доступний лише після реєстрації

Короткий опис.

- Якщо користувач зареєстрований і попередньо вийшов з додатку у нього є можливість увійти в систему. Якщо користувач реєструвався на пряму в системі, йому буде доступна форма в яку він має ввести електронну адресу та пароль. Якщо користувач реєструвався за допомогою облікового запису Google, йому необхідно натиснути на відповідну кнопку на формі та увійти, вибравши певний обліковий запис.
- Після входу користувач буде автоматично перенаправлений на сторінку бібліотеки.

3. Додавання книги

Основна діюча особа: Користувач.

Інші учасники прецеденту: відсутні

Зв'язки з іншими варіантами використання: доступний лише після входу в систему

Короткий опис

- Користувач має можливість додавати необмежену кількість книг. Для додавання книги необхідно відкрити форму. У форму необхідно ввести такі данні: обкладинка книги, назва книги, автор, кількість сторінок і рік випуску. Також присутні додаткові поля: чи була книга прочитана та скільки разів.
- Книги діляться на три списки по статусу: «прочитані», «читаю», «планую прочитати». Якщо при додаванні книги не вказувати, що вона була прочитана, книга автоматично з'явиться в списку «планую прочитати».

4. Редагування книги

Основна діюча особа: Користувач.

Інші учасники прецеденту: відсутні

Зв'язки з іншими варіантами використання: доступний лише після входу в систему та при наявності в бібліотеці хоча б однієї книги.

Короткий опис.

- При натисканні на книгу відкривається повна інформація про неї. Також на цій сторінці є можливість відредагувати дані. Користувач може відредагувати обкладинку книги, назву, автора, кількість сторінок та рік випуску.
- Якщо книга вже прочитана, то є можливість відредагувати її рейтинг та відгук до неї.

5. Видалення книги

Основна діюча особа: Користувач.

Інші учасники прецеденту: відсутні

Зв'язки з іншими варіантами використання: доступний лише після входу в систему та при наявності в бібліотеці хоча б однієї книги.

Короткий опис.

- Книги, які знаходяться в списках «прочитані» та «планую прочитати» можна видаляти. Книги, які знаходяться в списку «читають» видалити не можливо, це буде доступно лише після закінчення читання книги і при умові, що дана книга не додана в дійсне планування.

6. Створення планування

Основна діюча особа: Користувач.

Інші учасники прецеденту: відсутні

Зв'язки з іншими варіантами використання: доступний лише після входу в систему та при наявності в бібліотеці хоча б однієї книги.

Короткий опис

- В додатку є можливість створити планування читання. Для створення тренування необхідно ввести дату початку та дату кінця планування, також вибрати з бібліотеки книги, які планується прочитати. Книги доступні тільки з списків «прочитані» та «планую прочитати».
- Після того, як всі данні вказані, користувач натискає відповідну кнопку початку.
- Коли планування розпочате, користувач бачить перед собою графік, на якому є дві криві: перша- план(кількість сторінок, які користувач має прочитати в кожен день, щоб досягти бажаного результату), друга- фактичний результат користувача. Також на сторінці доступна секція «Моя мета прочитати», на якій вказані кількість книжок, які заплановані, кількість днів та кількість книжок, які залишилось прочитати. Секція результати містить форму для вводу статистики та список, у якому вказана дата, книга та кількість сторінок, які були прочитані в цей день.
- У верхній частині сторінки є два таймера: до закінчення року та до досягнення мети

7. Фіксація результатів

Основна діюча особа: Користувач.

Інші учасники прецеденту: відсутні

Зв'язки з іншими варіантами використання: доступний лише після входу в систему та при наявності дійсного планування.

Короткий опис

- В секції «Статистика» є форма, яка дозволяє ввести результат читання. Щоб додати статистику необхідно додати дату (дати доступні в діапазоні від старту планування до поточної дати), обрати книгу, яку користувач читав, та вказати кількість прочитаних сторінок. Можна вводити декілька даних на одну дату, якщо інформація буде про одну книгу, то сторінки будуть додаватись.

8. Завершення планування

Основна діюча особа: Користувач.

Інші учасники прецеденту: відсутні

Зв'язки з іншими варіантами використання: доступний лише після входу в систему та при наявності дійсного планування.

Короткий опис

- Планування завершиться в двох випадках: якщо всі книги прочитані(статус завершеного планування буде «успішно») або вийшов час(статус завершеного планування буде «не успішно»).

9. Вихід з системи

Основна діюча особа: Користувач.

Інші учасники прецеденту: відсутні

Зв'язки з іншими варіантами використання: доступний лише після входу в систему.

Короткий опис

- У верхній частині додатку є кнопка «Вихід». При натисканні на неї, з'явиться модальне вікно, яке попередить користувача про те, що не

збережені дані будуть видалені. Користувач може підтвердити і вийти з системи, після чого його перекине на сторінку входу, або скасувати вихід.

Шляхом аналізу функціональних вимог було виявлено ключові елементи, які необхідні для розробки програмного комплексу для даної системи. На рисунку 2.2 представлена діаграма класів, яка відображає контролери системи.

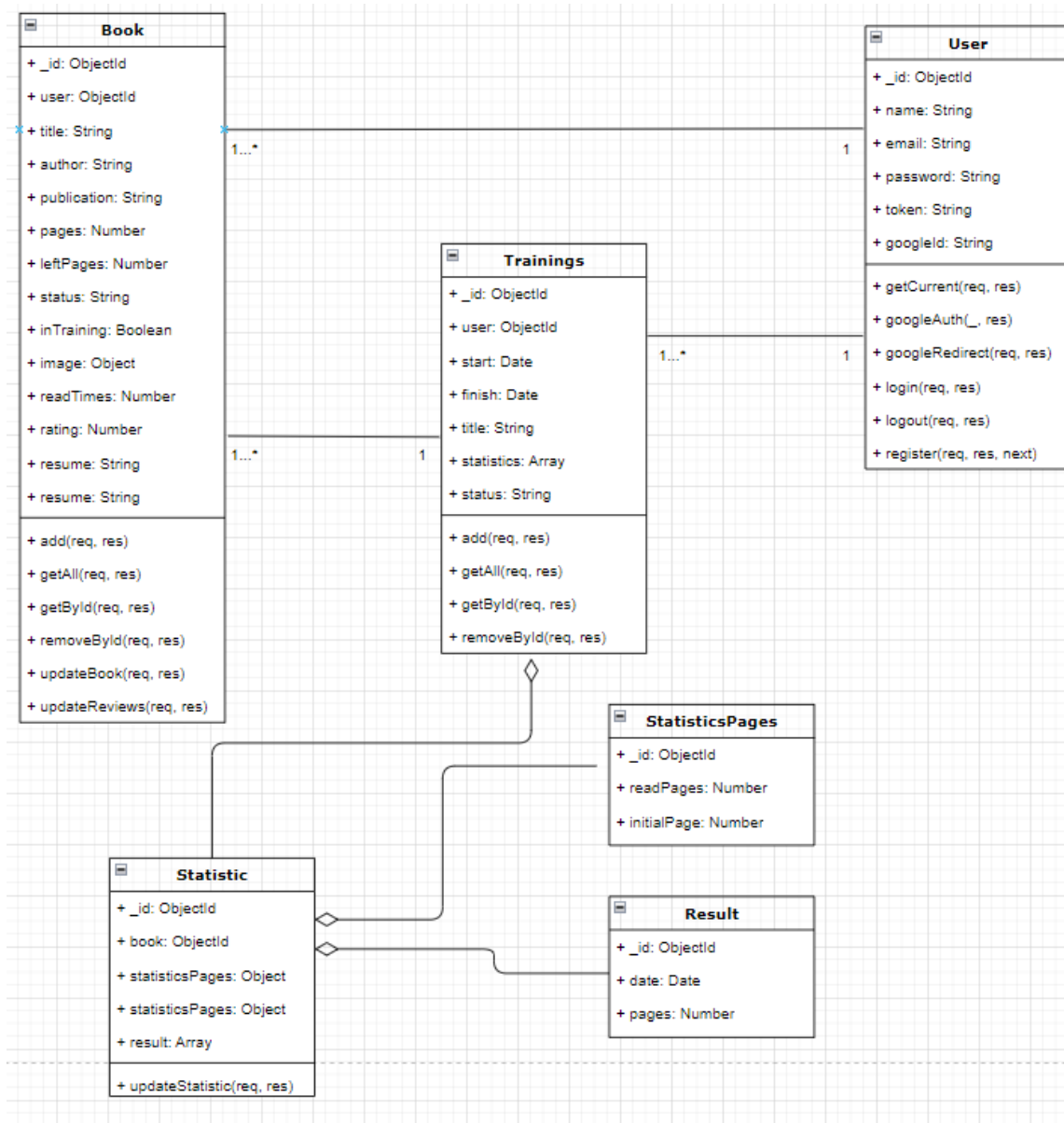


Рисунок 2.2. Діаграма класів

На даному рисунку є наступні класи:

User – клас, який містить методи для роботи з користувачем.

Поля:

- **_id** (тип ObjectId) – унікальний ідентифікатор;

- name (тип String) – ім'я користувача
 - email (тип String) – логін у вигляді пошти для реєстрації в системі
 - password (тип String) – пароль користувача
 - token (тип String) - символічний елемент інформації, який використовується для підтвердження ідентифікації користувача або додатку.
 - googleId (тип String) – ідентифікатор для авторизації через Google
- Методи:

- getCurrent(req, res) – отримати поточного користувача
- googleAuth(req, res) – авторизація через Google
- googleRedirect(req, res) – перенаправлення на автентифікацію Google
- login(req, res) – вхід в систему
- logout(req, res) – вихід з системи
- register(req, res) – реєстрація в системі

Book – клас, який містить методи для роботи з книгою

Поля:

- _id(ObjectId) – унікальний ідентифікатор
- user(ObjectId) – унікальний ідентифікатор користувача
- title(String) – назва книги
- author(String) – автор книги
- publication(String) – дата публікації
- pages(Number) – кількість сторінок
- leftPages(Number) – прочитані сторінки в книзі
- status(String) – статус книги («планую прочитати», «читаю», «прочитано»)
- inTraining(Boolean) – чи знаходиться книга в тренуванні
- image(Object) – фотографія обкладинки книги
- readTimes(Number) – кількість прочитаних разів
- rating(Number) – оцінка книги
- resume(String) – відгук по книзі

Методи:

- add(req, res) – додавання книги
- getAll(req, res) – отримати всі книги
- getById(req, res) – отримати книгу по ідентифікатору
- removeById(req, res) – видалити книгу по ідентифікатору
- updateBook(req, res) – оновити дані книги
- updateReviews(req, res) - оновити відгук

Training – клас, який містить методи для роботи з метою читання

Поля:

- _id(ObjectId) – унікальний ідентифікатор
- user(ObjectId) – унікальний ідентифікатор користувача
- start(Date) – початок мети читання
- finish(Date) – кінець мети читання
- title(String) – назва мети тренування
- statistics(Array) – статистика
- status(String) – статус мети тренування

Методи:

- add(req, res) – додати мету тренування
- getAll(req, res) – отримати всі планування читання
- getById(req, res) – отримати планування читання книг по ідентифікатору
- removeById(req, res) – видалити планування читання

Statistics – клас, який містить методи для роботи з метою статистики

Поля:

- _id(ObjectId) – унікальний ідентифікатор
- book(ObjectId) – унікальний ідентифікатор книги
- statisticsPages(Object) – статистика прочитаних сторінок
- result(Object) – результат виконання мети читання

Методи:

- updateStatistic(req, res) – оновити результати читання

StatisticsPages - клас, який містить поля статистики прочитаних книг

Поля:

- `_id(ObjectId)` - унікальний ідентифікатор
- `readPages(Number)` – прочитаних сторінок книги
- `initialPages(Number)` – початкова кількість прочитаних сторінок

Result - клас, який містить прогрес читання

- `_id(ObjectId)` - унікальний ідентифікатор
- `date(Date)` – дата внесення прогресу
- `pages(Number)` – кількість прочитаних сторінок за певний день

2.2 Розробка бази даних системи

Для зберігання файлів та інформації у моєму додатку для контролю читання книг я використовую базу даних MongoDB, керовану за допомогою інструментів розробки. Ця NoSQL база даних дозволяє ефективно зберігати та організовувати дані без реляційної моделі.

