Полтавський університет економіки і торгівлі
Навчально-науковий інститут заочно-дистанційного навчання
Форма навчання заочна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____Олена Ольховська
«_____»_____2026 р.

КВАЛІФІКАЦІЙНА РОБОТА НА ТЕМУ:

«ПЕРСОНАЛЬНИЙ КНИЖКОВИЙ ТРЕКЕР І РЕКОМЕНДАЦІЙНА СИСТЕМА ДЛЯ ЧИТАЧІВ»

зі спеціальності 122 «Комп'ютерні науки»
освітня програма «Комп'ютерні науки»
ступеня бакалавр

Виконавець роботи Пелих Карина Романівна

_____«__»____2026 р.

(підпис)

Науковий керівник к. ф.-м. н., доц., _____

_____ «__»____2026 р.

(підпис)

Рецензент

ПОЛТАВА 2026

ЗАТВЕРДЖУЮ
Завідувач кафедри _____Олена ОЛЬХОВСЬКА
«_____» _____2026 р.

ЗАВДАННЯ ТА КАЛЕНДАРНИЙ ГРАФІК
ВИКОНАННЯ КВАЛІФАКАЦІЙНОЇ РОБОТИ

на тему «Персональний книжковий трекер і рекомендаційна система для читачів»
зі спеціальності 122 «Комп'ютерні науки»
освітня програма «Комп'ютерні науки»
ступеня бакалавр
Прізвище, ім'я, по батькові Пелих Карина Романівна
Затверджена наказом ректора № -Н від «_____» _____202 __ р.
Термін подання студентом роботи «___» _____ 20__ р.
Вихідні дані до кваліфікаційної роботи: публікації та лекційний матеріал з теми, навчальні тренажери в дистанційних курсах з комп'ютерних наук.
Зміст пояснювальної записки (перелік питань, які потрібно розробити)
ВСТУП
ПОСТАНОВКА ЗАДАЧІ
1. ІНФОРМАЦІЙНИЙ ОГЛЯД
1.1. Сучасні тенденції у створенні веб-сервісів для читання
1.2. Аналіз існуючих рішень
1.3. Вимоги до персонального трекера та рекомендаційної системи
2. ТЕОРЕТИЧНА ЧАСТИНА
2.1. Архітектура платформи та модель взаємодії: FSD, REST API
2.2. Огляд інструментів і технологій
2.3. Функціональна структура та моделювання платформи
3. ПРАКТИЧНА ЧАСТИНА
3.1. Реалізація інтерфейсу платформи UX/UI
3.2. Реалізація функціональної логіки
3.3. Інтеграція з сервером
3.4. Огляд платформи персонального книжкового трекера і рекомендаційної системи для читачів
ВИСНОВОК
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТОК А
Перелік графічного матеріалу: 3-4 аркуші графічного матеріалу, інші необхідні ілюстрації.
Консультанти розділів кваліфікаційної роботи

Розділ	ПІП, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Постановка задачі	Черненко О. О.		
Інформаційний огляд	Черненко О. О.		
Теоретична частина	Черненко О. О.		
Практична реалізація	Черненко О. О.		

Календарний графік виконання кваліфікаційної роботи

Зміст роботи	Термін виконання	Фактичне виконання
1. Вступ		
2. Вивчення методичних рекомендацій та стандартів та звіт керівнику		
3. Постановка задачі		
4. Інформаційний огляд джерел бібліотек та інтернету		
5. Теоретична частина		
6. Практична частина		
7. Закінчення оформлення		
8. Доповідь студента на кафедрі		
9. Доробка (за необхідністю), рецензування		

Дата видачі завдання «____»_____ 202_ р.

Здобувач вищої освіти Пелих Карина

Науковий керівник к. ф.-м. н., доц. Оксана ЧЕРНЕНКО

Результати захисту кваліфікаційної роботи

Кваліфікаційна робота оцінена на

_____

(балів, оцінка за національною шкалою, оцінка за ECTS)

Протокол засідання ЕК № ____ від «____» _____ 202_ р.

Секретар ЕК_____ _____

(підпис) (ініціали та прізвище)

Затверджую
Зав. кафедрою _____
к.ф.-м.н. Олена ОЛЬХОВСЬКА
«____» _____ 202_ р.

Погоджено
Науковий керівник _____
к. ф.-м. н., доц. Оксана ЧЕРНЕНКО
«____» _____ 202_ р.

План

кваліфікаційної роботи ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки
освітня програма 122 Комп'ютерні науки
Пелих Карина Романівна
Прізвище, ім'я, по батькові
на тему «Персональний книжковий трекер і рекомендаційна система для читачів»

ВСТУП
ПОСТАНОВКА ЗАДАЧІ
1. ІНФОРМАЦІЙНИЙ ОГЛЯД
1.1. Сучасні тенденції у створенні веб-сервісів для читання
1.2. Аналіз існуючих рішень
1.3. Вимоги до персонального трекера та рекомендаційної системи
2. ТЕОРЕТИЧНА ЧАСТИНА
2.1. Архітектура платформи та модель взаємодії: FSD, REST API
2.2. Огляд інструментів і технологій
2.3. Функціональна структура та моделювання платформи
3. ПРАКТИЧНА ЧАСТИНА
3.1. Реалізація інтерфейсу платформи UX/UI
3.2. Реалізація функціональної логіки
3.3. Інтеграція з сервером
3.4. Огляд платформи персонального книжкового трекера і рекомендаційної системи для читачів
ВИСНОВОК
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТОК А

Здобувач вищої освіти _____ ПЕЛИХ Карина
«____» _____ 202_ р.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, скорочень, символів
API	Application Programming Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
FSD	Feature-Sliced Design
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
JWT	JSON Web Token
localStorage	Місце у браузері, де можна зберігати дані
REST API	Representational State Transfer API
RTK	Redux Toolkit
SPA	Single Page Application
UX/UI	User Experience / User Interface

ВСТУП

Читання книг залишається важливим елементом для розвитку практично кожної особистості в сучасному світі що є достатньо необхідним, оскільки сприяє розвитку мислення, покращує когнітивні здібності, збагачує словниковий запас та знижує рівень стресу. Крім того, це є одним з інструментів особистісного та професійного зростання, що є досить популярним сьогодні для того щоб бути конкурентоспроможним та освіченою особистістю, що цінується найбільше.

З розвитком технологій кількість споживання інформації зросла відповідно до досліджень кожен день середньостатистична людина обробляє від 34 до 74 GB інформації [1]. Щодня ми стикаємося з проблемою організації читання через те, що маємо доступ до великої кількості інформації також це залежить і від швидкого темпу життя тож іноді важко стежити за власним прогресом: скільки сторінок було прочитано, яку книгу завершено чи яку тільки плануємо читати. Це особливо помітно серед студентів та працівників IT-сектору, які постійно працюють з великим обсягом інформації.

Тому персональний книжковий трекер став невід'ємною частиною життя, адже допомагає бачити обсяг виконаної роботи, час, витрачений на читання, а також підтримувати стабільність і мотивацію.

Незважаючи на те, що в світі існує велика кількість платформ та застосунків. Іноді деякими досить складно користуватися, або надається мінімальний доступ до функціоналу, наприклад, є можливість додати книгу але немає можливість слідкувати за прогресом читання. Чи не зручний інтерфейс, або навіть заплутана навігація і це є досить вагомий недолік. Через який більшість користувачів покидають платформи та застосунки, також більшість із платформ не мають можливості фіксувати активність користувача під час читання фізичних книг.

Тож, спираючись на дослідження вказані вище, було вирішено створити персональний книжковий трекер та рекомендаційну систему, що дозволить користувачам читати фізичні книги, та ефективно слідкувати за результатами та бути мотивованими в розвитку рухатися далі.

Основні переваги платформи книжкового трекера та рекомендаційної системи:

- зручність: інтуїтивний інтерфейс, можливість пошуку та додавання книг.
- відстеження прогресу та його візуалізація, що підтримує мотивацію та допомагає досягати цілей.
- організація бібліотеки: облік прочитаних, наявних книг і тих, що плануються для читання.

Метою кваліфікаційного проєкту є створення платформи книжкового трекера та рекомендаційної системи, що забезпечить комфортний, зручний інтерфейс та організовану електронну бібліотеку.

Розробка включала проєктування інтерфейсу, вибір технологій і архітектури, програмну реалізацію, адаптацію та тестування відображення на різних пристроях.

Об’єктом розробки є створення платформи для організації процесу зручного відстеження прогресу користувача.

Предметом розробки є веб-платформа книжкового трекера, що містить функціонал автентифікації та авторизації що надаватиме доступ до входу на веб-платформу через відправку даних користувача через форму, три сторінки: головну з рекомендаціями, фільтром і пагінацією, сторінку додавання книг і перегляду обраних, та сторінку трекера.

У результаті було спроєктовано та розроблену веб-платформу, що полегшить процес організації власної електронної бібліотеки. Цей проєкт підкреслює значення інтеграції сучасних веб-технологій для створення ефективних інструментів розвитку.

Методи які були використані при розробці: під час розробки було використано сучасні мови програмування зокрема (JavaScript, TypeScript) а також бібліотеки і фреймворки (React, Styled-components, React-router-dom, Redux-toolkit); для побудови архітектури було застосовано FSD підхід, що сприяло швидкій навігації в коді. У процесі проєктування UX/UI основним завданням було поєднати в платформі ключові принципи ефективного дизайну, а також було враховано аспекти аналітичної обробки даних та збору інформації для створення комфортного користувацького досвіду.

Новизна даної роботи полягає у поєднання мінімалістичного інтерфейсу, інтерактивності та ефективності, що сприяло створенню платформи із зручним інтерфейсом та простим дизайном, який не відволікатиме від основної задачі та не буде перенавантажувати.

Окрім цього, маршрут користувача на платформі був спланований саме таким чином, щоб концентруватися саме на читанні.

Особливістю є те, що система орієнтована на роботу з фізичною книгою. Користувач має можливість слідкувати за прогресом та статистикою. Це дозволяє оцінювати власний прогрес та сприяти подальшій роботі. Тож саме це є особливістю цієї роботи, платформа з рекомендаціями та трекером для роботи з фізичними книгами, яка є універсальною для кожного користувача.

Практична значимість полягає у розробці персонального книжкового трекера і рекомендаційної системи для читачів яка може бути використана для організації процесу читання та відстеження прогресу. Платформа розроблена для відстеження прогресу за допомогою статистики та підтримує мотивацію до регулярного читання та саморозвитку.

Пояснювальна записка до кваліфікаційної роботи складається з чотирьох розділів:

- перший розділ

- другий розділ
- третій розділ

Обсяг пояснювальної записки до кваліфікаційної роботи складає: 62 сторінки, в тому числі з яких: основна частина – 58 сторінок та 5 сторінок додатку, літературних джерел – 15 назв, рисунків – 29.

ПОСТАНОВКА ЗАДАЧІ

Задачею кваліфікаційної роботи є проєктування та розробка веб-платформи на тему «Персональний книжковий трекер і рекомендаційна система для читачів», яка призначена для організації процесу читання, відстеження прогресу і підвищення мотивації. Платформа має сприяти кращій організації та забезпечувати натхненням для регулярного читання.

Основні завдання кваліфікаційної роботи:

- проаналізувати сучасні платформи для читання, виявити переваги та можливі недоліки;
- обрати архітектуру та технологій для реалізації платформи;
- обґрунтувати вибір технологій для зручності;
- спроєктувати інтерфейс для користувача, має бути зрозумілим та інтуїтивним, а також адаптивним відносно інших девайсів для зручного використання;
- спроєктувати та реалізувати функціонал авторизації, що включає реєстрацію, автентифікацію та можливість виходу з облікового запису;
- розробити основний функціонал платформи: трекер читання, навігація, фільтрування книг, додавання нових;

Основну увагу під час роботи було приділено створенню зручної платформи, яка полегшує відстеження прогресу та організації власної бібліотеки.

