Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка мобільного додатка для організації та ведення персональної віртуальної бібліотеки

Виконав: студент IV курсу, групи СН-43
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Яцун М.О.
(підпис) (прізвище та ініціали)

Керівник _____ Литвиненко Я.В
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Липак Г.І.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Бойко І.В
(підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«27» січня 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Яцуну Мартину Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка мобільного додатка для організації та ведення персональної віртуальної бібліотеки

Керівник роботи Литвиненко Ярослав Володимирович, доктор технічних наук, професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*07*» *травня* 2025 року № *4/7-444*

2. Термін подання студентом завершеної роботи 28 червня 2025 р.

3. Вихідні дані до роботи Літературні та інтернет джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. РОЗДІЛ 1. Аналіз предметної області та обґрунтування вибору засобів розробки, 1.1 Актуальність теми, 1.2 Постановка задачі, 1.3 Порівняння існуючих мобільних рішень для управління книгами, 1.4 Порівняння технологій розробки мобільних додатків, 1.5 Обґрунтування вибору React Native, 1.6 Висновок до першого розділу, РОЗДІЛ 2 Практична реалізація мобільного додатку BookLoop, 2.1 Загальна структура додатку, 2.2 Інтерфейс користувача та навігація, 2.3 Екрани додатку та їхнє призначення, 2.4 Реалізація основної функціональності, 2.5 Висновки до другого розділу, РОЗДІЛ 3. Практичні аспекти реалізації та перевірки функціонування додатку, 3.3 Використані бібліотеки та інструменти, 3.4 Підсумкова оцінка реалізації, 3.5 Висновки до третього розділу, РОЗДІЛ 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Тема роботи, 2. Актуальність, 3. Причини розробки, 4. Мета проєкту, 5. Завдання проєкту, 6. Аналіз аналогів, 7. Порівняння функціоналу, 8. Вибір технологій, 9. Концепція додатку, 10. Фірмовий стиль, 11. Додаткова продукція, 12 Візуалізація бренду, 13. Інтерфейс користувача, 14 Можливості додатку 15. Демонстрація функцій, 16. Висновок

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин Віктор Степанович, доцент		

7. Дата видачі завдання 27 січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	27.01.2025	
2.	Підбір та опрацювання літературних джерел по темі кваліфікаційної роботи	10.03.2025–14.03.2025	
3.	Розробка мобільного додатка для організації та	16.03.2025–10.04.2025	
	ведення персональної віртуальної бібліотеки		
4.	Оформлення розділу «Аналіз предметної області та	21.04.2025–25.04.2025	
	обґрунтування вибору засобів розробки»		
5.	Оформлення розділу «Практична реалізація	08.05.2025–15.05.2025	
	мобільного додатку BookLoop»		
6.	Оформлення розділу «Практичні аспекти релізації	19.05.2025–23.05.2025	
	та перевірки функціонування додатку»		
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	02.06.2025–04.06.2025	
8.	Виконання завдання до підрозділу «Основи охорони	05.06.2025–7.06.2025	
	праці»		
9.	Оформлення кваліфікаційної роботи	08.06.2025-10.06.2025	
10.	Нормоконтроль	11.06.2025-13.06.2025	
11.	Перевірка на плагіат	18.06.2025	
12.	Попередній захист кваліфікаційної роботи	20.06.2025	
13.	Захист кваліфікаційної роботи	29.06.2025	
.			

Студент _____ Яцун М.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Литвиненко Я.В
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка мобільного додатку для організації та ведення персональної віртуальної бібліотеки // Кваліфікаційна робота освітнього рівня «Бакалавр» // Яцун Мартин Олегович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-43 // Тернопіль, 2025 // С. – 65 , рис. – 28, табл. – 3 , додат. 9 – , бібліогр. – 38.

Ключові слова: мобільний додаток, персональна бібліотека, організація книг, react native, інтерфейс користувача, кросплатформенна розробка, віртуальна бібліотека, зберігання інформації.

Кваліфікаційна робота присвячена розробці мобільного додатка для ведення персональної віртуальної бібліотеки.

У першому розділі обґрунтовано актуальність теми, визначено мету та завдання, проаналізовано існуючі рішення, порівняно технології мобільної розробки та обґрунтовано вибір React Native (RN) для реалізації проєкту.

У другому розділі розглянуто ключові етапи створення мобільного додатку BookLoop – від структури проєкту та навігації до реалізації основних функцій. Подано приклади коду, скріншоти інтерфейсу та описано логіку взаємодії. Отримано повноцінну функціональну основу для подальшого тестування у третьому розділі.

У третьому розділі кваліфікаційної роботи описано практичну реалізацію мобільного додатку BookLoop, проведено ручне тестування основних функцій (додавання книг, нотаток, фільтрація, офлайн-доступ), а також проаналізовано використані технології, такі як React Native, Expo та AsyncStorage. Об'єктом дослідження є програмний продукт для організації віртуальної бібліотеки, предметом – методи розробки та тестування його основних функцій.

ANNOTATION

Development of a Mobile Application for Organizing and Maintaining a Personal Virtual Library// Qualification work of the educational level «Bachelor» // Yatsun Martyn // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-43 // Ternopil, 2025 // P. - 65, fig. – 28 , tabl. – 3 , annexes. – 9 , references – 38 .