Структура бази даних мого додатку включає чотирьох колекцій:

1. books – таблиця, яка зберігає інформацію про книги;
2. collections – таблиця, яка зберігає інформацію про колекції;
3. trainings – таблиця, яка зберігає інформацію про тренування;
4. users – таблиця, яка зберігає інформацію про користувача;

Опис кожної з таблиць бази даних наведений у таблицях 2.3 – 2.7, де детально розглянуті всі поля, типи даних, обмеження та зв'язки між таблицями. Це дозволяє чітко зрозуміти структуру бази даних та її функціональні можливості.

Таблиця 2.3

Таблица "books"

Поле	Тип	Призначення
_id	ObjectId	Ідентифікатор книжки
user	ObjectId	Ідентифікатор користувача, якому належить книга
title	String	Назва книги

author	String	Ім'я автора
publication	String	Дата публікації книги
pages	Int32	Кількість сторінок в книзі
leftPages	Int32	Кількість сторінок, які залишилось прочитати
status	String	Статус книги («читаю», «планую прочитати», «прочитано»)
inTraining	Boolean	Чи знаходить книга зараз в якомусь активному тренуванні
image	Object	Об'єкт з інформацією про картинку(public_id – ідентифікатор в cloudinary, url: посилання на картинку)
readTimes	Int32	Кількість прочитаних разів
collections	Array	Масив колекцій, який складається з ідентифікаторів колекцій

Таблиця 2.4

Таблиця "collections"

Поле	Тип	Призначення
_id	ObjectId	Ідентифікатор колекції
user	ObjectId	Ідентифікатор користувача, якому належить колекція
name	String	Назва колекції

Таблиця 2.5

Таблиця "trainings"

Поле	Тип	Призначення
_id	ObjectId	Ідентифікатор тренування

user	ObjectId	Ідентифікатор користувача, якому належить тренування
start	Date	Дата початку тренування
finish	Date	Дата завершення тренування
title	String	Назва тренування
statistics	Array	Масив інформації про книжки в тренування. Від складається з об'єктів в яких зберігається інформація про ідентифікатор книги, к-сть прочитаних сторінок, к-сть сторінок які потрібно прочитати та масив результатів (масив об'єктів, який зберігає інформацію про те, скільки сторінок даної книги і в який день було прочитано)
status	String	Статус тренування («Планується», «Активне», «Завершине»)

Таблиця 2.6

Таблиця "users"

Поле	Тип	Призначення
_id	ObjectId	Ідентифікатор користувача
name	String	Ім'я користувача
email	String	Електронна пошта користувача
password	String null	Пароль користувача зашифрований за допомогою бібліотеки bcryptjs. (Якщо користувач реєструється і входить в систему за допомогою Google, то в цілях безпеки це поле буде null)

token	String	Токен JWT. Генерується для розпізнавання користувача.
googleID	String null	Ідентифікатор користувача в Google. Якщо користувач входить в систему вводячи електронну пошту та пароль, то це поле буде null
createdAt	Date	Дата створення користувача
updatedAt	DATE	Дата оновлення користувача

1. Реєстрація та Авторизація Користувача

Таблиця Users (Користувачі): Користувач реєструється на платформі, заповнюючи форму, яка містить поля «name», «email» та «password». Сервер перевіряє, чи існує користувач із такою ж електронною адресою в базі даних MongoDB. Якщо користувач із цією електронною адресою вже існує, буде повернено повідомлення про помилку. Якщо користувача не знайдено, сервер шифрує пароль для безпеки за допомогою бібліотеки bcryptjs. Потім у базі даних створюється новий запис із полями _id, name, email, password (хешований пароль), token (null), googleID (null), createdAt (поточна дата) і updatedAt (поточна дата). Після успішної реєстрації сервер надсилає користувачеві підтвердження реєстрації.

Під час авторизації користувач вказує адресу електронної пошти та пароль у формі входу. Сервер перевіряє, чи існує в базі даних користувач із наданою адресою електронної пошти. Якщо користувача знайдено, сервер порівнює введений пароль із зашифрованим паролем у базі даних за допомогою bcryptjs. Якщо паролі збігаються, сервер генерує JWT, що містить ідентифікатор користувача (_id) та іншу необхідну інформацію (наприклад час дії токenu). Запис користувача в базі даних оновлюється шляхом додавання згенерованого JWT до поля маркера, а також оновлюється поле updatedAt. Сервер надсилає користувачеві маркер JWT, який зберігається на стороні клієнта і використовується для автентифікації в наступних запитах.

Після реєстрації та авторизації Google користувач вибирає опцію входу Google і проходить автентифікацію за допомогою Google OAuth2.0. Після успішної автентифікації Google повертає інформацію користувача, включаючи його ідентифікатор Google, ім'я та електронну адресу. Сервер перевіряє, чи існує в базі даних користувач із таким ідентифікатором Google. Якщо користувача не знайдено, у базі даних створюється новий запис із полями `_id`, `name`, `email`, `googleID`, `createAt` і `updateAt`, але поле пароля залишається пустим. Маркер JWT генерується та зберігається в полі маркера. Якщо користувача знайдено, його запис оновлюється, включаючи JWT і поле `updateAt`. Сервер надсилає JWT користувачеві для подальшої автентифікації.

2. Додавання книги

Таблиця Books (Книги): Коли користувач додає книгу до програми, його ідентифікатор використовується для пов'язування книги з певним користувачем. Спочатку система перевіряє, чи існує книга з такою ж назвою та автором у бібліотеці користувача. Якщо така книга вже існує, користувач отримує повідомлення про помилку.

Якщо надано зображення книги, воно буде завантажено на платформу Clouinary, а збережені посилання та ідентифікатор зображення буде додано до книги. Якщо зображення не надано, використовуватиметься зображення за умовчанням.

Потім у базі даних MongoDB створюється новий запис книги. Запис містить ідентифікатор користувача, назву книги, ім'я автора, дату публікації, кількість сторінок, кількість сторінок, що залишилися для читання, статус книги (наприклад, «читаю», «планую прочитати», «прочитано»), інформацію про те, чи це книга в активному навчанні, об'єкт з інформацією про зображення, скільки разів книга була прочитана та масив з ідентифікаторами колекції, до якої належить книга. Якщо запис створено успішно, користувач отримує підтвердження про додавання книги.

3. Створення тренування

Таблиця Trainings (Тренування): Коли користувач створює нову програму читання книг, система виконує кілька важливих кроків, щоб забезпечити достовірність і цілісність даних. Спочатку перевіряється дійсність наданих ідентифікаторів книг, у тому числі ідентифікаторів повторного читання. Якщо будь-який із цих ідентифікаторів недійсний, користувач отримує повідомлення про помилку.

Потім для кожної книги, зазначеної в навчанні, система перевіряє, чи книга не включена вже в інше навчання. Якщо книга вже знаходиться в активному навчанні, користувач отримує відповідне повідомлення. Потім перевіряються дати початку та завершення навчання, щоб переконатися, що дата завершення є пізнішою за дату початку.

Потім система створює запис про навчання в базі даних MongoDB, де зберігається інформація про користувача, дата початку та закінчення навчання, назва навчання, статус навчання та таблиця з інформацією про книги, які охоплює навчання. Кожен елемент масиву статистики зберігає ідентифікатор книги, кількість прочитаних сторінок, кількість сторінок, які потрібно прочитати, і табло, яке показує, скільки сторінок було прочитано щодня.

Потім система оновлює дані для кожної книги, включеної в навчання, позначаючи її як участь у навчанні та скидаючи кількість прочитаних сторінок до нуля, якщо необхідно. Якщо будь-яке оновлення книги не вдається, тренування буде видалено, а користувач отримає повідомлення про помилку. Після успішного створення та оновлення даних користувач отримує підтвердження успішного створення тренування з усією необхідною інформацією.

4. Редагування книги

Таблиця Books: Процес редагування книги в додатку виглядає наступним чином. Спочатку користувач надсилає запит на редагування книги, надаючи ідентифікатор книги та свої облікові дані. Система перевіряє, чи існує книга з таким ідентифікатором, що належить даному користувачеві. Якщо книга не знайдена, користувач отримає повідомлення про помилку.

Потім система перевіряє, чи не змінилося зображення книги. Якщо нове зображення відрізняється від поточного, система завантажує нове зображення в Cloudinary, використовуючи натомість ідентифікатор попереднього зображення. Після успішного завантаження нового зображення нова URL-адреса та ідентифікатор зображення будуть збережені.

Потім система створює оновлений об'єкт книги, який містить усі нові дані, включаючи зображення. Книга оновлюється в базі даних MongoDB новими даними. Якщо оновлення пройде успішно, користувач отримає підтвердження з новою інформацією про книгу. Якщо під час оновлення виникне помилка, користувач отримає внутрішнє повідомлення про помилку сервера з проханням перевірити введені дані.

5. Додавання статистики в тренуванні

Таблиця Trainings (Тренування): Процес додавання статистики до вашого навчання в додатку виглядає наступним чином. Користувач надсилає запит на оновлення статистики навчання, вказуючи свій ID, номер книги, кількість прочитаних сторінок і дату. Система перевіряє, чи існує навчання з таким ідентифікатором, що належить цьому користувачеві, і чи входить зазначена книга до навчання. Якщо тренування або книга не знайдено, користувач отримає повідомлення про помилку.

Потім система оновлює кількість сторінок книги, прочитаних під час навчання. Якщо загальна кількість прочитаних сторінок перевищує кількість сторінок у книзі, вона береться за загальну кількість сторінок. Оновлені дані зберігаються в тренінгу, а новий запис із датою та кількістю прочитаних сторінок додається до таблиці лідерів.

Потім система оновлює інформацію про книгу в базі даних. Кількість прочитаних сторінок збільшується, і якщо вона дорівнює загальній кількості сторінок у книзі, статус книги змінюється на «прочитана», книга знімається з активного навчання, а кількість прочитаних книг збільшується по одному. Оновлені дані зберігаються в базі даних.

Нарешті, система перевіряє стан навчання за допомогою функції оновлення статусу навчання. Якщо новий статус відрізняється від попереднього, він оновлюється в базі даних. Після завершення всіх оновлень користувач отримує підтвердження з оновленими даними навчання.

2.3 Проектування та реалізація алгоритмів роботи системи

Як основний алгоритм, було вирішено використовувати звичайний алгоритм односторінкового додатку (SPA) для клієнтської частини системи (див. Рисунок 2.3). При переході на сторінку додатку, сервер надсилає користувачеві актуальну версію сторінки (клієнтську частину). Потім, залежно від дій користувача, клієнт надсилає запити на сервер. Після отримання відповіді від серверу, клієнт змінює вигляд сторінки.