Відповідно до цього, визначено наступний перелік тем для реалізації:

- вибір технологій та обґрунтування їхнього вибору;
- вибір архітектури;
- розробка функціоналу платформи.
- розробка прототипу для підвищення зручності.

Платформа має забезпечити зручний та організований потік роботи. Структура платформи має вигляд:

- сторінка реєстрації.
- сторінка автентифікації.
- сторінка рекомендацій містить: фільтр, блок із рекомендаціями, пагінація.
- сторінка бібліотека містить: обрані книги та маркер прогресу (в процесі читання/прочитана), можливість додати власну книгу.
- сторінка читання містить: трекер для відстежування прогресу читання .

Мета - створити платформу, яка забезпечує: просту організацію бібліотеки та зручне відстежування прогресу, підвищення продуктивності та мотивації користувачів, інтуїтивний та адаптивний інтерфейс для всіх типів пристроїв.

1. ІНФОРМАЦІЙНИЙ ОГЛЯД

Застосунки та платформи для відстеження прогресу читання стали невід'ємною частиною життя людей, які прагнуть розвитку, або хочуть сформувати звичку читати. З поширенням цифрових технологій, попит на електронні книжкові трекери зріс. Багато користувачів переходить на електронний формат, тому виникає потреба у зручному інструменті, який допомагає контролювати прогрес. Завдяки статистиці, аналітиці та можливості встановлювати цілі, такі сервіси мотивують читати більше.

Також у сучасному світі увага користувачів розпорошується через велику кількість інформації, що знижує рівень концентрації та регулярність читання, що є досить серйозною проблемою для більшості.

На сьогодні є ряд платформ, такі як Goodreads, Librarius, Rork, які надають можливість відстеження прогресу. Проте вони мають певні недоліки, як-от перевантажений інтерфейс чи обмежені можливості персоналізації.

Отже, важливим компонентом сучасних платформ є рекомендаційні системи, які можуть базуватися на контентному аналізі або поведінці користувачів. Вони дозволяють підбирати книги та покращувати користувацький досвід.

Таким чином, актуальним є створення платформи, яка поєднує простий та зручний інтерфейс, можливість відстеження читання фізичних книг та ефективну рекомендаційну систему.

1.1. Сучасні тенденції у створенні веб-сервісів для читання

Сучасні тенденції у створенні веб-сервісів мають дуже цікавий напрямок, який полягає в тому, щоб з прямувати всю увагу на глибокому читанні. Це означає відсутність подразника підчас сеансу читання, що стосується дизайну, надається перевага створенню мінімалістичного

дизайну, використання темних кольорів, плавні анімації та контроль прогресу.

Основна ідея створення персонального книжкового трекера і рекомендаційної системи полягає у забезпеченні комфорту та ефективності користувача. Застосунок має бути простим, інтуїтивним, адаптованим для смартфонів та пристроїв з великим екраном. З урахуванням таких тенденції:

- Адаптивність – підтримка смартфонів, планшетів та десктопів.
- Персоналізація – рекомендаційні алгоритми.
- Трекінг та аналітика – відстеження часу, кількості прочитаних сторінок.
- Зручний UX/UI – інтуїтивний інтерфейс, кольорові маркери прогресу, прості кнопки для додавання книг.

Розглянемо основні платформи, застосунки та їх особливості.

1.2. Аналіз існуючих рішень

Librarius — це український застосунок для читання, що пропонує можливість вибору книг на прокат або купівлю. У додатку можна стежити за прогресом, переглядати статистику читання. Застосунок має додатковий функціонал.

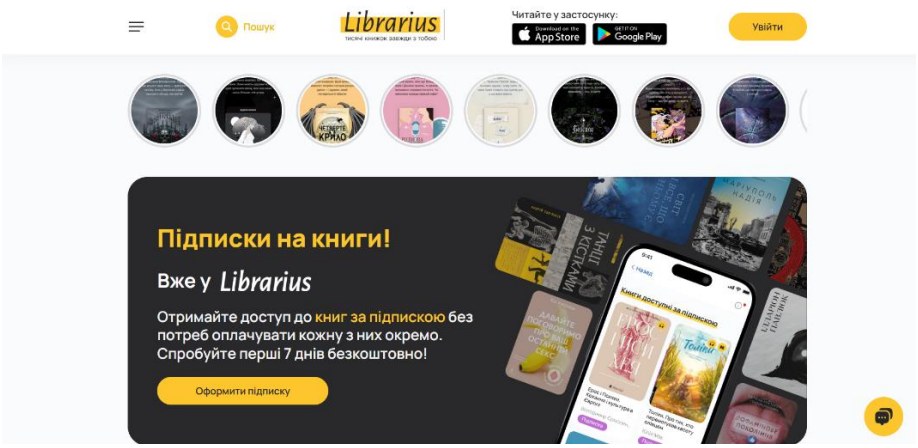

Рис. 1.1. - Librarius

Goodreads — одна з найпопулярніших платформ серед читачів, належить Amazon і є найбільшим онлайн-каталогом книг. Можна додавати книгу в бажане, відзначати прогрес, ставити оцінки, залишати відгуки та коментувати.

Рис. 1.2. - Goodreads

Rork — український застосунок, що працює як читацький щоденник. Допомагає стежити за витраченим часом, кількість прочитаних сторінок на день та швидкість читання. Сервіс має соціальні функції для взаємодії з іншими користувачами, а також підтримує створення текстових і фото нотаток. Також розроблена конфіденційність, користувач сам вирішує, які книги робити публічними.

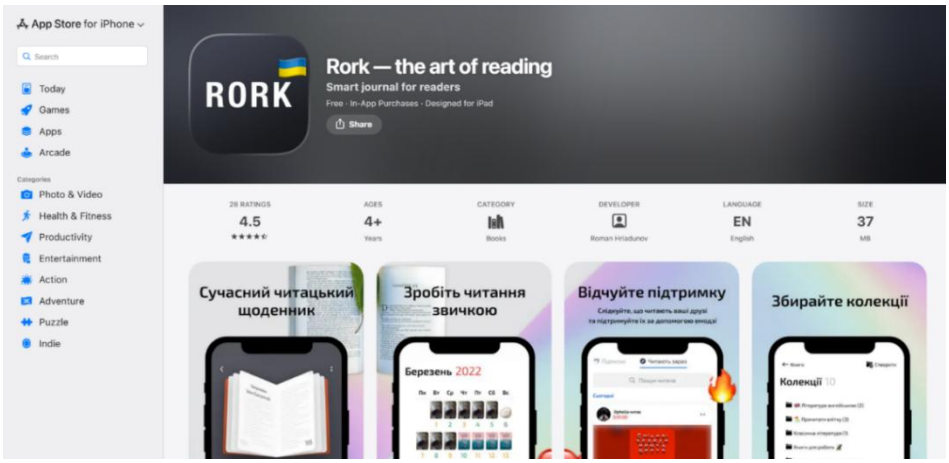

Рис. 1.3. - Rork

Basmo — застосунок дає змогу користувачам під ’єднатися до місцевих бібліотек за допомогою бібліотечної картки. Так вони отримують доступ до всього, що може запропонувати вибрана установа. Чудовий варіант для існування бібліотек у цифрову еру.

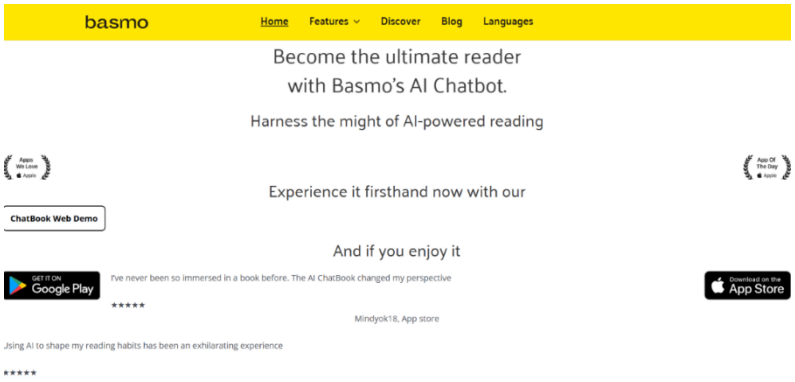

Рис. 1.4. - Basmo

Bookly — додаток для відстеження читання, який допомагає керувати бібліотекою, відстежувати прогрес читання книг, ставити цілі та аналізувати статистику. Він пропонує функції, такі як таймер для читання, нотатки до книг, читацькі виклики та персоналізовані поради, щоб сформувати звичку читати та підвищити продуктивність.

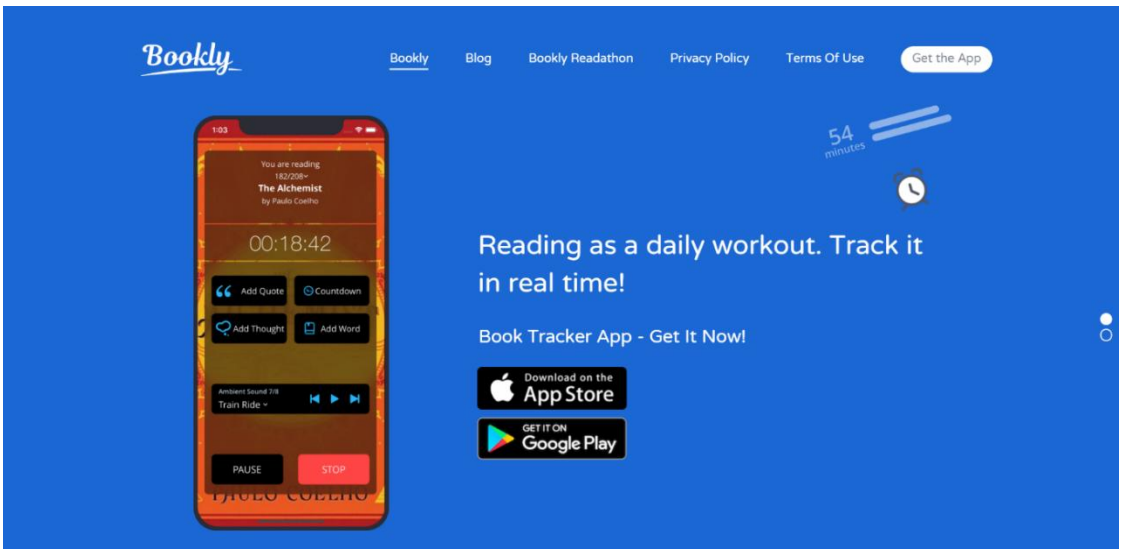

Рис 1.5. - Bookly

Аналітичне порівняння

Аналіз наявних додатків та платформ для відстеження прогресу читання показує, що всі мають свої переваги і особливості. Це стосується функціоналу оскільки в деяких застосунках можливості для користувача значно відрізняються.

Наприклад, порівняймо застосунки Rock та Librarius. Обидва мають велику кількість корисного функціоналу, але вони абсолютно різні, та орієнтовані на різні потреби.

Rock – застосунок, який буде корисним для користувачів, що хочуть ділитися думками. Має приємний кольоровий дизайн та інтуїтивно зрозумілий інтерфейс. Відносно новий продукт, має великі перспективи розвитку.

Librarius – фактично є онлайн-магазином: можна купувати книги, стежити за прогресом, користуватися офлайн, має різноманітний вибір книг українською мовою.

Bookly – пропонує детальну статистику та аналітику, дає глибоку інформацію: скільки часу було витрачено на читання, кількість прочитаних сторінок, швидкість, коли був день читання, як змінювався прогрес. Це великий плюс для користувачів, яким важлива самодисципліна і регулярність.

Goodreads – платформа яка дає можливість для користувачів створювати власну “бібліотеку”: додавати прочитані та плановані книги, створювати списки (“to-read”, “reading”, “read”), ставити оцінки, писати відгуки, зберігати цитати, вести читацький щоденник, брати участь у групах, марафонах, книжкових спільнотах. Також пропонує рекомендації на основі рейтингів, історії переглядів, книг у полиці.

Отже, вибір застосунків та платформ широкий і залежить від потреб та бажань користувачів. Після детального аналізу платформ для відстеження

прогресу читання стає зрозуміло, що потрібно спроєктувати комфортну платформу з урахуванням основних переваг і особливостей перерахованих застосунків.

Це допоможе створити безпечний та корисний інструмент який можна використовувати на різних девайсах у будь-який час.

Таблиця 1.1 – Переваги та недоліки сучасних платформ для відстеження прогресу читання

№	Перевага	Недолік
1	Доступ до сервісу з будь-якого пристрою	Залежність від інтернету
2	Синхронизація даних між пристроями.	Ризик втрати або витоку персональних даних
4	Наявність аналітики та статистики читання	Частина функцій може бути платною
5	Простий дизайн інтерфейсу та наявність рекомендацій	Перевантаженість інтерфейсу в окремих сервісах
7	Підтримка різних форматів читання та нотаток	Залежність від підтримки та оновлень сервісу

Отже, сучасні платформи для відстеження читання мають значні переваги у зручності, аналітиці та персоналізації. Водночас існують певні недоліки, пов’язані із залежністю від цифрових технологій та необхідністю постійної взаємодії користувача із сервісом. Саме тому при розробці власної платформи важливо забезпечити баланс між функціональністю, простотою використання та комфортом користувача.

1.3. Вимоги до персонального трекера та рекомендаційної системи

Після ретельного дослідження ринку, який пропонує велику кількість різних інструментів зокрема персональних трекерів та рекомендаційних систем, було виявлено значну кількість мобільних застосунків та невелику кількість платформ які в більшості мають бібліотеки з англомовними книгами. Деякі з цих сервісів є достатньо громіздкі та містять багато функціоналу який іноді не доречний, також на деяких платформах заплутана навігація. Додатки мають складний процес реєстрації, це може бути однією з причин коли користувач просто не має бажання затримуватися на цьому етапі та припиняє взаємодію з сервісом на перших кроках. З огляду на це, для створення сучасної, інтуїтивної та ефективної платформи необхідно врахувати кілька ключових аспектів, що забезпечать комфорт, функціональність і приємний досвід використання.

На основі проведеного аналізу було визначено основні аспекти, які потрібно врахувати для створення ефективного, комфортного та візуально приємного, а також інтуїтивно зрозумілого веб сервісу який буде допомагати реалізувати цілі користувачів.

Перелік основних вимог для платформи:

- Інтуїтивна та легка навігація. Навігація – це перше з чим зустрічається користувач після відкриття платформи. Вона має бути зрозумілою без додаткових інструкцій. Проста навігація може створювати відчуття контролю, що може бути практично значущим для того, щоб затримати користувача як найдовше.
- Приємний та візуально не нав’язливий інтерфейс. Дизайн відіграє важливу роль у взаємодії користувача із платформою. Надмірно яскраві кольори можуть відштовхнути, велика кількість деталей можуть зробити перебування користувача на платформі не комфортним.

- Адаптивність та кросплатформність.

Кількість технологій для створення веб сервісів дивовижна так як і кількість браузерів. У світі існує близько 200 браузерів, з них 7 є найпопулярнішими. Основна задача полягає в тому щоб розробити платформу яка буде працювати однаково у різних браузерах та на різних пристроях. Особливо це стосується смартфонів так як світова статистика говорить, що мобільні телефони є основним засобом доступу до інтернету.

Рис. 1.1. - Statcounter

- Швидка відповідь на дії користувача. Користувачі звикли до швидкої взаємодії із сервісами, навіть 30 секундна затримка відповіді може погіршити досвід користувача. Якщо додаток довго завантажує контент, зависає, або має дивну поведінку - це знижує інтерес до використання сервісу.
- Безпека даних. Платформа має бути безпечною для використання, а також забезпечити конфіденційність всіх даних які має користувач у в кабінеті. Тобто платформа має гарантувати надійний захист інформації, а також не передавати дані третім сторонам.

Отже, має вийти продукт який задовільнить користувацькі потреби буде сприяти розвитку та допомагатиме у відстеженні прогресу.

Основна ідея зробити сервіс легким, швидким, безпечним. Дуже важливо уникнути перевантаження функціоналом, створити якісний алгоритм рекомендації та приємний інтерфейс. Також потрібно створити архітектуру яка в майбутньому даватиме можливість масштабування та обрати технології.

Вибір технологій дуже важливий крок для реалізації платформи, тому що безпосередньо впливає на продуктивність сервісу, швидкість розробки, зручність інтерфейсу та безпеку даних, тому він має бути обґрунтованим.

2. ТЕОРЕТИЧНА ЧАСТИНА

2.1. Архітектура платформи та модель взаємодії: FSD, REST API

В цьому розділі розглянемо та обґрунтуємо вибір архітектури для проєкту та взаємодії клієнта із сервером. Це важливий крок для створення проєкту із можливістю масштабувати у майбутньому. Потрібно створити логічну архітектуру, яка забезпечить зручність підтримки, а також дозволятиме створювати чітку, пряму структуру. Тож в даному розділі розглянемо архітектуру взаємодії клієнта з сервером тобто REST API (Representational State Transfer Application Programming Interface) та FSD (Feature – Sliced Design) це буде стосуватися структури фронтенду.

Розпочнемо з REST API – (Representational State Transfer Application Programming Interface) – є найпоширенішою архітектурою яка пояснює як взаємодіє користувач із сервером тобто як фронтенд та бекенд спілкуються один з одним за допомогою HTTP протоколу. Слугує прошарком між застосунком та базою даних.

Принципи роботи REST API

- Клієнт-Сервер – патерн який забезпечує незалежну роботу клієнтських та серверних компонентів. Клієнт це фонтенд застосунок який відправляє запит на сервер бекенд це сервер який обробляє запити та повертає відповідь на фронтенд.
- Stateless – не має стану, сервер та клієнт не відстежують стан один одного тобто вони є незалежними, що робить взаємодію гнучкою та масштабованою.
- Єдиний інтерфейс - тобто існує спільна мова між серверами та клієнтами, яка дозволяє вносити зміни без порушення роботи всієї системи.

- Кешування – програма має можливість використовувати отримані дані відповіді пізніше для еквівалентного запити протягом певного періоду.
- Стандартний набір методів для роботи із HTTP протоколом, їх снує багато нам потрібні деякі із них GET - отримання даних, POST – додавання даних у базу, PUT – оновлення даних із повним перезаписом, PATCH – оновлення конкретних властивостей об’єкта, DELETE – видалення даних із бази даних.

Використання REST API для платформи персональний книжковий трекер і рекомендаційна система для читачів. Ця архітектура буде слугувати прошарком для зв’язку нашого фронтенду з бекендом. Буде використовуватися для додавання, видалення та оновлення даних на фронтенді.

Принципи роботи FSD

FSD (Feature – Sliced Design) - це архітектура яку будемо використовувати для фронтенду. Допомагає організувати код так щоб було легко орієнтуватися в структурі та швидко знаходити потрібні компоненти, також допомагає без додаткових зусиль масштабувати проект робити компоненти не залежними один від одного.

Це є архітектура яку найчастіше застосовують у React, щоб організувати проєкт та не загубитися у великій кількості функціоналу.

Основна ідея архітектури FSD це розділити проєкт на шари:

- App – В цьому шарі міститься все, що потрібно для запуску веб-сервісу це може бути глобальні стилі, провайдери, навігація.
- Processes – документація говорить, що цей шар є застарілим, але має призначення для складних між сторінкових функціоналів.
- Pages – містить всі сторінки веб-сервісу або великі частини сторінок.

- Widgets – призначений для компонентів які є самодостатніми інтерактивними елементами на сторінці які мають пряму взаємодію із користувачами.
- Features – це шари який призначений для компонентів які мають користь для загальної бізнес логіки.
- Entities – бізнес-сутності з якими працює проєкт, наприклад користувач або продукт.
- Shared – цей шар містить в собі пере використовуваний код, який потрібний для цілого проєкту.

Тобто вихідний проєкт має містить таку архітектуру.

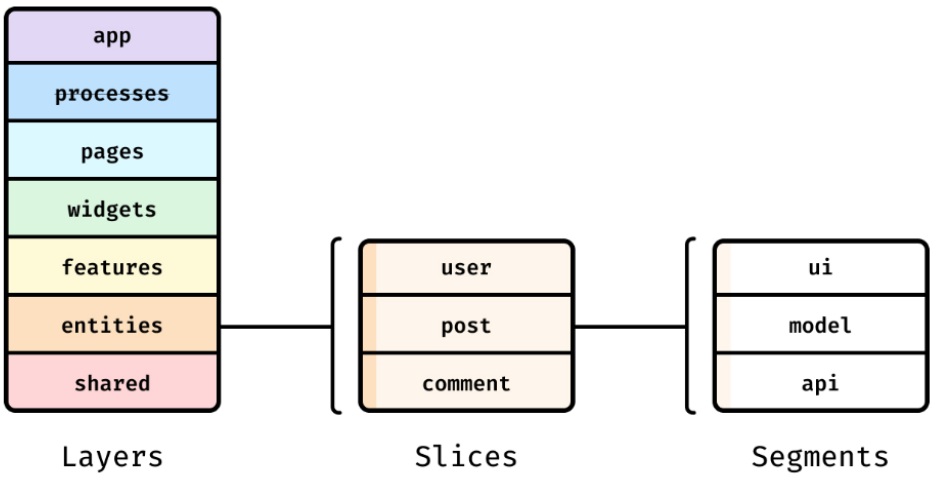

Рис. 1.1. - Архітектура FSD фронтенд.

Також кожен шар поділяється на слайс це потрібно для того, щоб не загубитися в проєкті, тому що вони тісно групують код за сенсом. В свою чергу слайси поділяються на сегменти такі як:

- UI – місить в собі все, що пов’язано і зовнішнім виглядом сайту.
- Api – тут описується взаємодія із бекендом, функції запиту та ін.
- Model – модель даних містить в собі схеми валідації, інтерфейси, бізнес логіку.
- Lib – тут ми можемо описати бібліотечний код.

- Config – містить файли які потрібні для можливих налаштувань.

Отже, для створення платформи трекера та рекомендаційної системи буду використовувати REST API та FST.

REST API використовується для взаємодії із сервером, щоб мати можливість отримати дані, додати, відредагувати або видалити з бази даних.

Для архітектури фронтенду буде використано FSD, тому що це є сучасна заміна хаотичній структурі компонентів у React. Оскільки поділ на шари та слайси забезпечить чітку структуру проєкту, та допоможе створити максимальну не залежність компонентів і полегшити подальшу роботу над проєктом.

Враховуючи наведені дані, поєднання REST API та FST для такого типу платформи забезпечує високу гнучкість, однорідність, швидку орієнтацію в проєкті через чітку архітектуру.

2.2. Огляд інструментів і технологій

Вибір технологій є ключовим етапом у плануванні платформи, оскільки це безпосередньо впливає на швидкість розробки. Потрібно обрати технології, які забезпечать зручну, стабільну та ефективну роботу веб-сервісу. Тож у поточному розділі розглянемо технології як будуть використанні у розробці та обґрунтуємо вибір.

Для взаємодії між фронтендом та сервером буде використано REST API, що забезпечує стандартизовану передачу даних та розподіл відповідальності між клієнтом та сервером. Для створення інтерфейсу, відображення даних та реалізації інтерактивної логіки буде використано бібліотеку яка забезпечить реактивну розробку компонентів та ефективну роботу з DOM.

React - це бібліотека для побудови інтерфейсу, що базується на створенні компонентів та перевикористанні їхньої логіки. React містить в собі інструмент який оптимізує роботу із DOM, що впливає на продуктивність застосунку.