Рисунок 2.3. Загальний алгоритм роботи додатку

Склавши загальний алгоритм додатку, далі необхідно продумати алгоритм для окремих випадків. Для представлення подібних сценаріїв зазвичай використовуються діаграми діяльності (activity diagram). Даною діаграмою можна додатково роз'яснити деякі дії у додатку.

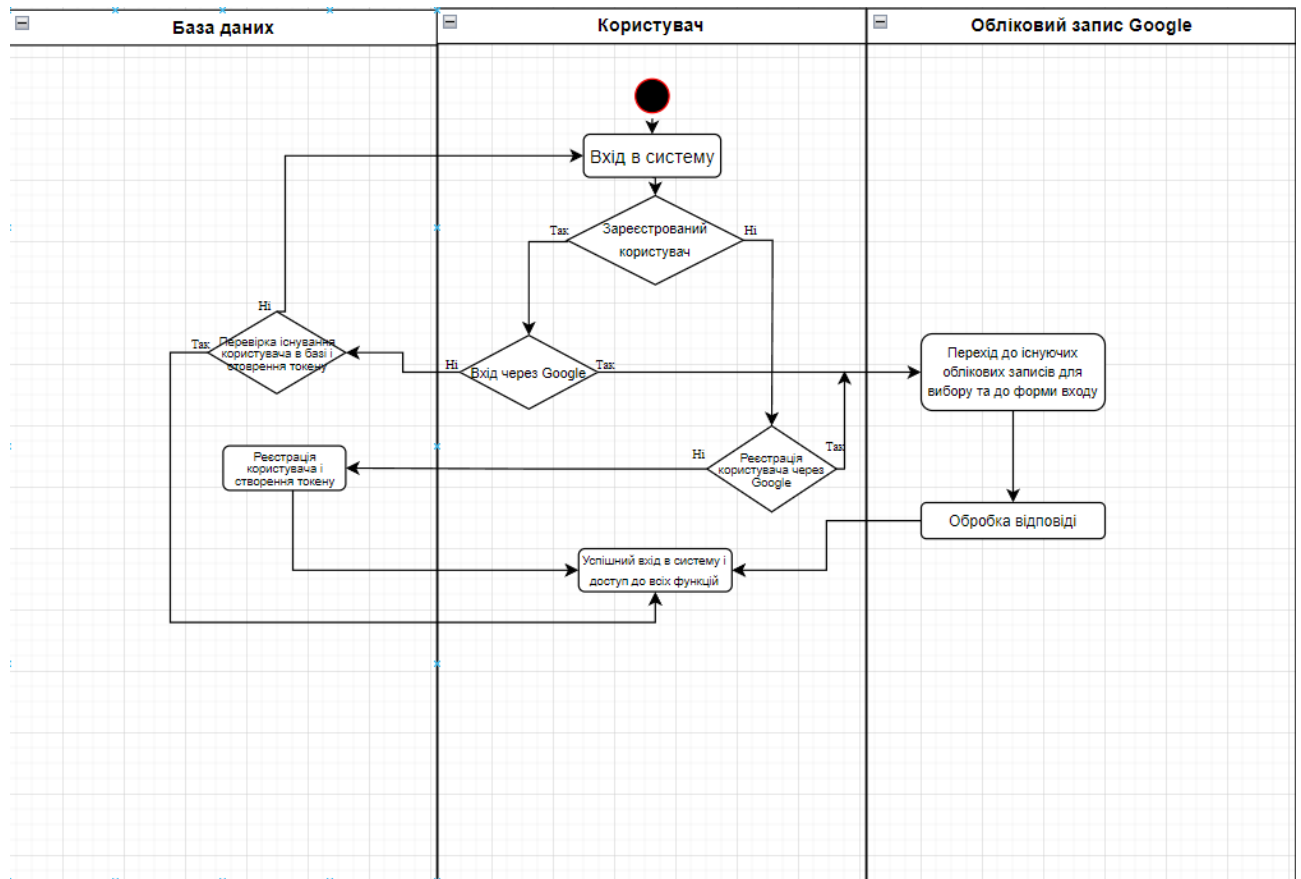


Рисунок 2.4. Діаграма діяльності «Авторизація користувача»

Процес авторизації веб-додатку є важливою складовою для забезпечення безпеки та конфіденційності користувачів. Розглянемо процес авторизації:

1. Сторінка Авторизації

- Користувач потрапляє на сторінку авторизації при переході за відповідним посиланням веб-додатку.
- Якщо користувач вже зареєстрований у системі, він має дві можливості для входу: за допомогою логіна та пароля або через обліковий запис Google.

2. Реєстрація нового користувача

- Якщо користувач не зареєстрований в системі, він може перейти за відповідним посиланням для реєстрації.

- Користувачу відкривається нова форма для реєстрації. Він має можливість вибрати метод реєстрації: через логін та пароль або через обліковий запис Google.

3. Авторизація за допомогою логіну та пароля:

- Користувач вводить свій логін та пароль.
- Фронтенд виконує валідацію введених даних і виводить повідомлення про помилку, якщо дані не відповідають вимогам.
- Якщо дані валідні, фронтенд надсилає запит на бекенд.
- На бекенді перевіряється наявність користувача з введеним логіном та паролем в базі даних.
- Якщо користувач знайдений, на бекенді генерується токен для аутентифікації.
- Токен надсилається на фронтенд.
- Фронтенд обробляє отриманий токен та відбувається вхід користувача в систему.

4. Реєстрація нового користувача:

- Користувач вводить свій логін, пароль та ім'я на сторінці реєстрації та натискає кнопку "Зареєструватися".
- Фронтенд валідує введені дані, перевіряючи їх на відповідність вимогам.
- Якщо введені дані вважаються валідними, фронтенд надсилає запит на бекенд.
- На бекенді перевіряються дані на додаткові вимоги та додавання користувача в базу даних.
- Після успішної реєстрації користувача створюється токен аутентифікації та надсилається на фронтенд.
- Фронтенд обробляє отриману відповідь і виконує вхід нового користувача в систему.

5. Процес входу та реєстрації через обліковий запис Google:

- Користувач обирає варіант "Увійти через Google" на сторінці авторизації.

- Веб-додаток відправляє запит на авторизацію через Google до серверів Google за допомогою Google OAuth 2.0.
- Google вимагає від користувача підтвердження доступу до облікового запису. Користувач повинен увійти в свій обліковий запис Google та надати дозвіл на доступ до певних інформаційних ресурсів.
- Після успішного входу і надання дозволу, Google генерує токен доступу, який передається веб-додатку.
- Веб-додаток отримує токен доступу від Google і використовує його для запиту інформації про користувача з серверів Google, такої як ім'я та електронна адреса.
- Веб-додаток перевіряє, чи існує користувач з вказаною електронною адресою у системі. Якщо такий користувач вже існує, то він автоматично входить в систему.
- Якщо користувача з такою електронною адресою не знайдено, веб-додаток може запропонувати користувачу зареєструватися в системі, використовуючи отримані дані від Google.
- Після входу або реєстрації користувача в системі, йому створюється токен аутентифікації для подальшого доступу до системи.

2.4 Реалізація функціоналу застосунку для контролю читання книг

Перший етап для користувача – це авторизація в систему. На цьому етапі користувач може увійти в систему, якщо він вже зареєстрований або перейти на сторінку реєстрації

Регістрація:

```
const register = async (req, res, next) => {
  const { email, password, name } = req.body;
  const user = await User.findOne({ email });
  if (user) {
    throw new Conflict(`User with ${email} already exist`);
  }
  const hashPassword = bcrypt.hashSync(password, bcrypt.genSaltSync(10));
  await User.create({
    name,
    email,
```

```

    password: hashPassword,
  });
  res.status(201);
  next();
};

```

Перевіряє, чи існує користувач з вказаною електронною адресою. Якщо користувача не знайдено, створюється новий обліковий запис. Пароль користувача хешується за допомогою bcrypt та зберігається в базі даних.

Вхід:

```

const login = async (req, res) => {
  const { email, password } = req.body;
  const user = await User.findOne({ email });
  if (!user) {
    throw new Unauthorized(`Email or password is wrong or email isn't verify`);
  }
  const passCompare = bcrypt.compareSync(password, user.password);
  if (!passCompare) {
    if (user.password === "" && user.googleId) {
      throw new Unauthorized(`Please, use Google to login!`);
    }
    throw new Unauthorized(`Password is wrong`);
  }
  const payload = {
    id: user._id,
  };
  const token = jwt.sign(payload, SECRET_KEY, { expiresIn: "3h" });
  await User.findByIdAndUpdate(user._id, { token });
  const { name } = user;
  res.status(200).json({
    message: "success",
    code: 200,
    data: {
      token,
      user: {
        name,
        email,
      },
    },
  });
};

```

Перевіряє, чи існує користувач із зазначеною електронною адресою. Якщо користувач не знайдений або пароль введено неправильно, буде виведено відповідну помилку. Якщо введено неправильний пароль, але користувач авторизований Google, з'явиться повідомлення про те, що авторизацію Google застосовано. Якщо авторизація пройшла успішно, генерується JWT, який

використовується для доступу до захищених ресурсів і зберігається в базі даних користувача. Маркер і дані користувача повертаються.

Авторизація через Google:

```
const googleAuth = async (_, res) => {
  const stringifiedParams = queryString.stringify({
    client_id: process.env.GOOGLE_CLIENT_ID,
    redirect_uri: `${process.env.BASE_URL}/api/auth/google-redirect`,
    scope: [
      "https://www.googleapis.com/auth/userinfo.email",
      "https://www.googleapis.com/auth/userinfo.profile",
    ].join(" "),
    response_type: "code",
    access_type: "offline",
    prompt: "consent",
  });
  return res.redirect(
    `https://accounts.google.com/o/oauth2/v2/auth?${stringifiedParams}`
  );
};
```

Створює URL-адресу для авторизації через Google OAuth2. Після натискання цього посилання користувач перенаправляється на сторінку авторизації Google, де дає згоду на доступ до своїх даних.

```
const googleRedirect = async (req, res) => {
  const code = req.query.code;
  const tokenData = await axios({
    url: `https://oauth2.googleapis.com/token`,
    method: "post",
    data: {
      client_id: process.env.GOOGLE_CLIENT_ID,
      client_secret: process.env.GOOGLE_CLIENT_SECRET,
      redirect_uri: `${process.env.BASE_URL}/api/auth/google-redirect`,
      grant_type: "authorization_code",
      code,
    },
  });
  const userData = await axios({
    url: "https://www.googleapis.com/oauth2/v2/userinfo",
    method: "get",
    headers: { Authorization: `Bearer ${tokenData.data.access_token}` },
  });

  const { id: googleId, name, email } = userData.data;
  let user = await User.findOne({ email });
  if (!user) {
    user = await User.create({
      name,
      email,
      password: "",
    });
  }
};
```

```

    });
  }
  const payload = {
    id: user._id,
  };
  const token = jwt.sign(payload, process.env.SECRET_KEY, { expiresIn: "3h" });
  await User.findByIdAndUpdate(user._id, { token, googleId });
  return res.redirect(
    `${process.env.FRONTEND_URL}/google-
    redirect/?token=${token}&name=${user.name}&email=${user.email}`
  );
};

```

Обробляє зворотній виклик після авторизації через Google. Отримує доступ до даних користувача, отриманих від Google, таких як ім'я, електронна адреса та ідентифікатор Google. Перевіряє, чи існує користувач з вказаною електронною адресою. Якщо користувач не знайдено, створюється новий обліковий запис. Генерується JWT-токен, який використовується для доступу до захищених ресурсів, і зберігається в базі даних користувача. Повертається токен та дані користувача.