Причини вибору React:

- React має багату екосистему та велике ком'юніті розробників, що спрощує розв'язання найрізноманітніших завдань, прискорює розробку та забезпечує стабільну підтримку.
- Робота з DOM реалізована через абстракцію, що зменшує кількість маніпуляції на пряму, що підвищує продуктивність застосунку та забезпечує високу швидкість.
- Virtual DOM дозволяє ефективно відстежувати зміни у структурі інтерфейсу. React порівнює різні версії віртуального дерева та оновлює лише ті елементи які змінилися.
- Легко інтегрується з зовнішніми бібліотеками та інструментами.

TypeScript – це чудовий інструмент який допомагає уникнути помилок на етапі розробки. Завдяки типам допомагає виявити проблеми, що сприяють зменшенню кількості помилок та пришвидшує роботу.

Також завдяки даному інструменту підвищується читабельність, код стає зрозумілішим, оскільки типи допомагають описати потрібну поведінку, також добре інтегрується з React. Дозволяє типізувати компоненти, пропси, стан, що робить поведінку коду передбачуваною та стійкою. В результаті отримаємо надійний застосунок, що компілює у JavaScript.

Причини вибору TypeScript:

- Забезпечує статичну типізацію, що знижує кількість помилок.
- Допомагає у проєктуванні та структуруванні логіки.
- Підвищує читабельність і передбачуваність коду.
- Інтегрується з React

RTK (Redux Toolkit) - це сучасний потужний набір інструментів для роботи з Redux. Спрощує створення стану для всього додатку, керуємо з одного міста в проєкті. Що робить код компактним, та забезпечує безпечне оновлення стану. Розв'язує проблему отримання даних із сервера. Тож є оптимізованим інструментом для управління станом, який робить Redux менш складним забезпечує роботу із даними та спрощує інтеграцію із API.

Причини вибору RTK (Redux Toolkit) :

- Спільний глобальний стан
- Визначені правила оновлення
- Передбачувана структура
- Інтегрується з Resct, Type Script

StyledComponent – це бібліотека яка дозволяє писати стилі у компонентах прив’язуючи їх до певних елементів розмітки. Що допомагає ізолювати стилі, читабельність і логічне групування. Компонент містить розмітку та стилі в одному місці, легше читати і переносити між проєктами. Також є можливість змінювати стилі динамічно залежно від пропсів.

Причини вибору StyledComponent:

- Динамічність
- Інкапсуляція стилі знаходяться в одному місті та застосовуються лише до певних елементів
- Легко змінювати, керувати та перевикористовувати.
- Відсутність ризику конфлікту стилів.

Отже, для реалізації платформи будуть використовуватися такі інструменти: React, Type Script, RTK (Redux Toolkit), Styled Component. Кожна з перелічених технологій відповідає за певний функціонал та займає важливу роль у розробці платформи. React відповідає за побудову інтерфейсу із високою продуктивністю. TypeScript використовується для типобезпечності та підвищення надійності коду, RTK допомагає

структуровано керувати глобальним станом застосунку, що є необхідним для синхронізації, а також Styled Component дозволяє гнучко створювати стилі для компонентів, забезпечуючи їх ізоляцію та легку підтримку.

Всі перелічені технології легко інтегруються, що забезпечує можливість створити платформу зручною та надійною для користувачів. Поєднання цих технологій забезпечує створення платформи із високою продуктивністю та простою підтримкою.

2.3. Функціональна структура та моделювання платформи

Моделювання платформи це відповідальна частина, потрібно врахувати всі деталі та продумати як компоненти будуть взаємодіяти один з одним. Тож основне завдання поєднати в платформі ключові принципи ефективного UI/UX дизайну:

- Послідовність цей принцип дає змогу користувачам передбачати, як працюватимуть різні компоненти додатку, ґрунтуючись на їхньому попередньому досвіді.
- Простота - спрощений дизайн допомагає користувачам зосередитися на виконанні завдань без відволікання.
- Орієнтованість на користувача цей метод зосереджений на глибокому розумінні потреб, вподобань і досвіду користувачів та адаптації системи до цих вимог.
- Ефективність - ефективний дизайн мінімізує труднощі у взаємодії, допомагаючи користувачам легко виконувати завдання.

Враховуючи вище перелічені принцип дизайну перейдемо до їх застосування для проєктування зручного інтерфейсу. Для цього було використано платформу Miro щоб спроектувати потік та інтерфейс платформи.

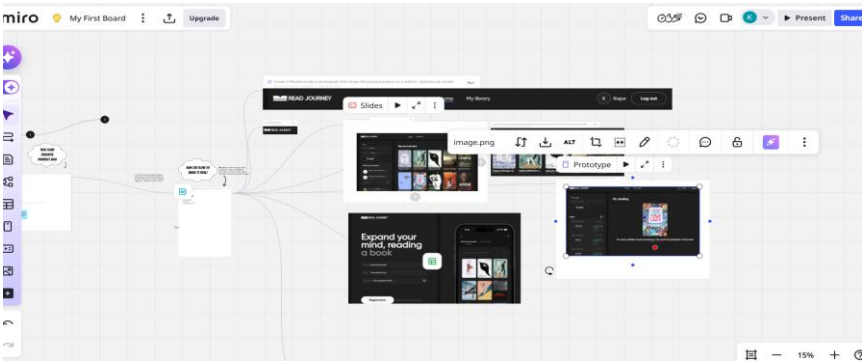

Рис. 2.1. – Приклад проєктування платформи

Прототипний інтерфейс

Сторінка реєстрації – містить форму з трьома полями для введення даних користувача, посилання на сторінку авторизації та блоком-герой, щоб зробити сторінку візуально привабливою.

Головна сторінка - складається із трьох блоків, заголовок сайту, рекомендації та бічної панелі.

Сторінка бібліотеки - містить три блоки, заголовок сайту, блок обраних книг та бічної панелі.

Сторінка трекера - складається також з трьох блоків, заголовок сайту, поточна книга та бічна панель з трекером.

Отже, було розглянуто декілька прикладів планування платформи але після створення поточного прототипу було вирішено що саме цей приклад найкраще задовольняє принципи ефективного UI/UX дизайну

Палітра кольорів та шрифт

Щоб користувачам було легко та приємно працювати з платформою, було розроблено ключові елементи дизайну:

- Кольори**:** синій, білий і сірий створюють сучасний і професійний вигляд, а яскраві акценти виділяють важливі кнопки.

Рис. 2.2. – Палітра кольорів

- Шрифти: Gilroy забезпечують легке читання на будь-яких пристроях. Дизайн UI/UX робить портал зручним та привабливим, дотримуючись принципів інтуїтивності та ефективності взаємодії користувачів із сервісом.

Також була проведена робота над тим як відбуватиметься потік на сайті. Який шлях матиме користувач від початку реєстрації та аж до виходу із кабінету.

Дизайн створювався за принципом mobile-first, що дозволяє оптимізувати досвід користувачів на смартфонах — основному пристрої для сучасного читання. Також буде реалізована адаптація платформи для того щоб користувачі мали змогу зайти в свій кабінет із смартфону та інших пристроїв.

Опитування користувачів для покращення досвіду платформи

Для перевірки зручності та зрозумілості інтерфейсу було проведено опитування серед потенційних користувачів платформи. Перед тестуванням були розроблені макети, що дозволило швидко змоделювати основні сценарії взаємодії: реєстрацію, перегляд рекомендацій, додавання книги, використання трекера читання та навігацію між сторінками.

Учасникам було запропоновано виконати низку типових дій та оцінити інтерфейс за кількома критеріями: дизайн, контрастність,

читабельність шрифтів, легкість навігації та зрозумілість взаємодії з компонентами.

У процесі було виявлено низку недоліків, серед яких найважливішою проблемою став низький контраст окремих елементів інтерфейсу, що негативно впливало на сприйняття важливих елементів. Після аналізу фідбеку дизайн було модернізовано: змінено колірні акценти, посилено контраст кнопок та збільшено кегль шрифтів на ключових елементах.

Після внесення коректив прототип повторно протестували, і результати було отримано позитивні результати. Отримані результати стали основою для остаточного дизайну платформи.

3. ПРАКТИЧНА ЧАСТИНА

3.1. Реалізація інтерфейсу платформи UX/UI

При розробці платформи основний акцент був зроблений на зручності та інтуїтивності під час взаємодії. Дизайн створювався з урахуванням принципів UX/UI, що дозволяють зменшити когнітивне навантаження та забезпечити швидке виконання основних дій. Основна увага приділялася простоті та комфортності взаємодії з платформою, щоб користувач не відчував перевантаження під час роботи із системою.

Основна перевага платформи це кольори та навігація, швидко, ненав’язливо та просто. Також основна увага була спрямована на те, щоб користувач міг швидко зрозуміти логіку платформи, без додаткових пояснень. Для цього було використано такі патерни, як бокове меню та карткове відображення контенту. Також важливим було спланувати навігацію по платформі тож було реалізовано навігацію у вигляді чіткої та структурованої системи для цього було розроблено шапку платформи, що забезпечує швидкий доступ до ключових функцій таких, як бібліотека та домашня сторінка. Окрім цього, візуальне виділення активного розділу допомагає користувачу орієнтуватися в поточному контексті

Компоненти інтерфейсу, у дизайні присутня велика кількість однотипних компонентів таких як (кнопки, картки, модальні вікна), що дало змогу реалізувати цілісний інтерфейс. Кнопки в інтерфейсі мають чітку візуальну ієрархію, кожна відповідає за певну дію (наприклад, відтворення, додавання або підтвердження) кожна виділена акцентним кольором також є другорядні менш контрастні. Також передбачено різний стан кнопок, що забезпечує зворотний зв’язок під час взаємодії. Ключовим компонентом є картки книги, що відображають контент (назву, автора, кількість сторінок), також вони містять зображення. Під час взаємодії з

карткою з’являється модальне вікно. Такий підхід дозволяє компактно структурувати інформацію та забезпечує зручну навігацію між елементами. Також на платформі є форми, за допомогою яких відбувається взаємодія із платформою, кожна форма має валідацію, що забезпечує можливість додати нову книгу до бібліотеки чи відфільтрувати потрібну. Модальні вікна використовуються для виконання додаткових дій з переходом на нову сторінку (наприклад, додавання контенту). Вони затемнюють фоновий інтерфейс, фокусуючи увагу користувача на конкретній книзі. Такий підхід зменшує відволікання та спрощує виконання потрібних дій.

Візуалізація для цього було обрану темну тему, яка зменшує навантаження на очі, відповідає сучасним тенденціям, забезпечує краще фокусування а також темна тема часто асоціюється з якісними сервісами. Для акцентування уваги було обрано такі кольори як (зелений, червоний, синій, білий), щоб привернути увагу користувача до важливих елементів таких як (кнопки, активні стани, повідомлення). А також було вирішено подавати інформацію блоково для кращого сприйняття через те, що інформація легше сприймається, мозок швидше обробляє контент, а також виглядає чисто та структуровано. Вибір такої моделі візуалізації сприяє тому, що користувач довше лишається на платформі, менше втомлюється та краще концентрується на книзі через відсутність подразників. Окрім цього застосовані такі UX-ефекти як залучення до дій та живий інтерфейс.

Адаптивність, окрім цього було розроблено інтерфейс з урахуванням різних розмірів екранів, що надає можливість використовувати платформу на різних пристроях. Було використано адаптивну верстку, де основою реалізації став підхід **mobile-first**, при якому розробка інтерфейсу починається з найменших екранів. Це дозволило забезпечити високу продуктивність на мобільних пристроях. Після мобільної версії інтерфейс

поступово масштабується за допомогою правил **media queries**, що забезпечують реалізацію адаптивної верстки. Точки перелому для мобільної версії від 0 до 767 px, для планшетів від 768 px і до 1440 px та для десктопів від 1440 px, додаючи складніші елементи та розширюючи функціональність для більших екранів. Елементи гнучко підлаштовуються під різні девайси та забезпечуються комфортне використання як на десктоп так і на мобільних пристроях. Також для забезпечення однакового відображення інтерфейсу в різних браузерах було використано додаткову бібліотеку (normalize.css). Це дозволяє вирішує такі проблеми як відмінності стандартних стилів браузерів, забезпечення однакової роботи UI в різних браузерах.

Простота та зрозумілість одним із ключових принципів стала простота. Було уникнуто перевантаження інтерфейсу складними елементами або зайвою інформацією. Кожен елемент інтерфейсу має чітке функціональне призначення та використовується лише тоді, коли це дійсно необхідно. Користувач отримує тільки те, що потрібно для виконання дії. Особлива увага приділялася логічному розміщенню елементів. Основні дії (додавання книги, запуск трекера, фільтрація) користувача винесені у видимі та доступні області інтерфейсу. Це дозволяє користувачу виконувати необхідні операції у мінімальну кількість кроків наприклад:

- фільтрація книг — у лівій панелі;
- перехід до бібліотеки — через навігацію;
- старт роботи (onboarding “Start your workout”) — у лівій панелі;
- взаємодія з книгами — через великі клікабельні карточки;
- трекер — взаємодія через форму

3.2. Реалізація функціональної логіки

Реалізована функціональна логіка складається з основних процесів взаємодії користувача із системою. Система підтримує реєстрацію, вхід

користувачів та вихід з облікового запису. Для цього було використано механізм JWT та збереження даних до localStorage. Основний функціонал платформи пов'язаний із бібліотекою користувача та трекером. Для реалізації платформи було обрано такий архітектурний підхід як SPA — це як працює застосунок, FSD — це як він організований всередині, для того, що забезпечити швидку роботу платформа та зменшити взаємодію із DOM та FSD для зручної навігації по коду, що передбачає поділ коду на логічні шари та модулі за функціональністю.

Для розробки додатку було використано Visual Studio Code (VS Code) це редактор вихідного коду.

Проєкт було розгорнуто за допомогою Vite – це сучасний інструмент для збірки проєктів, було додано додаткові пакети для роботи з маршрутизацією, стилями, формами, даними.

Для роботи із архітектурою код було використано FSD. Що в свою чергу допомогло перетворити хаотичну структуру React на чітко організовану за відповідальністю ієрархію компонентів. Також було використано функціональний підхід до створення компонентів так як використання класів вже є застарілим а також функціональний компоненти простіше налаштовувати.

Структура клієнтської частини складається з:

components – містить інтерфейсні елементи, які не прив’язані до конкретної бізнес-логіки.

entities/card – відповідає за представлення основних сутностей предметної області.

features - містить реалізацію окремих функціональних можливостей системи, складається: авторизація користувача, додавання книг, фільтрація та трекінг читання.

guards – зберігається функціонал захисту маршрутів робить, в залежності від того чи користувач зареєстрований.

helpers – складається із допоміжних функцій, які містяться в різних частинах проєкту.

pages – зберігаються всі сторінки платформи, що відповідають окремими маршрутам.

redux- зберігається логіка керуванням стану проєкту, що включаю користувача, книги та фільтрацію.

shared – містить спільні ресурси які використовуються в різних модулях, зокрема універсальні компоненти.

Stylesheet – містить стилі та шрифти.

Types – містить типи які використовуються в різних компонентах.

Widgets – містить великі шматки функціональних компонентів.

Основний функціонал

Авторизація один із найважливіших модулів, що забезпечує доступ користувача до платформи через власний профіль. Складається з реєстрації **(код А.1)** або входу до системи **(код А.2)** через форму введення даних яка має обробку помилок та схему валідації **(код А.3)**. У разі успішної авторизації користувач автоматично перенаправляється на сторінку рекомендацій. Також є реалізований захист маршрутів за допомогою компонентів **PrivateRoute** та **PublicRoute (код А.4)**. Дані компоненти перевіряють чи користувач ввійшов в систему та чи має право на доступ до захищених маршрутів. Це забезпечує захист приватних даних та коректну навігацію в межах застосунку. Для виходу із системи створений функціонал який працює таким чином, при кліку на кнопку відправляється запити, та видаляємо JWT автоматично і перекидаємо на сторінку входу.

Рекомендації, даний блок виконує інформаційну функцію та відображає список книг, рекомендованих для прочитання. Після успішної авторизації користувач автоматично переходить на відповідну сторінку . Яка містить функціонал фільтрації, для швидкого пошуку книг, а також

можливість додавання обраних книг до власної бібліотеки. Також передбачений блок інструкції який описує, що можна створити власну бібліотеку, та розпочати тренування. Для зручності користувачів реалізовано пагінацію, що дозволяє ефективно переглядати великий обсяг даних.

При взаємодії з книгою відкривається модальне вікно із інформацією про книгу, зокрема кількість сторінок, автор та повну назву. Також передбачено можливість додавання книги до власної бібліотеки. Реалізований функціонал відповідає темі роботи та забезпечує зручний інструмент для організації читання і формування персональних рекомендацій.

Бібліотека є важливою частиною платформи, що забезпечує можливість формування та управління власною книжковою полицею користувача. Для цього було розроблено окрему сторінку, яка містить основний функціонал для роботи із книгами.

Зокрема, на сторінці представлена форма для додавання власної книги, що дозволяє користувачу самостійно вносити назву, автора та кількість сторінок. Це забезпечує зручність роботи з фізичними книгами та дозволяє відстежувати процес читання. Кожна картка містить основну інформацію про книгу та кнопку для її видалення. На сторінці реалізований селек для зручності користувача за допомогою якого можна відсортувати книги за статусом (наприклад, всі книги, у процесі читання, прочитані). А також при взаємодії з карткою книги користувач може додати її до трекера та розпочати процес відстеження прогресу. На сторінці також реалізовано блок рекомендацій, що відображає додаткові книги, які можуть зацікавити користувача. Це дозволяє поєднати управління власною бібліотекою з можливістю відкривати нові книги. Крім того, інтерфейс організований таким чином, щоб забезпечити швидку навігацію між основними

розділами платформи (Home, My library), що покращує загальний користувацький досвід.

Трекер читання є ключовим компонентом платформи, що забезпечує відстеження прогресу користувача та підтримує його мотивацію до регулярного читання. Основною ідеєю є можливість роботи з фізичною книгою з одночасною фіксацією результатів читання. Функціонал трекера реалізовано у вигляді форми, в якій користувач вводить номер початкової сторінки та запускає сесію читання за допомогою кнопки «Start». Для завершення сесії користувач вводить кінцеву сторінку та натискає кнопку «Stop». На основі введених даних система автоматично обчислює тривалість читання та кількість прочитаних сторінок.

Після завершення сесії відображається статистика читання, яка включає дату, кількість прочитаних сторінок, витрачений час та швидкість читання (сторінок за годину). Також реалізовано візуалізацію даних у вигляді графічного елемента, що дозволяє наочно оцінити ефективність читання. Додатково відображається прогрес-бар, який показує відсоток прочитаної книги та кількість сторінок, що залишилися до завершення. Це допомагає користувачу краще орієнтуватися у власному прогресі та планувати подальше читання. На сторінці також представлена картка книги, яка містить основну інформацію про поточну книгу, що читається (назва, автор). Усі елементи інтерфейсу організовані таким чином, щоб користувач міг швидко запускати та завершувати сесії читання без зайвих дій.

Управління станом застосунку реалізовано за допомогою інструменту Redux Toolkit (RTK). Оскільки платформа має декілька функціональних модулів, було обрано централізований підхід до зберігання даних, що забезпечує наявність єдиного джерела правди (single source of truth). Це дозволяє ефективно керувати станом застосунку та спрощує взаємодію між його компонентами. Структура стану організована

у вигляді окремих підмодулів, зокрема: авторизація (auth), книги (books) та фільтрація (filter). Такий поділ забезпечує логічне розмежування відповідальності та підвищує зрозумілість коду. Кожен підмодуль складається зі slice, operations (асинхронні операції) та selectors. Slice відповідає за опис стану та редюсерів, operations — за виконання асинхронних запитів до сервера, а selectors — за отримання необхідних даних зі стану. Глобальний store об'єднує редюсери auth, books та filter, забезпечуючи централізоване керування станом усього застосунку.

Логіка взаємодії компонентів побудована на основі бібліотеки React та архітектури одно сторінкового застосунку (SPA). Це дозволило створити платформу з високою швидкістю роботи та плавною взаємодією користувача з інтерфейсом. Під час взаємодії користувача із застосунком оновлюються лише ті компоненти, стан яких змінюється, без повного перезавантаження сторінки. Такий підхід забезпечує підвищення продуктивності та зменшує навантаження на браузер. Інтерфейс оновлюється на стороні клієнта, що дозволяє уникнути зайвих запитів до сервера. Це забезпечує швидке завантаження сторінок та покращує загальний користувацький досвід.

Окрім цього кожна дія користувача супроводжується повідомленням про успішність виконання або виникнення помилки. Це дозволяє користувачу розуміти стан системи та результати виконаних дій, що покращує взаємодію з платформою. Для стилізації застосунку було створено окремі папки для зберігання стилів та шрифтів, які підключено глобально для доступу в різних частинах проєкту.

Таким чином, реалізована функціональна логіка платформи забезпечує зручну взаємодію користувача із системою, швидке виконання основних дій та ефективне керування даними. Використання сучасних підходів до розробки, зокрема React, SPA-архітектури, FSD та Redux Toolkit,

дозволило створити масштабований та структурований застосунок із високою продуктивністю та зручним користувацьким інтерфейсом.

UML-діаграма

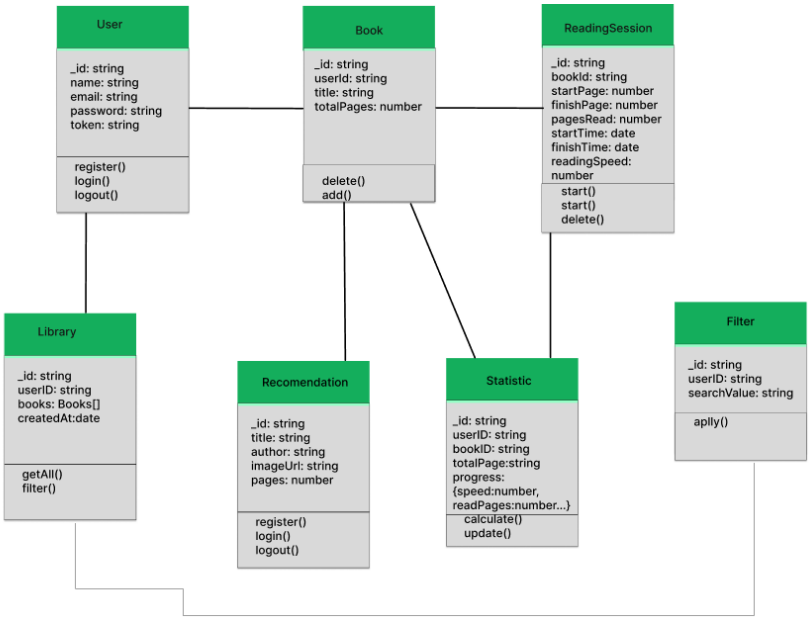

Рис.3.1 – UML-діаграма

Для проєктування структури платформи було розроблено UML-діаграму, яка відображає основні сутності системи, їх властивості, методи та взаємозв’язки між ними. Діаграма демонструє логіку організації даних у застосунку та взаємодію між ключовими компонентами платформи.

Основними сутностями системи є: User, Book, Library, ReadingSession, Recommendation, Statistic та Filter. Сутність User відповідає за роботу з користувачем та авторизацією, Book — за зберігання інформації про книги, Library — за формування персональної бібліотеки користувача, а ReadingSession — за відстеження процесу читання. Окремо реалізовано сутності для рекомендацій книг, статистики читання та фільтрації даних.

3.3 Інтеграція з сервером

Інтеграція клієнтської частини із сервером реалізована за допомогою HTTP-протоколу та побудована на архітектурі клієнт-сервер із використанням моделі «запит–відповідь». У проєкті для роботи із сервером використано бібліотеку Axios та інструмент керування станом Redux Toolkit. Такий підхід дозволяє ефективно організувати обмін даними між клієнтом і сервером, а також централізовано керувати отриманою інформацією.