Вхідний інтерфейс надає можливість користувачам авторизуватися в системі(код наведений в Додатку А). Форма входу дозволяє ввести адресу електронної пошти та пароль, забезпечуючи зручний та безпечний доступ до програми. Крім того, інтерфейс пропонує альтернативну можливість входу за допомогою облікового запису Google, що спрощує процес авторизації для користувачів.

Перед авторизацією введені користувачем дані перевіряються. Перевірка поля «Адреса електронної пошти» забезпечує відповідність стандартам формату електронної пошти, включаючи мінімальну та максимальну довжину, починаючи з літери або цифри, і дозволені символи. Поле «Пароль» також перевіряється на відповідність критеріям мінімальної та максимальної довжини, без пробілів і починається з літери або цифри.

Якщо аутентифікація пройшла успішно, користувач отримує доступ до основних функцій програми. Для нових користувачів доступна інформація та посилання про те, як зареєструватися.

яВикористання Redux:

Застосунок використовує бібліотеку Redux для керування станом додатку. Redux - це популярна бібліотека для керування станом додатків у JavaScript, що забезпечує простий та передбачуваний спосіб керування станом.

```
export const booksApi = createApi({
  reducerPath: 'booksRTK',
  baseQuery: axiosBaseQuery({
    // baseUrl: 'https://reading-marathons-backend.onrender.com/api',
    baseUrl: 'http://localhost:3001/api'
  }),
  tagTypes: ['Books', 'BookById', 'Trainings', 'TrainingById'],
  endpoints: builder => ({
    getBooks: builder.query({
      query: () => ({ url: '/books', method: 'GET' }),
      providesTags: ['Books'],
    }),
    addBook: builder.mutation({
      query: values => ({ url: '/books', method: 'POST', data: values }),
      invalidatesTags: ['Books'],
    }),
    deleteBook: builder.mutation({
      query: id => ({ url: `/books/${id}`, method: 'DELETE' }),
      invalidatesTags: ['Books', 'Trainings', 'TrainingById'],
    }),
    getBookById: builder.query({
      query: id => ({ url: `/books/${id}`, method: 'GET' }),
      providesTags: ['BookById'],
    }),
    updateBookReview: builder.mutation({
      query: params => ({
        url: `/books/${params.id}/review`,
        method: 'PATCH',
        data: params.data,
      }),
      invalidatesTags: ['Books', 'BookById'],
    }),
    updateBook: builder.mutation({
      query: params => ({
        url: `/books/${params.id}/update`,
        method: 'PATCH',
        data: params.data,
      }),
      invalidatesTags: ['Books', 'BookById'],
    }),
    getTrainings: builder.query({
      query: () => ({ url: '/trainings', method: 'GET' }),
      providesTags: ['Trainings'],
    }),
  }),
  // Інші операції для роботи з тренуваннями та колекціями
```

```

    }),
  });
  export const {
    useGetBooksQuery,
    useAddBookMutation,
    useDeleteBookMutation,
    useGetBookByIdQuery,
    useUpdateBookReviewMutation,
    useUpdateBookMutation,
    useGetTrainingsQuery,
    useGetTrainingByIdQuery,
    ...
  } = booksApi;

```

Цей код визначає API для взаємодії з сервером. Він використовує Redux Toolkit для створення API шляхом використання createApi функції.

Кожна операція має свою функцію, яку можна викликати, щоб зробити відповідний запит до сервера. Крім того, API забезпечує автоматичне визнання кешованих даних недійсними, коли дані оновлюються або видаляються за допомогою визначених тегів. Це забезпечує послідовність даних і оптимізує використання ресурсів.

Висновки до другого розділу

У другому розділі документації розглянуто процес розробки застосунку для контролю читання книг. Починаючи від аналізу вимог бізнесу та користувачів і закінчуючи впровадженням функціональних можливостей, у центрі уваги було створення зручного та ефективного продукту. Важливим етапом став аналіз варіантів використання, який дозволив створити оптимізоване середовище для моніторингу читання книг.

Було детально розглянуто реєстр варіантів використання та побудовано діаграму класів, яка дозволила чітко визначити структуру системи та функціональність кожного з її компонентів. Окремо розглянуто розробку бази даних та проектування алгоритмів системи, які сприяли оптимізації продуктивності та ефективності додатку.

Таким чином, заповнення цього розділу підтверджує успішну реалізацію функціональності програми контролю читання книг. Він відповідає вимогам і

очікуванням користувачів, забезпечуючи комфорт і ефективність використання. Крім того, продукт готовий до подальшої розробки та впровадження, що робить його придатним для використання в реальних умовах.

					ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						54

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ЗАСТОСУНКОМ ДЛЯ КОНТРОЛЮ ЧИТАННЯ КНИГ

3.1 Порядок встановлення та налаштування параметрів сайту

При налаштуванні сайту для контролю читання книг необхідно реалізувати всю систему з усіма її компонентами та налаштувати окремі елементи.

Опис процесу впровадження:

Клієнтська частина (Frontend): користувачі взаємодіють із веб-сайтом через веб-браузер, де інтерфейс користувача реалізовано за допомогою сучасних веб-технологій, таких як бібліотека React разом з JavaScript та Styled Components. Використання цих технологій забезпечує зручну та інтерактивну взаємодію з сайтом, що підвищує комфорт використання.

Серверна частина (Backend): серверна частина програми використовується для обробки запитів від клієнтів. У нашому випадку JavaScript використовується за допомогою фреймворку Node.js для реалізації бізнес-логіки, обробки запитів на аутентифікацію, роботи з книгами та інших функцій.

База даних: реляційна база даних використовується для зберігання даних про користувачів, книги, читання та іншу необхідну інформацію. У нашому випадку використовувався MongoDB, який забезпечує ефективне зберігання та роботу з даними.

Середовище реалізації: Додаток реалізовано на віртуальному сервері, що забезпечує стабільну та безпечну роботу системи.

Конфігураційні файли .env: Окремі файли .env використовуються для зберігання конфіденційних даних, таких як ключі доступу до бази даних, паролі та інші параметри, які не підлягають версійному контролю та забезпечують безпечне зберігання конфіденційних даних.

Архітектура системи:

1. Клієнтська частина:
 - React: бібліотека створення інтерфейсу, яка дозволяє ефективно керувати станом програми та забезпечує реактивне оновлення інтерфейсу.

- JavaScript: використовується для написання логіки для взаємодії з елементами інтерфейсу, такими як обробники подій і перевірка даних.
- Стилiзовані компоненти: дозволяє стилізувати компоненти безпосередньо у коді React, забезпечуючи зручний і безпечний для користувача код.

2. Серверна частина:

- Node.js: використовується для створення серверної частини програми.
- Express.js: фреймворк для Node.js, який дозволяє створювати веб-сервери та обробляти запити клієнтів.
- MongoDB: база даних NoSQL, яка забезпечує зберігання даних схеми документа та використовується для зберігання інформації про користувача, книгу та замовлення.

3. Взаємодія між компонентами:

- Клієнтська частина взаємодіє з серверною через HTTP-запити та відповіді.
- Серверна частина обробляє ці запити, виконує необхідні дії (наприклад, авторизація користувача, обробка платежу) і повертає відповідь.
- Взаємодія з базою даних MongoDB здійснюється через драйвери бази даних MongoDB для Node.js.
- JWT (веб-токени JSON) використовується для цілей безпеки та автентифікації.

Ця архітектура дозволяє побудувати ефективний і масштабований веб-додаток для моніторингу читання книг, забезпечуючи користувачеві зручну та безпечну роботу.

3.2 Структура інтерфейсу сайту з контролю читання книг

У цьому розділі буде представлений огляд інтерфейсу сайту з контролю читання книг. Аналіз базується на скріншотах інтерфейсу користувача, що відображають ключові елементи та функціональні можливості програмного продукту. Кожен скріншот супроводжуватиметься коротким описом, що підкреслює основні аспекти взаємодії користувачів з сайтом.

Користувача, який відкриває програму вперше, зустрічає форма входу, яка складається з кількох ключових елементів. По-перше, він має кнопку входу в Google, яка дозволяє вам швидко та безпечно увійти до свого облікового запису Google. Далі слідує форма входу, де користувач може ввести свій логін (адресу електронної пошти) і пароль. Ця форма взаємодіє з базою даних, щоб перевірити точність введених даних. Якщо користувач ще не має облікового запису, під кнопкою входу він може знайти посилання на сторінку реєстрації, де він може зареєструватися та створити обліковий запис, щоб продовжити використання програми. (Рисунок 3.1.)

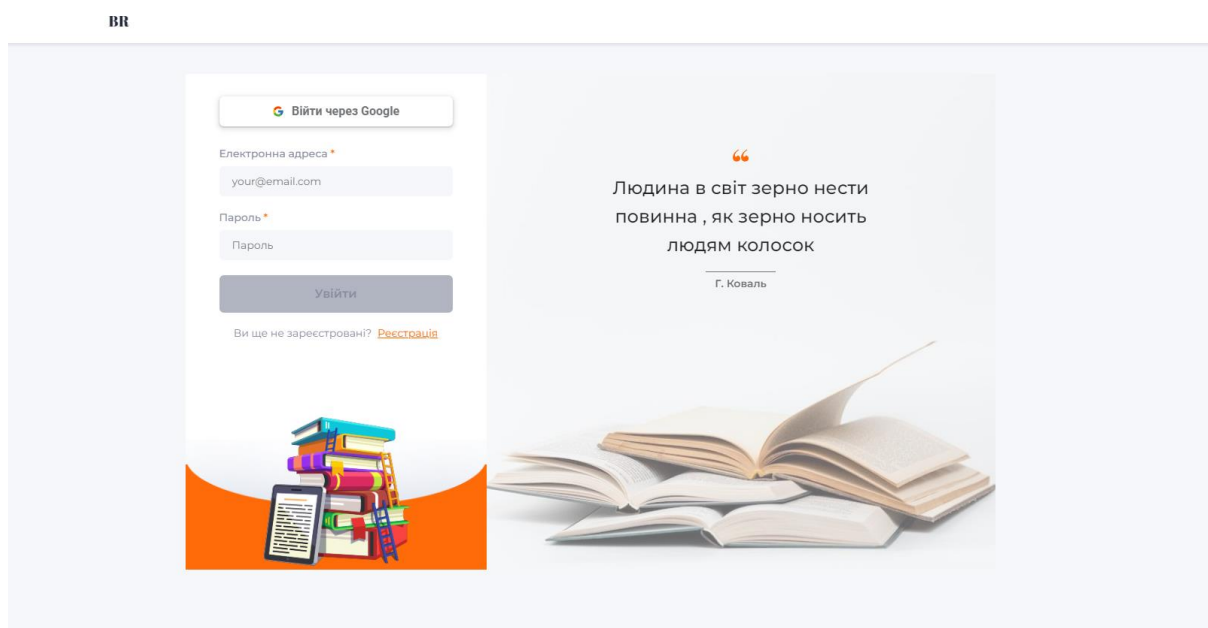


Рисунок 3.1. Сторінка входу в систему

Після переходу на сторінку реєстрації в розпорядженні користувача є два основних елементи. По-перше, є кнопка реєстрації в Google, яка дозволяє швидко та безпечно створити обліковий запис за допомогою облікового запису Google. По-друге, реєстраційна форма, яка містить поля для введення імені, електронної адреси, пароля та підтвердження пароля. Крім того, під кнопкою реєстрації користувач може знайти посилання на сторінку входу, якщо у нього вже є обліковий запис. (Рисунок 3.2.)

Для забезпечення правильності введених даних використовується схема перевірки. Ця схема перевіряє, чи дані, які ви вводите, відповідають певним

BR

Рисунок 3.2. Сторінка реєстрації

У верхній частині сторінки є пошуковий рядок, який дозволяє швидко знайти потрібні книги. Нижче ви знайдете вкладки, розділені на категорії "Маю намір прочитати", "Читаю" та "Прочитано", які групують книги залежно від їх статусу. Якщо в якійсь з категорій немає книг, відповідна вкладка буде неактивною.

Нижче наведено список книг, кожна з яких представлена у вигляді картки з інформацією про обкладинку, назву, автора, рік видання, кількість сторінок і прогрес читання. Загальна кількість прочитаних сторінок вказана у верхньому правому куті картки. Якщо книга вже прочитана, на картці також відображається оцінка книги за шкалою від 1 до 5 зірок. Крім того, є кнопка, яка відкриває модальне вікно, де користувач може додавати та редагувати відгук. (Рисунок 3.4.)

На цій же сторінці є кнопка додавання нової книги, яка відкриває модальне вікно з формою додавання книги. Цей інтерфейс забезпечує зручний і ефективний спосіб ведення особистої бібліотеки та взаємодії з книжковими матеріалами.

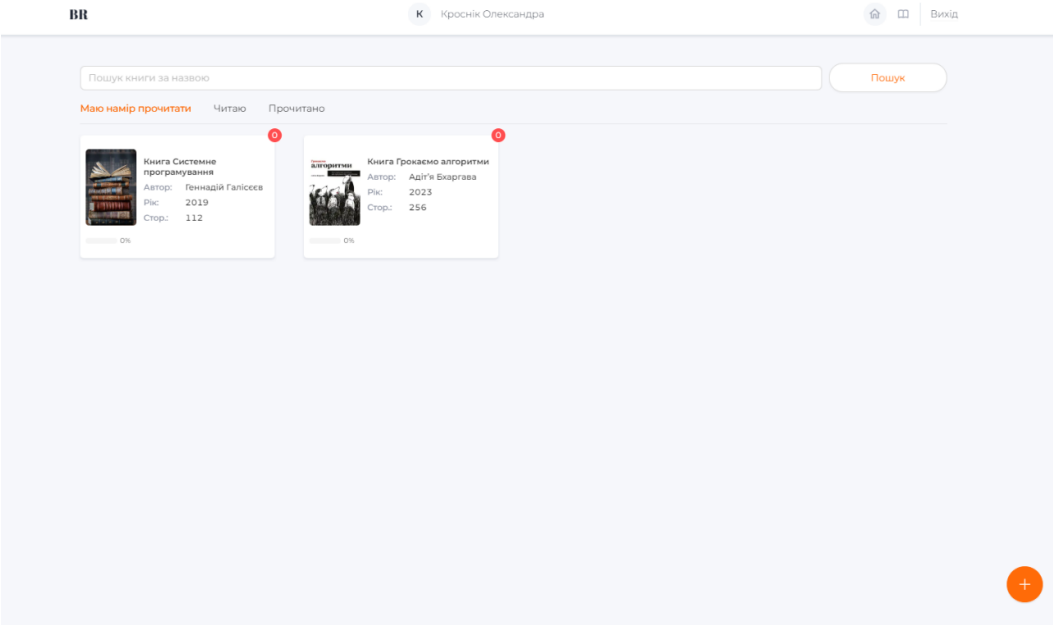


Рисунок 3.3. Сторінка бібліотеки користувача

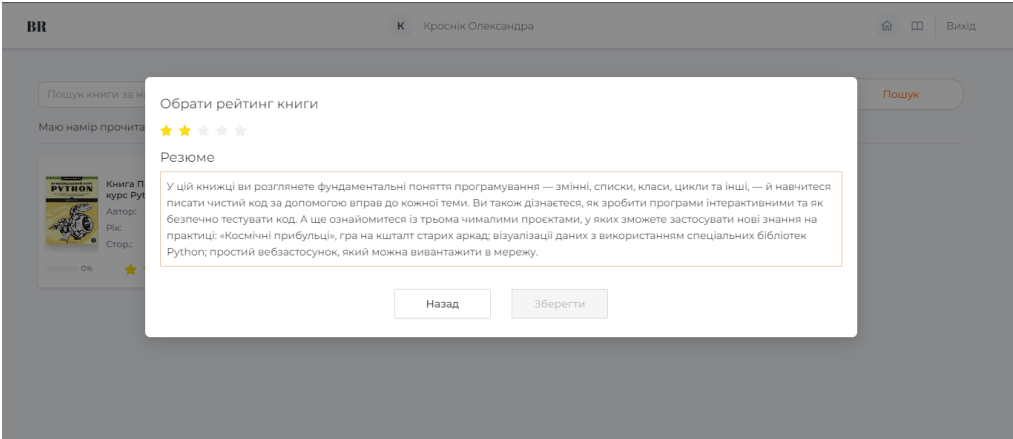


Рисунок 3.4. Форма редагування та додавання відгуку книги

Форма додавання книги в систему надає користувачам можливість вводити різноманітну інформацію про нові книги для подальшого зберігання та організації. Вона містить такі поля: обкладинка книги (з обмеженням розміру файлу), назва книги, рік видання, колекції, до яких вона належить, ім'я автора, кількість сторінок і статус книги (прочитана чи ні). Якщо книга була прочитана, можна вказати, скільки разів вона була прочитана. Ця форма спрощує процес оновлення бібліотеки та дозволяє зберігати важливі дані книги для подальшого використання. (Рисунок 3.5.)

←



Назва книги

Рік випуску

Колекція

Статус книги

☐ Прочитана

☒ Не прочитана

Автор книги

Кількість сторінок

Кількість прочитаних разів

Додати

Рисунок 3.5. Форма додавання книги в бібліотеку

При натисканні на книгу в системі відкривається додаткова інформація про неї, що розташовується справа від списку книг на екрані. У випадку, якщо користувач використовує мобільну версію застосунку, ця інформація з'явиться знизу екрану. Це забезпечує зручний доступ до детальної інформації про книгу без перешкоджання основному вмісту сторінки. Користувач може швидко переглянути ключові відомості про книгу, такі як обкладинка, назва, автор, рік видання та кількість сторінок. (Рисунок 3.6.)

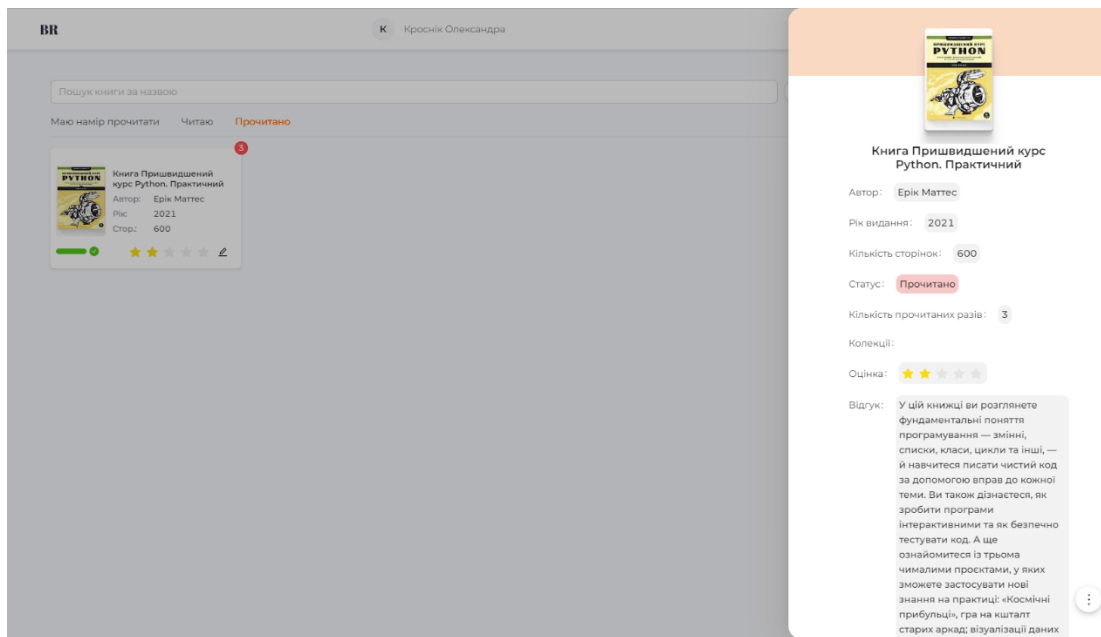


Рисунок 3.6. Відображення інформації про книгу

В нижній частині сторінки присутня кнопка, яка надає можливість користувачеві видалити конкретну книгу зі списку. При видаленні книги, вона автоматично видаляється з усіх тренувань, пов'язаних з нею. Також є функція редагування інформації про книгу, яка дозволяє змінювати обкладинку, назву, рік видання, кількість сторінок і статус прочитаності. Форма редагування має такий самий вигляд і поля для заповнення, що й форма для додавання нової книги, надаючи користувачеві зручний і уніфікований інтерфейс для взаємодії з даними. (Рисунок 3.7.)

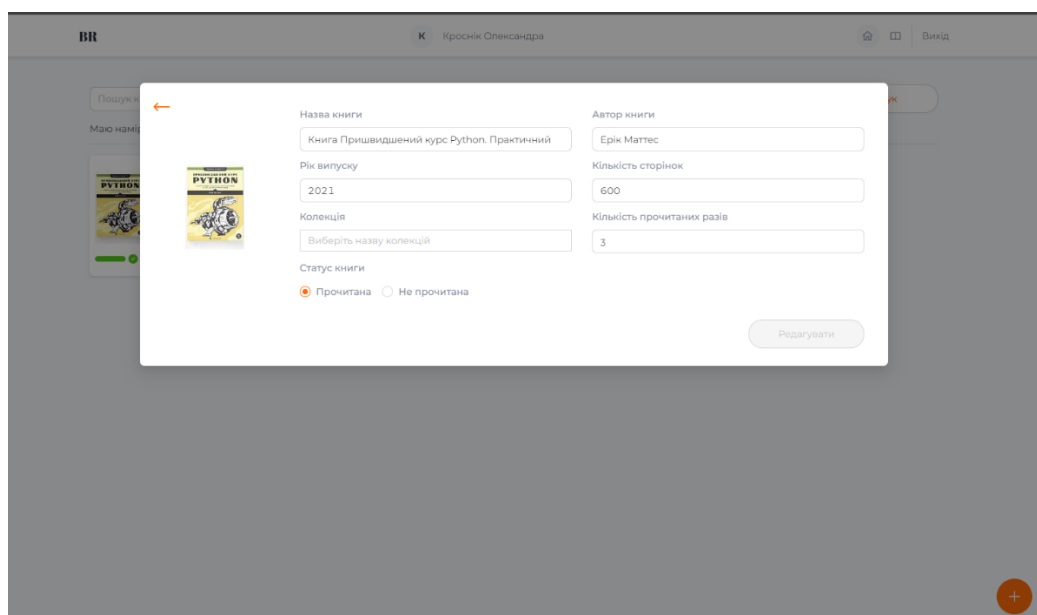


Рисунок 3.7. Редагування інформації про книгу

У хедері навігація складається з двох кнопок: кнопки "Додому", яка перенаправляє користувача на головну сторінку бібліотеки, та кнопки "Відкриті книги", яка веде на сторінку тренувань. Після переходу на сторінку тренувань у верхній частині доступна кнопка "Додати ціль", яка дозволяє користувачеві створювати нові тренування. Далі на сторінці розташований список тренувань, розділений на категорії за їх статусом: "Триваючі", "Заплановані" та "Завершені". Якщо тренувань з певним статусом немає, відповідний розділ не відображається.