Для виконання HTTP-запитів використовується окремо налаштований екземпляр Axios **(Код.А.5)** із базовим URL сервера. Це забезпечує зручність повторного використання та уніфікацію запитів у межах усього застосунку. Для централізованого контролю запитів використано **Axios interceptors**, які дозволяють автоматично додавати токен до кожного запиту та обробляти помилки на глобальному рівні. Крім того, реалізовано механізм встановлення токена авторизації у заголовках запитів, що дозволяє виконувати доступ до захищених ресурсів.

У проєкті використано модульний підхід до організації стану за допомогою **Redux Toolkit**, що передбачає розділення логіки за функціональними частинами. Для кращої організації взаємодії з сервером використано підхід із розподілом логіки модулів на slice, operations (асинхронні дії) та selectors. Асинхронні запити до сервера реалізуються через operations, які виконують отримання, додавання, оновлення або видалення даних, окрім цього тут відбувається обробка помилок які можуть трапитися під час запиту на сервер для цього використовуються конструкції try/catch також додано типізацію функцій запитів щоб уникнути помилок під час відправки даних та типізувати відповіді від сервера щоб забезпечити чітку структуру даних, які використовуються у додатку. Результати запитів зберігаються у відповідних slice, що дозволяє централізовано керувати станом застосунку.

Selectors використовуються для отримання необхідних даних зі стану та передачі їх до компонентів. Це забезпечує відокремлення логіки доступу до даних від UI та підвищує масштабованість застосунку.

Структура модулів Redux включає:

- auth — авторизація користувача
- books — робота з книгами
- filter — фільтрація даних
- store.ts — глобальне сховище стану

Такий підхід забезпечує масштабованість, читабельність і зручність підтримки коду.

Модуль auth, відповідає за реєстрацію та вхід на платформу а також за збереження токена і стан користувача. Окрім цього важливим є те, що саме завдяки цьому модулю реалізується контроль доступу до захищених ресурсів та збереження токена, що використовується в Axios, що створює можливість робити уніфіковані запити за інформацією користувача до сервера. Для структуризації модуль розділений на

- authSlice.ts де зберігаються дані стану, що стосується користувача та обробники різних сценаріїв,
- operations.ts, містить асинхронні дії (thunks), які взаємодіють із сервером: оновлення токена (GET), створення користувача, вхід, та вихід (POST).
- selectors.ts містить функції для отримання даних зі стану та відображення в різних компонентах наприклад дані користувача, статус користувача та оновлення.

Модуль books це один із ключових модулів, який відповідає за роботу з книгами. Його структура побудована за принципом розділення відповідальності

- **bookSlice.ts** містить початковий стан, reducers (синхронні зміни стану), які допомагають обробляти різні випадки в додатку,

видалення, додавання та відображення, обробку асинхронних дій (extraReducers). Саме тут визначається як змінюється список книг та як обробляється loading / error.

- **operations.ts** містить асинхронні дії (thunks), які взаємодіють із сервером отримання книг (GET), додавання (POST), оновлення (PUT/PATCH), видалення (DELETE).
- **selectors.ts** містить функції для отримання даних зі стану наприклад: список книг, статус завантаження, відфільтровані дані.

Модуль filter забезпечує збереження значень фільтрів (назва, автор), динамічне оновлення списку книг, фільтрація відбувається на клієнті. Містить такі файли:

- **filterSlice.ts** містить початковий стан, reducers (синхронні зміни стану) які допомагають обробляти відображення. Саме тут визначається як змінюється список книг.
- **operations.ts** містить асинхронні дії (thunks), які взаємодіють із сервером отримання книг (GET).
- **selectors.ts** містить функції для отримання даних зі стану наприклад: список відфільтрованих книг.

Модуль store.ts це центральна точка:

- об’єднує всі slice
- створює глобальний store
- підключає middleware

Приклад логіки:

```
export const store = configureStore({
  reducer: {
    books: booksReducer,
    auth: authReducer,
    filter: filterReducer,
```

```
  },
});
```

Завдяки такому підходу забезпечується асинхронна взаємодія з сервером без перезавантаження сторінки, що відповідає концепції SPA. Дані автоматично оновлюються після виконання запитів, а користувач отримує актуальну інформацію в реальному часі.

Для підвищення безпеки та стабільності роботи застосунку реалізовано механізми обробки помилок запитів, перевірки авторизації користувача та автоматичного оновлення інтерфейсу після зміни даних. Це дозволяє забезпечити коректну взаємодію користувача із платформою навіть у випадку виникнення помилок під час обміну даними із сервером.

3.4 Огляд платформи персонального книжкового трекера і рекомендаційної системи для читачів

Розроблена платформа надає можливість користувачеві створювати власну електронну бібліотеку, працювати зі списком рекомендованих книг, відстежувати прогрес читання, редагувати історію читання, а також додавати, видаляти та фільтрувати та сортувати книги за статусом.

Після відкриття веб-платформи користувач автоматично потрапляє на сторінку реєстрації.

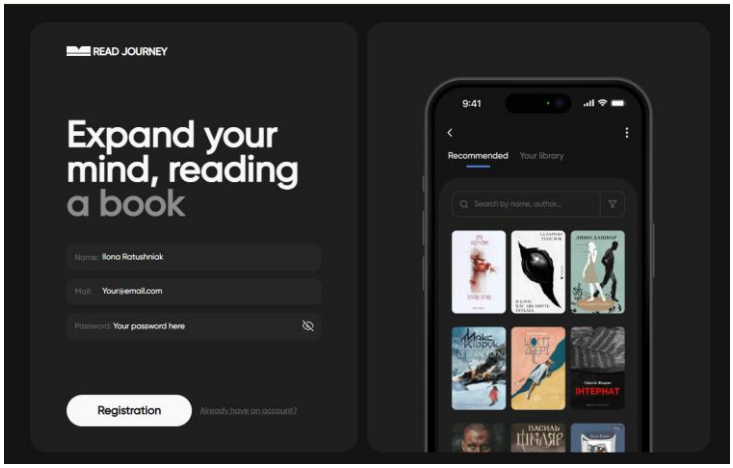

Рис. 3.2 — Сторінка реєстрацій

Ця сторінка складається з форми введення даних та посилання для переходу на сторінку входу. Форма містить механізм валідації, що забезпечує перевірку коректності введених даних перед їх відправленням на сервер. Усі поля є обов’язковими для заповнення. Окрім цього, розроблено функціонал приховування та відображення пароля за допомогою окремої кнопки, що покращує зручність взаємодії користувача з формою.

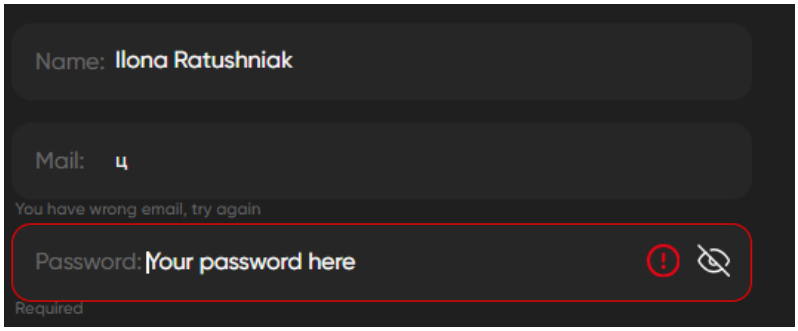

Рис. 3.3 — Валідація форми реєстрації

Дозволяє попереджати користувача про помилки введення ще до надсилання даних на сервер. Це знижує кількість некоректних запитів та підвищує стабільність роботи системи.

Для швидкого переходу між сторінками реалізовано навігаційне посилання на сторінку входу. Сторінка входу має подібну структуру, проте містить інший набір полів для авторизації користувача.

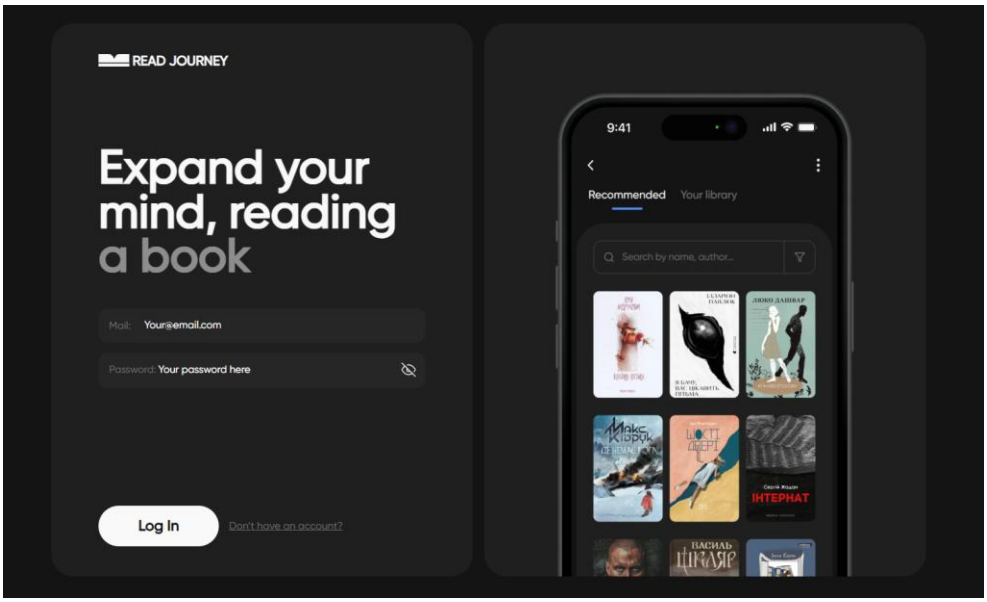

Рис. 3.4 — Сторінка входу

Забезпечує автентифікацію користувача в системі. Форма також підтримує валідацію введених даних та функціонал приховування пароля.

Для покращення взаємодії користувача з платформою реалізовано систему повідомлень, яка інформує про успішне виконання дій або виникнення помилок під час роботи із системою. Для цього було додано бібліотеку за допомогою якої було реалізовано систему повідомлень.

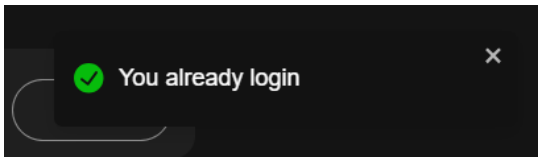

Рис. 3.5 — Взаємодія користувача із платформою

Сторінка рекомендації складається з таких елементів, як: заголовок сайту який містить навігаційне меню, інформацію про користувача та кнопку виходу з облікового запису. Основною частиною сторінки є список рекомендованих книг.

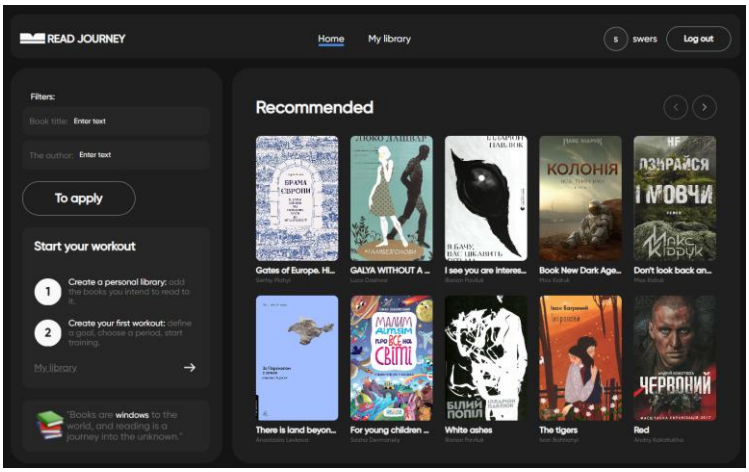

Рис. 3.6 — Сторінка рекомендації

Окрім цього, у правій частині сторінки розташована форма фільтрації за допомогою якої можна швидко знаходити потрібні книги за визначеними параметрами: назва книги та автор. Також реалізовано перевірку введених значень та додано валідацію, для покращення ефективності пошуку.

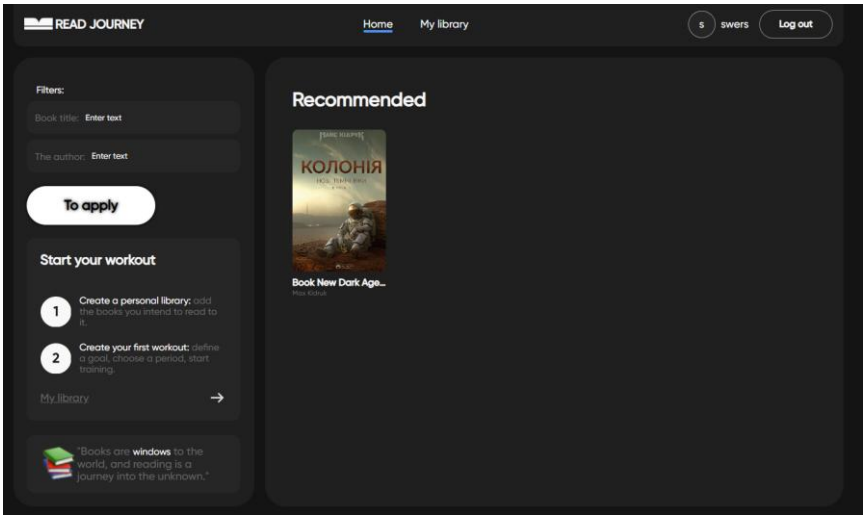

Рис. 3.7 — Форма фільтрації

Дозволяє швидко знаходити потрібні книги за визначеними параметрами. Також реалізовано перевірку введених значень для покращення ефективності пошуку.

У випадку відсутності книги система автоматично відображає стандартне зображення, що забезпечує цілісність інтерфейсу.

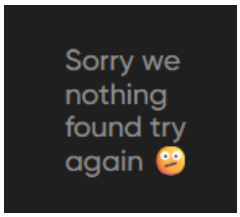

Рис. 3.8 — Зображення за замовчуванням

Для того щоб користувач швидко орієнтувався, як взаємодіяти із платформою розроблено інформаційний блок із підказками щодо створення власної бібліотеки та початку роботи з трекером читання. Демонструє основні кроки для створення власної бібліотеки.

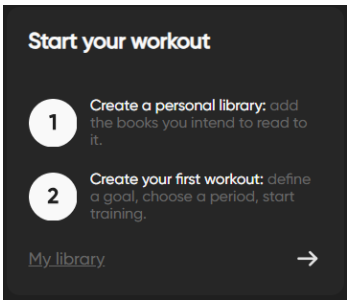

Рис. 3.9 — Інформаційний блок

Для зручного перегляду великої кількості книг реалізовано пагінацію у вигляді кнопок. Перемикання між сторінками здійснюється за допомогою кнопок, активна сторінка візуально виділяється стилями. Для кращої орієнтації було додано додаткові стилі, які показують, яка кнопка активна.

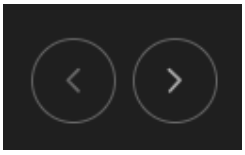

Рис. 3.10 — Пагінація книг

Для взаємодії з книгами використано модальні вікна. Розроблено модальне вікно за допомогою якого зручно керувати та додавати книги до бібліотеки.

Після натискання на картку книги відкривається додаткове вікно з детальною інформацією про книгу та кнопкою додавання до бібліотеки.

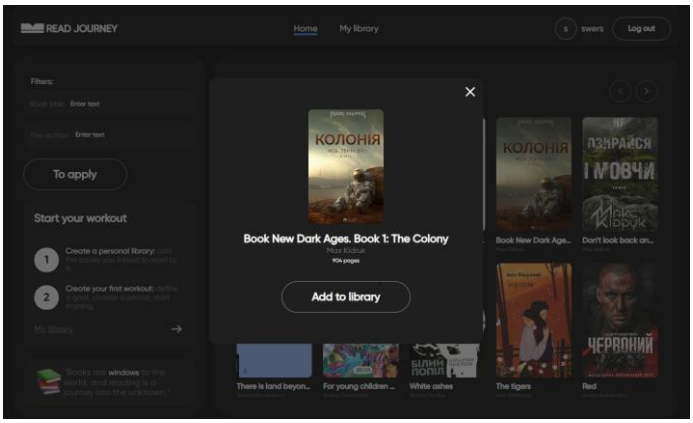

Рис. 3.11 — Модальне вікно

Після успішного додавання книги користувач автоматично перенаправляється до сторінки бібліотеки. Також відображається повідомлення про успішне виконання дії. Це зпроєктовано для того, щоб повідомити користувача, що додавання книги пройшло успішно та можна відразу починав читати. Після додавання книги з’являється модальне вікно із привітання.

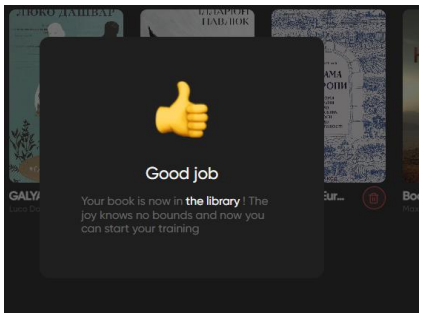

Рис. 3.12 — Модальне вікно привітання

Оскільки платформа орієнтована на роботу з фізичними книгами. Створена для того, щоб слідкувати за прогресом під час читання фізичних книг було реалізовано можливість самостійного додавання книги до бібліотеки користувача. Розроблено форму за допомогою якої можна створити власну книгу.

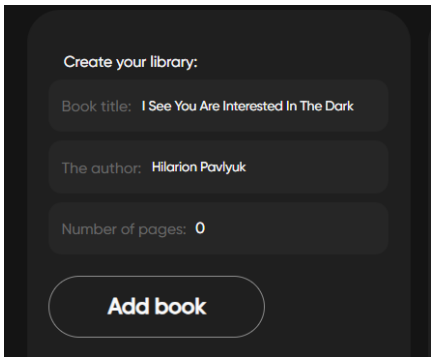

Рис. 3.13 — Форма додавання фізичних книг

Після створення нової книги вона автоматично додається до списку вибраних книг. На сторінці також реалізовано блок рекомендацій із додатковими книгами. А також для більш детального перегляду рекомендацій є посилання за допомогою якого можна перейти на сторінку із рекомендованими книгами. Це гарантує швидку та ефективну роботу із платформою, що базується на активних діях користувача. Кожна взаємодія із платформою сприяє підвищенню залученості користувача.

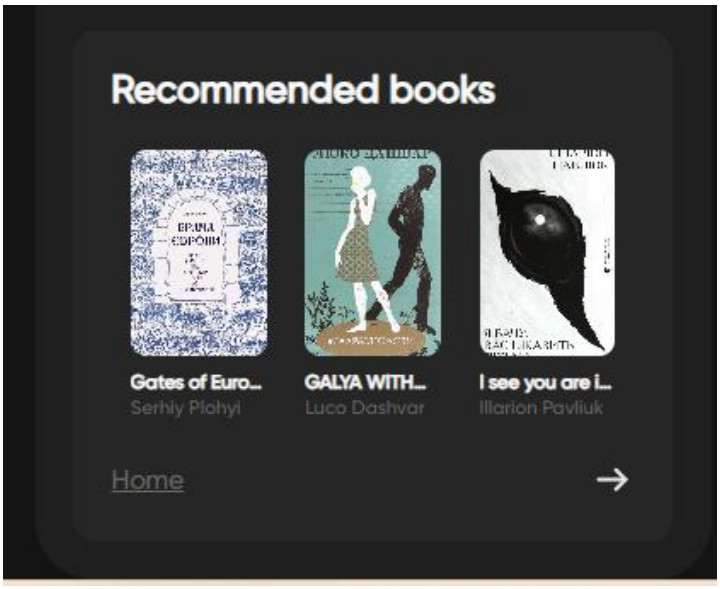

Рисунок 3.14 — Блок рекомендацій.

Для сортування книг за статусом було розроблено спеціальний select-компонент. Даний компонент забезпечує швидке сортування книг за станом. Містить чотири можливі стани за якими відбувається сортування, що дозволяє швидко відображати книги відповідно до їхнього стану:

- усі книги;
- у процесі читання;
- прочитані;
- заплановані до читання.

При виборі конкретного стану за допомогою стилів показано який саме стан було застосовано для сортування.

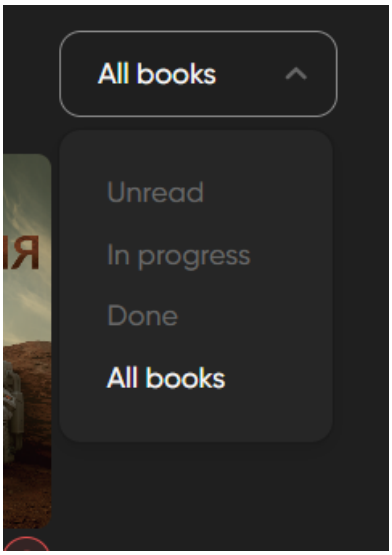

Рис. 3.15 — Селект для сортування книг

Сторінка трекера для відстеження процесу читання. Складається із таких елементів як: форма керування процесом читання, картка поточної книги, історія читання, блок статистики. Для того щоб користувач мав вибір як відображати дані було створено кнопки за допомогою яких надається можливість вибору як саме відображати статистику у вигляді історії читання чи progress bar. Progress bar розроблений для відображення загального прогресу читання.

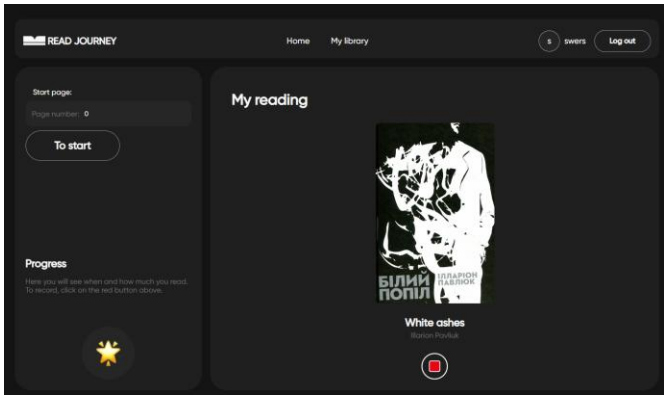

Рис. 3.16 — Сторінка трекера

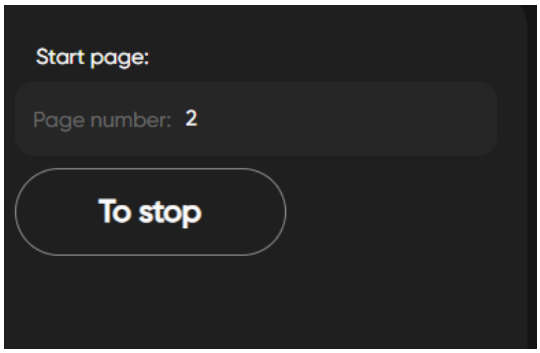

Рис. 3.17 — Форма трекера

Для запуску сесії читання користувач вводить номер сторінки та натискає кнопку «Start». Після завершення читання необхідно ввести кінцеву сторінку та натиснути кнопку «Stop». Для візуалізації результатів реалізовано блок статистики, який дозволяє користувачу аналізувати власний прогрес.

Для того щоб користувач мав можливість самостійно обирати спосіб відображення статистики, було створено спеціальні кнопки перемикання режимів перегляду. Це дозволяє змінювати формат подання статистичних даних залежно від потреб користувача та забезпечує більш зручний аналіз результатів читання.