Карточка кожного тренування містить наступну інформацію: назву книги, перелік книг, які призначені для даного тренування, статус кожної книги, який відображений у вигляді кругового графіку, що показує відсоток прочитаних сторінок. Також вказана дата початку та завершення тренування. Остання колонка містить інформацію про статус тренування: якщо тренування триває, вона називається "Прогрес" та показує відсоток виконання тренування; якщо тренування заплановане, називається "Початок через" та вказує кількість часу до початку; якщо тренування завершене, вона може мати статус "Успішне" або "Не виконане", а також показує, яку частину, відсотково, було виконано. (Рисунок 3.8.)

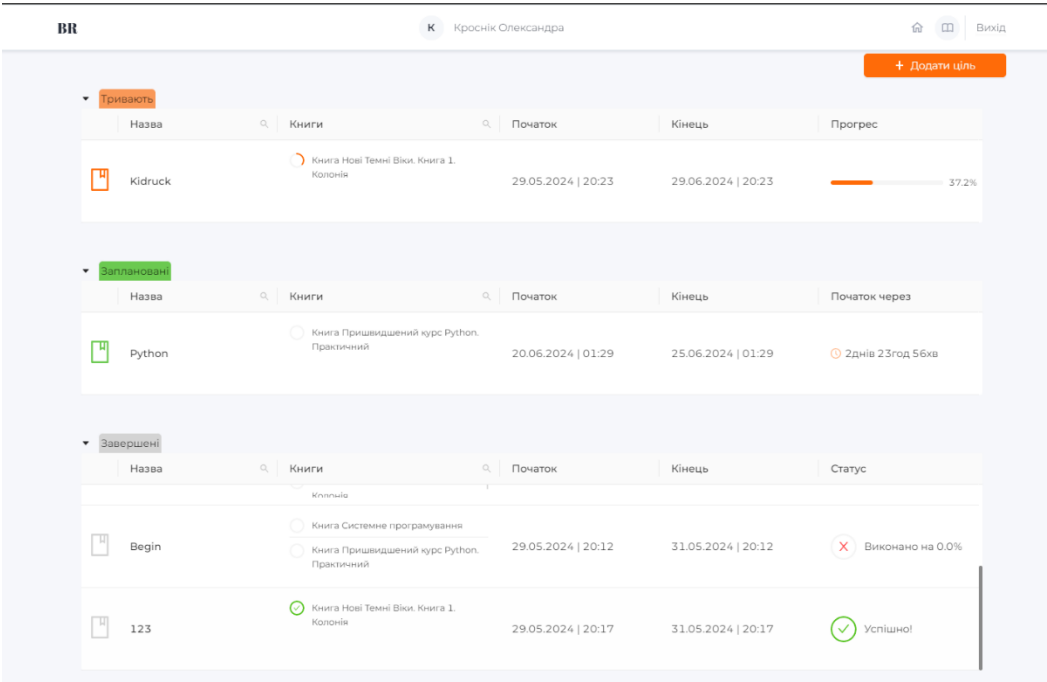


Рисунок 3.8. Сторінка тренувань

При переході на сторінку додавання тренування користувачу буде доступно вказати дату початку та дату кінця тренування. Дата початку може бути вибрана відповідно до поточної дати, а дата завершення не може бути раніше, ніж дата початку. Далі користувач має можливість обрати книги, які будуть доступні для тренування. Якщо книга вже знаходиться в активному тренуванні або запланована на майбутнє, вона не буде включена до списку доступних книг. Для додавання книги до тренування користувачу потрібно натиснути відповідну кнопку. (Рисунок 3.9.)

Рисунок 3.9. Сторінка формування тренування

Після додавання книг список оновлюється, а також формується графік читання та справа відображається інформація про кількість книг та кількість сторінок. Після натискання кнопки "Почати тренування" відкриється форма. Якщо серед вибраних книг є ті, які вже розпочали читатись, але не були повністю прочитані, користувач може вибрати опцію "Продовжити читання" або "Скасувати". Після вибору всіх налаштувань користувач повинен підтвердити свій вибір, вказавши назву тренування, та після цього тренування розпочнеться. (Рисунок 3.10.)

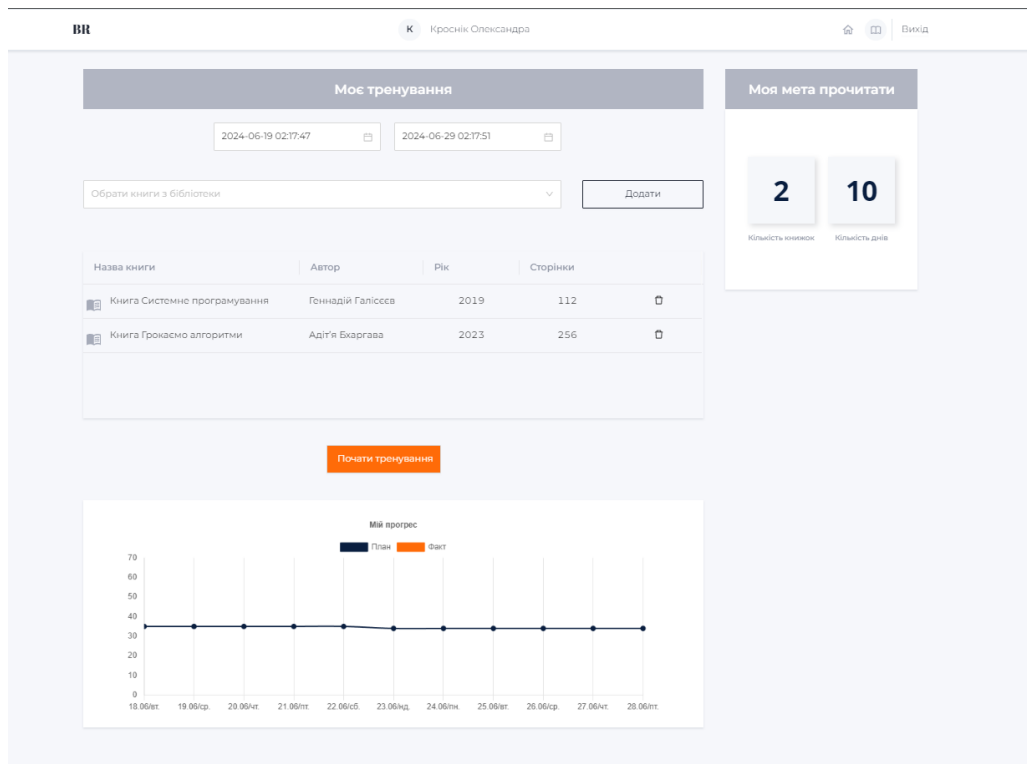


Рисунок 3.10. Сторінка тренування готового до додавання

Коли тренування має статус "Завершено", користувач може переглянути інформацію про нього, але немає можливості додати нові дані чи редагувати існуючі. Ця сторінка призначена виключно для перегляду результатів тренування. (Рисунок 3.11.)

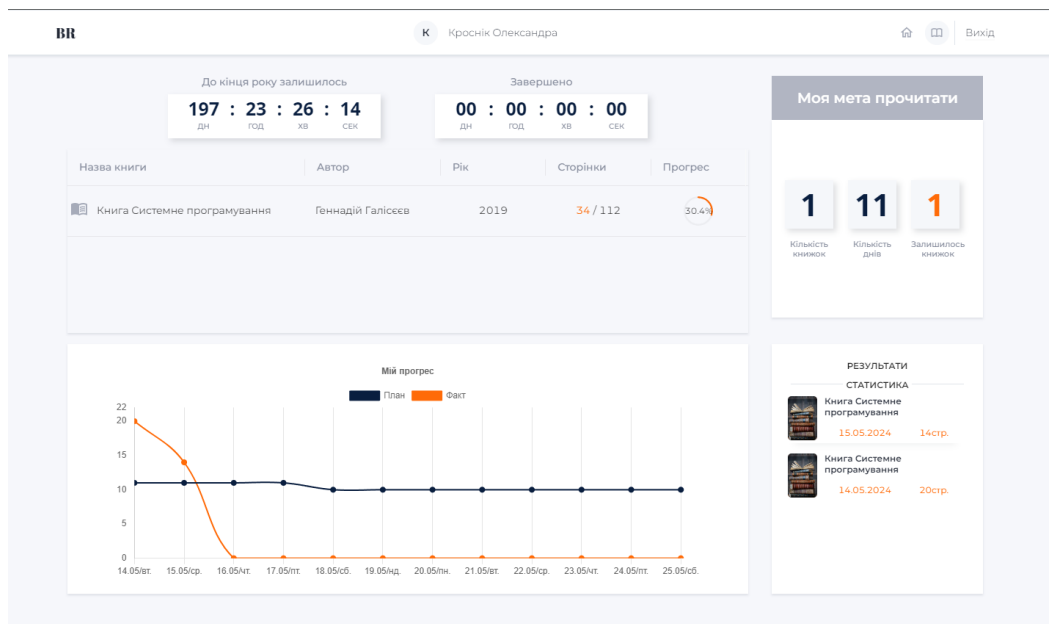


Рисунок 3.11. Сторінка тренування зі статусом «Завершено»

У випадку тренування зі статусом "Заплановане", у верхній частині сторінки відображається таймер, який показує час до початку тренування. У

блоці з результатами тренувань немає жодних даних або форм для додавання інформації, оскільки тренування ще не розпочалося. (Рисунок 3.12.)

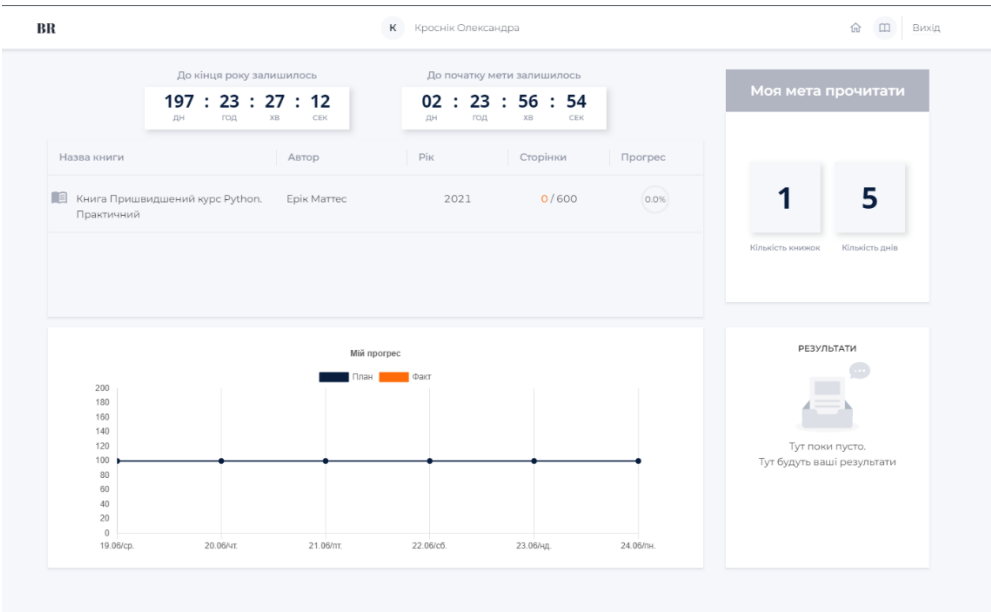


Рисунок 3.12. Сторінка тренування зі статусом «Заплановане»

Коли тренування активне, у верхній частині сторінки відображається таймер до завершення тренування. Також доступна форма для введення результатів, яка містить поля для обрання книги та дати (доступна лише в діапазоні від дати початку тренування до поточної дати) та введення кількості прочитаних сторінок. Ця форма дозволяє користувачеві вносити дані про читання під час тренування. (Рисунок 3.13.)