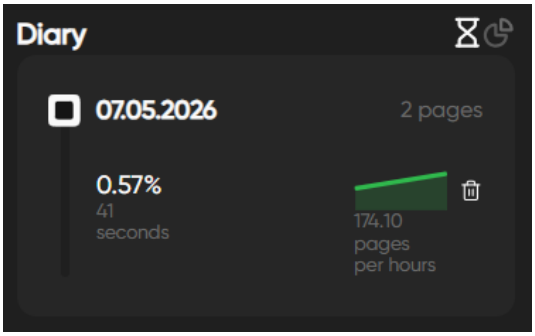

Рис. 3.18 — Записник

Записник відображає основні показники процесу читання, зокрема кількість прочитаних сторінок, тривалість читання, дату проведення сесії

та швидкість читання. Для кращого сприйняття інформації реалізовано декілька варіантів візуалізації даних.

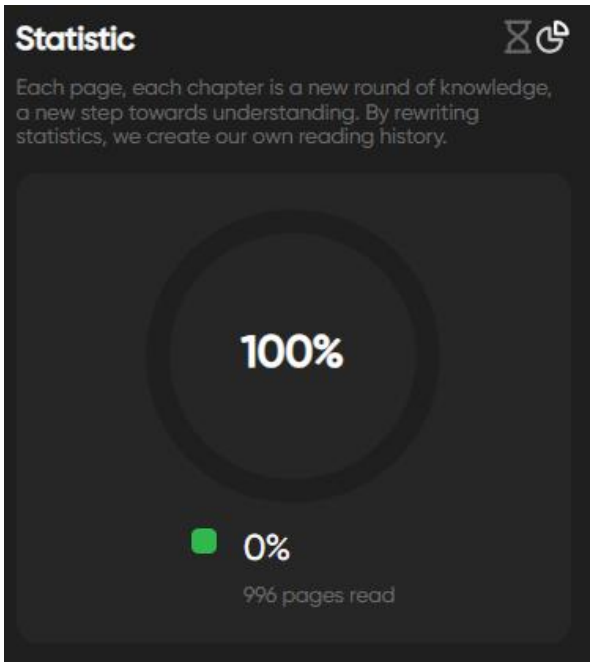

Рис. 3.19 — Progress bar

Progress bar візуально відображає загальний прогрес читання книги. Прогрес-бар показує кількість уже прочитаних сторінок, а також кількість сторінок, що залишилися до завершення книги. Відображення прогресу у відсотковому співвідношенні дозволяє користувачу швидко оцінити етап завершення читання та підтримує мотивацію до досягнення поставленої цілі.

Такий підхід дозволяє користувачу аналізувати власний прогрес та оцінювати ефективність читання. Окрім цього, реалізовано можливість редагування історії читання. Користувач може видаляти окремі записи про сесії читання, що забезпечує гнучке керування історією та дозволяє коригувати помилково додані дані.

Керування процесом читання відбувається за допомогою форми. Після натискання на кнопку старт запускається трекер який записує дату, час та

розраховує кількість прочитаних символів та потенційну швидкість читання. Після завершення читання відображається модальне вікно із привітання про прочитання книги та записується статус що книга вже є прочитаною. Для повторного читання потрібно видалити книгу із бібліотеки або очистити історію читання книги.

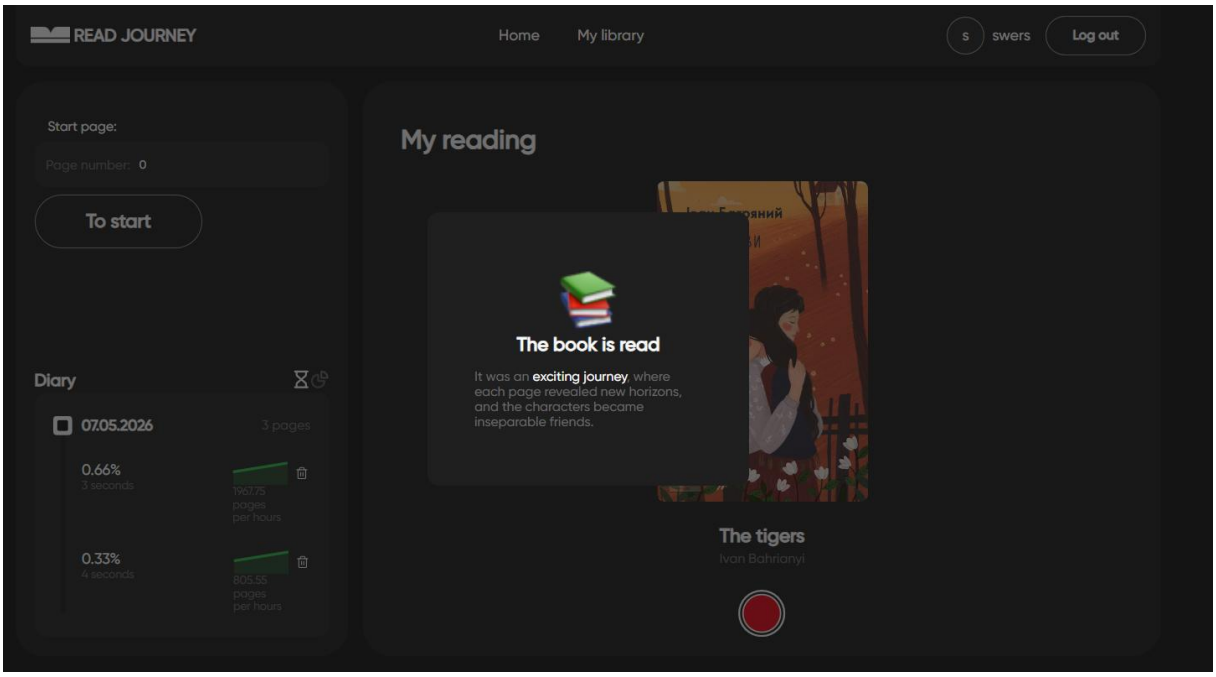

Рис. 3.20— модальне вікно із привітання про прочитання книги

Для повторного читання потрібно видалити книгу із бібліотеку або очистити історію читання книги.

ВИСНОВОК

Отже, під час виконання кваліфікаційної роботи на тему «Персональний книжковий трекер і рекомендаційна система для читачів» було спроєктовано прототип та розроблено вебплатформу, яка призначена для відстеження прогресу читання, управління особистою бібліотекою та отримання персоналізованих рекомендацій.

Для реалізації інтерфейсу та функціональної логіки було використано сучасні інструменти: React, TypeScript, Redux Toolkit та Styled Components, що забезпечило модульність, високу продуктивність та простоту підтримки проєкту.

Проєктування та розробка платформи включали кілька ключових етапів: аналіз ринку та існуючих рішень, визначення потреб і очікувань користувачів, формування вимог до системи, проєктування архітектури (FSD) та моделі взаємодії із сервером (REST API).

У межах роботи було реалізовано основні функціональні модулі платформи: систему авторизації користувачів, рекомендаційну систему, фільтрацію контенту, управління бібліотекою книг та трекер читання. Платформа підтримує створення власної бібліотеки, додавання та видалення книг, відстеження прогресу читання, перегляд статистики та взаємодію з рекомендаційним контентом.

У результаті було розроблено вебплатформу, яка демонструє можливість ефективного поєднання сучасних підходів фронтенд-розробки та реалізації персоналізованих сервісів для користувачів. Створене рішення є функціональним та адаптивним.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. React. Документація [Електронний ресурс] – Режим доступу: React
2. TypeScript. Документація [Електронний ресурс] – Режим доступу: Typescript
3. Styled Componet. Документація [Електронний ресурс] – Режим доступу: Styled Component
4. Redux Toolkit. Документація [Електронний ресурс] – Режим доступу: Redux-Toolkit
5. JavaScript. Документація [Електронний ресурс] – Режим доступу: Mdn
6. Світова статистика використання інтернету за типами пристроїв [Електронний ресурс] – Режим доступу: Статистика
7. UX/UI Design. Дизайн [Електронний ресурс] – Режим доступу: Desing UX/UI
8. What is REST API. REST API [Електронний ресурс] – Режим доступу: What is REST_API
9. FSD. Документація [Електронний ресурс] – Режим доступу: Feature-sliced.design
10. React development best practices [Електронний ресурс] – Режим доступу: Best-practices-react
11. Як стоворити сайт з нуля [Електронний ресурс] – Режим доступу: Stages-of-creating-a-web
12. Frisbie M. Professional JavaScript for Web Developers. 5 th ed. — John Wiley & Sons Ltd, 2024. — 1104 c.
13. Makarevich N. Advanced React: deep dives, investigations, performance and techniques. — DeveloperWay Publishing, 2023
14. O’Reilly Media. JavaScript: The Definitive Guide. 7 th ed. — O’Reilly Media, Inc., 2020. — ISBN 1491952024.

15. Martin R. Чистий кодер. Кодекс поведінки для професійних розробників / Роберт С. Мартін. — Фабула, 2023. — 256 с.

ДОДАТОК А

Код А.1 – useRegisterFormikSubmit

```
export const useRegisterFormikSubmit = () => {
  const dispatch = useAppDispatch();

  const navigate = useNavigate();

  const formik = useFormik({
    initialValues: {
      name: '',
      email: '',
      password: '',
    },
    validateOnChange: true,
    validateOnBlur: true,
    validationSchema: registrationScheme,

    onSubmit: values => {
      dispatch(registerThunk(values))
        .unwrap()
        .then(() => {
          navigate('/recommended');

          toast.success('You already registration ', {
            position: 'top-right',
            hideProgressBar: true,
            theme: 'dark',
```

```
        });
      })
      .catch(() => {
        toast.error('Something went worn.Try again!');
      });
  },
 });
 return formik;
};
```

Код А.2 – useLoginFormSubmit

```
export const useLoginFormSubmit = () => {
 const dispatch = useAppDispatch();

 const navigate = useNavigate();

 const formik = useFormik({
  initialValues: {
   email: '',
   password: '',
  },

  validationSchema: LoginScheme,

  onSubmit: value => {
   dispatch(loginThunk(value))
    .unwrap()
    .then(() => {
     navigate('/recommended');
```

```
          toast.success('You already login', {
            position: 'top-right',
            hideProgressBar: true,
            theme: 'dark',
          });
        })
        .catch(() => toast.error('Something went worn.Try again!'));
    },
  });

  return formik;
};
```

Код А.3 –loginScheme та registrationScheme

```
export const LoginScheme = Yup.object({
  email: Yup.string()
    .matches(
      /^\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,3}$/,
      'You have wrong email, try again'
    )
    .required('Required'),
  password: Yup.string()
    .min(7, 'Enter a valid Password*')
    .max(12, 'Enter a valid Password*')
    .required('Required'),
});

export const registrationScheme = Yup.object({
  name: Yup.string().required('Required'),
```

```
  email: Yup.string()
    .matches(
      /^\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,3}$/,
      'You have wrong email, try again'
    )
    .required('Required'),
  password: Yup.string()
    .min(7, 'Enter a valid Password*')
    .max(12, 'Enter a valid Password*')
    .required('Required'),
});
```

Код А.4 – PrivateRoute та PublicRoute

```
export const PrivateRoute = ({ children }: Children) => {
  const isLoggedIn = useAppSelector(isLoggedInSelect);
  const location = useLocation();

  if (!isLoggedIn) {
    return <Navigate to={'/login'} state={{ from: location }} />;
  }
  return children;
};

export const PublicRoute = ({ children }: Children) => {
  const isLoggedIn = useAppSelector(isLoggedInSelect);
  const location = useLocation();

  if (isLoggedIn) {
    return <Navigate to={location.state?.from ?? '/'} />;
```

```
  }
  return children;
};
```

Код А.5 – Axios Instance

```
export const instance = axios.create({
  baseURL: 'https://readjourney.b.goit.study/api',
});

export const setToken = (token: string) => {
  instance.defaults.headers.common.Authorization = `Bearer ${token}`;
};

export const clear = () => {
  instance.defaults.headers.common.Authorization = ``;
};
```