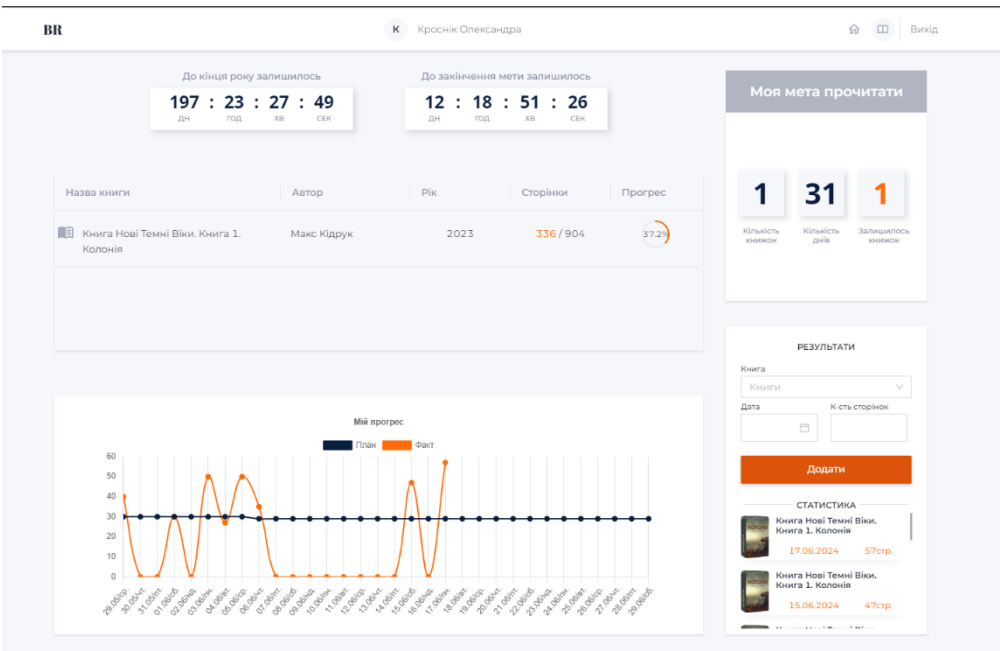


Рисунок 3.13. Сторінка тренування зі статусом «Тривають»

3.3 Тестування сайту

Тестування веб-сайту, призначеного для контролю читання книг, є важливим кроком у забезпеченні його надійності, функціональності та зручності для користувачів. Процес тестування включає кілька основних аспектів, таких як функціональне тестування, тестування продуктивності та тестування зручності використання.

Функціональне тестування: під час функціонального тестування перевіряється, чи працюють усі функції системи, пов'язані з контролем читання книг. Це включає перевірку того, що користувачі можуть увійти та зареєструватися, додавати та видаляти книги, редагувати тренування та перевіряти, чи правильно відображаються статуси книг і тренувань.

Тестування зручності використання: мета цього тесту — оцінити, наскільки легко користувачі можуть взаємодіяти з системою керування читанням книг. Оцінюється інтуїтивність і зрозумілість інтерфейсу, зручність навігації між сторінками і наявність основних функцій.

Тестування продуктивності: цей тест перевіряє швидкість роботи системи при роботі з великими обсягами даних. Оцінюється час відкриття сторінок, завантаження списків книг і тренувань, а також швидкість виконання різних операцій.

Тестування безпеки: мета цього тестування — перевірити систему на вразливість системи безпеки та захистити її від потенційних загроз. Це включає перевірку витоків конфіденційної інформації, захоплення сеансу, а також забезпечення перевірки даних і захисту від SQL-ін'єкцій і атак XSS.

Регресійне тестування: регресійне тестування перевіряє, чи з'являються нові помилки під час внесення змін або додавання нових функцій до системи. Особливу увагу приділено впливу нових змін на існуючий функціонал.

Кросбраузерне та кросплатформне тестування: цей тест перевіряє сумісність системи з різними веб-браузерами (Chrome, Firefox, Safari, Edge) та операційними системами (Windows, MacOS, Android, iOS).

Юзабіліті тестування: Метою тестування зручності використання є оцінка того, наскільки легко користувачі можуть виконувати різні завдання в системі керування читанням книг. Перевіряються простота навігації, ясність і зрозумілість інструкцій, а також наявність допоміжних функцій (наприклад, пошук або фільтрація).

Кожен із цих типів тестів допоможе підтвердити, що система контролю читання книг відповідає вимогам щодо функціональності, зручності використання, безпеки та продуктивності.

Висновки до третього розділу

У розділі, присвяченому структурі інтерфейсу сайту контролю читання книг, наведено детальний огляд основних компонентів і функцій програмного забезпечення. Аналіз базується на скріншотах інтерфейсу користувача, які демонструють ключові елементи та можливості системи. Кожен скріншот супроводжується коротким описом, що висвітлює основні аспекти взаємодії користувача з сайтом.

Перші етапи взаємодії з веб-сайтом включають процеси входу та реєстрації користувача. Завдяки інтуїтивно зрозумілому інтерфейсу користувачі можуть швидко та зручно отримати доступ до системи.

Далі було детально розглянуто основні елементи інтерфейсу бібліотеки, зокрема розподіл книг за статусами «Я маю намір прочитати», «Я читаю» та «Я буду читати». Ця структура дозволяє користувачам ефективно керувати своїми читацькими уподобаннями та відстежувати свій прогрес у читанні.

На основі аналізу інтерфейсу також встановлено, що система забезпечує можливість зручного додавання та редагування книг у бібліотеці. Завдяки інтуїтивно зрозумілому процесу додавання та редагування користувачі можуть легко керувати своїми списками для читання та тренуваннями. Усі ці функції та можливості сприяють створенню зручного й ефективного середовища керування читанням книг, яке відповідає потребам і вимогам користувачів.

ВИСНОВКИ

У цій дипломній роботі реалізовано веб-застосунок спрямований на створення онлайн-платформи, яка дозволить користувачам ефективно контролювати те, як вони читають книги. В основі проекту лежить використання передових веб-технологій і принципів дизайну для створення зручного та інтуїтивно зрозумілого інтерфейсу.

У першому розділі дипломної роботи детально розглядаються вимоги до програмного продукту. Аналізуються потреби користувачів і функції, які необхідно реалізувати для задоволення цих потреб. Цей етап є ключовим для подальшого розвитку, оскільки вся архітектура продукту будується на основі цих вимог.

Друга частина присвячена технічній реалізації проекту. Він включає вибір технології, проектування архітектури системи, розвиток функціональності та взаємодію між компонентами. Цей етап вимагає ретельного планування та організації, щоб забезпечити ефективну інтеграцію всіх компонентів продукту.

Третій розділ визначає дизайн і функціональність програмного забезпечення. Описує інтерфейс користувача, його ключові елементи та взаємодію з користувачем. Надання зручного та інтуїтивно зрозумілого інтерфейсу є одним із головних завдань цього розділу.

Підсумовуючи, робота показує, що використання сучасних веб-технологій та відповідний проектний підхід дозволяють створити ефективну та функціональну платформу контролю читання книг. Такий підхід сприяє задоволенню потреб користувачів і досягненню передбачуваних цілей проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. HTML Documentation [Електронний ресурс] - Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/HTML>
2. CSS Documentation [Електронний ресурс] - Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/CSS>
3. JavaScript Documentation [Електронний ресурс] - Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
4. Як створити обліковий запис Google-Обліковий запис Google Довідка. GoogleHelp:<https://support.google.com/accounts/answer/27441?hl=uk> (дата звернення: 17.06.2024).
5. W3Schools HTML Tutorial [Електронний ресурс] - Режим доступу до ресурсу: <https://www.w3schools.com/html/>
6. W3Schools CSS Tutorial [Електронний ресурс] - Режим доступу до ресурсу: <https://www.w3schools.com/css/>
7. W3Schools JavaScript Tutorial [Електронний ресурс] - Режим доступу до ресурсу: <https://www.w3schools.com/js/>
8. Stack Overflow [Електронний ресурс] - Режим доступу до ресурсу: <https://stackoverflow.com/>
9. Figma [Електронний ресурс] - Режим доступу до ресурсу: <https://www.figma.com/>
10. ReactJS Documentation [Електронний ресурс] - Режим доступу до ресурсу: <https://reactjs.org/docs/getting-started.html>
11. Node.js Documentation [Електронний ресурс] - Режим доступу до ресурсу: <https://nodejs.org/en/docs/>
12. Express [Електронний ресурс] - це фреймворк для веб-застосунків, побудованих на Node.js. Express - Node.js web application framework. URL: <https://expressjs.com/uk/>
13. JWT.ІО. [Електронний ресурс] JSON Web Tokens - jwt.io. URL: <https://jwt.io/>

14. bcryptjs. [Электронный ресурс] - npm.
<https://www.npmjs.com/package/bcryptjs>
15. Ant Design of React - Ant Design. Ant Design [Электронный ресурс] - The world's second most popular React UI framework.
URL: <https://ant.design/docs/react/introduce>
16. Getting Started | Redux Toolkit. Redux Toolkit | Redux Toolkit. [Электронный ресурс] URL: <https://redux-toolkit.js.org/introduction/getting-started>
17. Getting Started with Redux | Redux. Redux [Электронный ресурс] - A JS library for predictable and maintainable global state management | Redux.
URL: <https://redux.js.org/introduction/getting-started>
18. Joi.dev. joi.dev. [Электронный ресурс]
URL: <https://joi.dev/api/?v=17.13.0>
19. Yup. npm. [Электронный ресурс] URL: <https://npmjs.com/package/yup>
20. Починаючи | Axios Docs. Axios. [Электронный ресурс]
URL: <https://axios-http.com/uk/docs/intro>
21. React SDK - Image and Video Transformations and Delivery | Cloudinary [Электронный ресурс]. Image and Video Upload, Storage, Optimization and CDN. URL: https://cloudinary.com/documentation/react_integration
22. dayjs. npm. [Электронный ресурс]
URL: <https://www.npmjs.com/package/dayjs>
23. cors. npm. [Электронный ресурс]
URL: <https://www.npmjs.com/package/cors>
24. Duggal A. How to Use PropTypes in React. [Электронный ресурс] freeCodeCamp.org. URL: <https://www.freecodecamp.org/news/how-to-use-proptypes-in-react/>
25. Getting Started with ESLint - ESLint - Pluggable JavaScript Linter. [Электронный ресурс] Find and fix problems in your JavaScript code - ESLint - Pluggable JavaScript Linter. URL: <https://eslint.org/docs/latest/use/getting-started>
26. http-errors. npm. [Электронный ресурс]
URL: <https://www.npmjs.com/package/http-errors>

27. Mongoose v8.4.3: [Электронный ресурс] Schemas. Mongoose ODM v8.4.3. URL: <https://mongoosejs.com/docs/guide.html>

28. React DOM APIs – React. [Электронный ресурс] React. URL: <https://react.dev/reference/react-dom>

29. styled-components: Documentation. [Электронный ресурс] styled-components. URL: <https://styled-components.com/docs>

30. Team M. D. MongoDB Documentation. [Электронный ресурс] MongoDB: The Developer Data Platform | MongoDB. URL: <https://www.mongodb.com/docs/>

ДОДАТКИ

					ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						72

Форма входу користувача

```
import { Formik } from 'formik';
import * as yup from 'yup';
import { FcGoogle } from 'react-icons/fc';
import { useSelector } from 'react-redux';
import { useDispatch } from 'react-redux';
import PulseLoader from 'react-spinners/PulseLoader';

import { authOperations } from 'redux/auth';
import { getIsLoggedIn, getIsPendingState } from 'redux/auth';
import { booksApi } from 'redux/RTKQuery/booksApi';

import {
  FormWrapper,
  StyledForm,
  FieldWrapper,
  FieldName,
  AccentedMark,
  StyledField,
  ValidationError,
  GoogleButton,
  SubmitButton,
  StyledLink,
  FormBottomStyled,
  FormBottomImage,
  LinkWrapper,
  IsRegistredParagraph,
  LoginTextStyled,
} from '../AuthenticationComponent.styled';

const validationSchema = yup.object().shape({
  email: yup
    .string('Enter your email')
    .email('Enter a valid email')
    .min(10, 'Email must contain at least 10 characters')
    .max(63, 'Email must contain no more than 63 characters')
    .matches(/^^[a-zA-Z0-9]/, 'Name must start with letter or number')
```

```

    .matches(
      /^[a-zA-Z0-9.!#$%&'*/+=?^_`{|}~-]{2,})+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/,
      'The Email field can only contain Latin letters, numbers and signs, and at
      least 2 charachters before "@"'
    )
    .required('Email is a required field'),
password: yup
  .string('Enter your password')
  .matches(/^[a-zA-Z0-9]/, 'Password must start with letter or number')
  .matches(
    /^[a-zA-Z0-9@.!\#$%&'*/+=?^_`{|}~-]*$/,
    'Password must not contain spaces'
  )
  .min(5, 'Password is too short - should be 5 chars minimum')
  .max(30, 'Password must contain no more than 30 characters')
  .required('Password is a required field'),
});

const initialValues = {
  email: '',
  password: '',
};

const LoginForm = () => {
  const dispatch = useDispatch();
  const isLoggedIn = useSelector(getIsLoggedIn);
  const isPending = useSelector(getIsPendingState);

  const handleSubmit = (values, actions) => {
    dispatch(
      booksApi.util.invalidateTags([
        { type: 'Books' },
        { type: 'BookById' },
        { type: 'Trainings' },
        { type: 'Statistics' },
      ])
    );

    dispatch(authOperations.logIn(values));
  };

```

```

    isLoggedIn && actions.resetForm();
  };

  return (
    <>
      <FormWrapper>
        <Formik
          initialValues={initialValues}
          validationSchema={validationSchema}
          onSubmit={handleSubmit}
        >
          {({ isValid, touched }) => {
            return (
              <StyledForm name="LoginForm">
                <LoginTextStyled>Вхід</LoginTextStyled>
                <GoogleButton href="http://localhost:3001/api/auth/google">
                  <FcGoogle />
                  Війти через Google
                </GoogleButton>

                <FieldWrapper>
                  <FieldName htmlFor="email">
                    Електронна адреса <AccentedMark>*</AccentedMark>
                  </FieldName>
                  <StyledField
                    id="email"
                    name="email"
                    type="text"
                    placeholder="your@email.com"
                    autoComplete="off"
                  />
                  <ValidationError name="email" component="div" />
                </FieldWrapper>

                <FieldWrapper>
                  <FieldName htmlFor="password">
                    Пароль <AccentedMark>*</AccentedMark>
                  </FieldName>

```

```

        <StyledField
          id="password"
          name="password"
          type="password"
          placeholder="Пароль"
          autoComplete="off"
        />
        <ValidationError name="password" component="div" />
      </FieldWrapper>

      <SubmitButton
        type="submit"
        disabled={!touched.email && !touched.password} || !isValid}
      >
        Увійти
        {isPending && <PulseLoader color="white" size="4px" />}
      </SubmitButton>
      <LinkWrapper>
        <IsRegisteredParagraph>
          Ви ще не зареєстровані?{' '}
        </IsRegisteredParagraph>
        <StyledLink to="/registration">Реєстрація</StyledLink>
      </LinkWrapper>
    </StyledForm>
  );
}}
</Formik>
<FormBottomStyled>
  <FormBottomImage data-status="login" />
</FormBottomStyled>
</FormWrapper>
</>
);
};

export default LoginForm;

```

Реалізація функціоналу сторінки створення нового тренування.

```

import Container from 'components/Container';
import FormAdd from './FormAdd/FormAdd';
import useGoalAdd from './useGoalAdd';
import GoalTable from '../commonComponents/GoalTable/GoalTable';
import BooksListEmptyMobile from '../commonComponents/BooksListEmptyMobile';
import BooksListFilledMobile from '../commonComponents/BooksListFilledMobile';
import StartTrainingButton from './StartTrainingButton';
import ProgressChart from '../commonComponents/ProgressChart/ProgressChart';
import MyGoal from '../commonComponents/MyGoal';
import { StyledContainer } from '../commonGoalStyles.styled';
import ConfirmGoalModal from './ConfirmGoalModal/ConfirmGoalModal';
const GoalAdd = () => {
  const {
    booksForSelect,
    booksForTable,
    setBooksForTable,
    start,
    finish,
    setStart,
    setFinish,
    isMobile,
    isTablet,
    isDesktop,
    submitBooks,
    deleteBookFromTable,
    numberOfBooks,
    numberOfDays,
    isFirstRender,
    getPagesCount,
    getDateCountBetweenDates,
    addButtonDisable,
    initiateBookReadingChallenge,
    confirmModalVisible,
    booksToConfirm,
    onAddGoal,
    setConfirmModalVisible,
  }

```

```

    titleGoal,
    setTitleGoal,
  } = useGoalAdd();
  const addPartContent = (
    <div>
      <FormAdd
        books={booksForSelect}
        selectedBooks={submitBooks}
        startTime={start}
        finishTime={finish}
        setStartTime={setStart}
        setFinishTime={setFinish}
      />
      {!isMobile ? (
        <GoalTable books={booksForTable} onDeleteBook={deleteBookFromTable} />
      ) : (
        <>
          {booksForTable.length === 0 ? (
            <BooksListEmptyMobile />
          ) : (
            <BooksListFilledMobile
              books={booksForTable}
              onClick={setBooksForTable}
            />
          )}
        </>
      )}
      <StartTrainingButton
        htmlType="button"
        onClick={initiateBookReadingChallenge}
        disabled={addButtonDisable}
      />
    </div>
  );
  return (
    <>
      {!isFirstRender && (
        <Container>
          {isDesktop && (

```

```

<StyledContainer>
  {addPartContent}
  <MyGoal days={numberOfDays()} books={numberOfBooks()} />
  <ProgressChart
    planData={{
      pages: getPagesCount(),
      dates: getDateCountBetweenDates(),
    }}
  />
</StyledContainer>
)}
{!isDesktop && (
  <>
    {isTablet && (
      <MyGoal days={numberOfDays()} books={numberOfBooks()} />
    )}
    {addPartContent}
    {isMobile && (
      <MyGoal days={numberOfDays()} books={numberOfBooks()} />
    )}
    <ProgressChart
      planData={{
        pages: getPagesCount(),
        dates: getDateCountBetweenDates(),
      }}
    />
  </>
)}
{confirmModalVisible && (
  <ConfirmGoalModal
    visible={confirmModalVisible}
    books={booksToConfirm}
    onOk={onAddGoal}
    onCancel={setConfirmModalVisible}
    isMobile={isMobile}
    titleGoal={titleGoal}
    setTitleGoal={setTitleGoal}
  />
)}</Container>)) </>));

```

Реалізація функціоналу перевірки чи користувач авторизований

```

const { User } = require("../models");
const { Unauthorized } = require("http-errors");
const jwt = require("jsonwebtoken");

const { SECRET_KEY } = process.env;

const auth = async (req, res, next) => {
  try {
    const { authorization = "" } = req.headers;
    const [bearer, token] = authorization.split(" ");
    if (bearer !== "Bearer") {
      throw Unauthorized("Not authorized");
    }
    const { id } = jwt.verify(token, SECRET_KEY);
    const user = await User.findById(id);
    if (!user || !user.token) {
      throw Unauthorized("Not authorized");
    }
    req.user = user;
    next();
  } catch (error) {
    if (error.message === "invalid signature") {
      error.status = 401;
    }
    next(error);
  }
};

module.exports = auth;

```

Реалізація моделі тренування

```

const { Schema, model } = require("mongoose");
const Joi = require("joi");

const trainingSchema = Schema({
  user: {
    type: Schema.Types.ObjectId,
    ref: "user",
    require: true,
  },
  start: {
    type: Date,
    require: true,
  },
  finish: {
    type: Date,
    require: true,
  },
  title: {
    type: String,
    require: true,
  },
  statistics: [
    {
      book: {
        type: Schema.Types.ObjectId,
        ref: "book",
        require: true,
      },
      statisticsPages: {
        type: {
          readPages: {
            type: Number,
            require: [true, "Please, write pages you have read!"],
          },
          initialPage: {
            type: Number,
            require: [true, "Please, write initial page!"],
          },
        },
      },
      result: {
        type: [
          {
            date: {
              type: Date,
              require: [true, "Please, enter date!"],
            },
            pages: {
              type: Number,
              require: [true, "Please, write pages you have read!"],
            },
          },
        ],
      },
    },
  ],
});

```

```

    },
  ],
  status: {
    type: String,
    require: true,
  },
});

const joiSchema = Joi.object({
  start: Joi.date().required(),
  finish: Joi.date().required(),
  title: Joi.string().required(),

  statistics: Joi.object({
    book: Joi.string(),
    statisticsPages: Joi.object({
      readPages: Joi.number().required().messages({
        "any.required": "Please, write pages you have read!",
        "number.base": "Please provide a valid number for readPages",
      }),
      initialPage: Joi.number().required().messages({
        "any.required": "Please, write initial page!",
        "number.base": "Please provide a valid number for initialPage",
      }),
    }),
    result: Joi.array().items({
      date: Joi.date().required(),
      pages: Joi.number().required(),
    }),
  }),

  status: Joi.string().required(),
});

const joiSchemaAddTraining = Joi.object({
  start: Joi.date().required(),
  finish: Joi.date().required(),
  title: Joi.string().required(),
  books: Joi.array().items(Joi.string()).min(1),
  booksToRestartReading: Joi.array().items(Joi.string()),
});

const joiSchemaAddStatistic = Joi.object({
  book: Joi.string().required(),
  date: Joi.date().required(),
  pages: Joi.number().required(),
});

const Training = model("training", trainingSchema);

module.exports = {
  Training,
  joiSchema,
  joiSchemaAddTraining,
  joiSchemaAddStatistic,
};

```