Keywords: mobile application, personal library, book organization, React Native, user interface, cross-platform development, virtual library, data storage.

The qualification thesis is dedicated to the development of a mobile application for managing a personal virtual library.

The first chapter justifies the relevance of the topic, defines the goal and objectives, analyzes existing solutions, compares mobile development technologies, and provides a rationale for choosing React Native (RN) for implementing the project.

Chapter Two covered the key stages of developing the BookLoop mobile application – from project structure and navigation to the implementation of core features. Code snippets, interface screenshots, and logic explanations were provided. As a result, a fully functional foundation has been established for further testing and evaluation in Chapter Three.

The third chapter of this qualification paper presents the practical implementation of the BookLoop mobile application. It includes manual testing of core features (book addition, notes, filtering, offline access) and an analysis of the technologies used, such as React Native, Expo, and AsyncStorage.

The object of the study is a mobile application for managing a personal library, while the subject is the process of its development and functionality testing.

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

API (Application Programming Interface) – інтерфейс прикладного програмування, набір методів для взаємодії між різними частинами програмного забезпечення.

CSS (Cascading Style Sheets) – каскадні таблиці стилів, що відповідають за оформлення елементів інтерфейсу.

HTML (HyperText Markup Language) – мова розмітки для створення структури вебсторінок та інтерфейсів.

RN (React Native) – фреймворк для кросплатформеної розробки мобільних додатків з використанням JavaScript.

UI (User Interface) – інтерфейс користувача, тобто візуальна частина додатку, з якою взаємодіє користувач.

UX (User Experience) – досвід користувача при взаємодії з додатком, включає зручність, інтуїтивність, емоційне сприйняття.

API (англ. Application Programming Interface) – Інтерфейс прикладного програмування

ЗМІСТ

ВСТУП

У сучасному світі цифрових технологій спостерігається значне зростання попиту на інструменти, які дозволяють ефективно структурувати особисті знання та управляти інформацією. Незважаючи на велику кількість навчальних і розважальних ресурсів, читання книг залишається однією з провідних форм інтелектуального розвитку, самовдосконалення та проведення дозвілля. Проте багато користувачів стикаються з труднощами в організації своєї бібліотеки, відстеженні прогресу в читанні, систематизації нотаток та створенні списків літератури для подальшого вивчення.

Розробка зручного і багатофункціонального мобільного додатка для управління особистою віртуальною бібліотекою є одним із основних завдань у галузі сучасних цифрових рішень. Головна мета цієї кваліфікаційної роботи полягає у створенні такого додатку за допомогою технології React Native, що гарантує кросплатформенну сумісність та оптимізацію процесу розробки.

Розроблений додаток надає користувачам зручні інструменти для організації та управління інформацією про книги. Він дозволяє сортувати видання за статусами, відслідковувати прогрес читання, створювати нотатки, додавати книги до улюблених списків і формувати добірки літератури для майбутнього читання. У процесі розробки було проведено глибоке дослідження, яке допомогло визначити ключові вимоги до функціоналу. Реалізовано основні елементи інтерфейсу та логіку роботи додатка. Також проведено тестування для перевірки коректності роботи функцій і відповідності поставленим завданням.

Для реалізації проєкту були використані сучасні технічні методи: компонентна архітектура, контексти для управління станом, а також структурована організація коду зі стилями, винесеними в окремий файл. Створений додаток має практичне значення та перспективи для подальшого розвитку.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РОЗРОБКИ

1.1 Актуальність теми

У сучасному цифровому світі мобільні додатки стали основним засобом щоденного доступу до інформації. Згідно з інформацією DataReportal, у 2025 році середній користувач смартфона проводитиме приблизно 4 години 37 хвилин на день, користуючись своїм пристроєм [1]. При цьому понад 89% цього часу припадає саме на мобільні додатки, що підкреслює їхнє важливе значення в повсякденному житті [2]. За даними Statista, у 2024 році кількість завантажень мобільних додатків перевищила 218 мільярдів, що на 7% більше порівняно з попереднім роком [3].

Окреме місце займають додатки, призначені для читання та самоосвіти. Як зазначає Digital Trends, дедалі більше користувачів обирають цифрові інструменти для ведення персональної бібліотеки, фіксації прочитаного і планування. Це пов’язано з їхньою мобільністю, зручним користуванням та адаптивними інтерфейсами [4]. Водночас огляд користувацького досвіду, опублікований на Medium, підтверджує зростаючий інтерес до функціоналу, який дозволяє організовувати читацьку активність, робити нотатки і відстежувати прогрес у читанні [5].

Розробка застосунків для зручного управління особистою бібліотекою є актуальним рішенням, що відповідає запитам сучасного цифрового суспільства. Такий інструмент не лише дозволяє організувати читацькі звички, але й мотивує до регулярного читання, глибшого осмислення літератури та збереження унікальних вподобань кожного користувача. У майбутньому подібні рішення можуть сприяти створенню нової культури взаємодії з книгами, ефективно поєднуючи технології з індивідуалізованим підходом до читача.

1.2 Постановка задачі

Ця робота присвячена створенню мобільного додатку, що допоможе користувачам організувати власну віртуальну бібліотеку. Ідея полягає в тому, щоб кожен міг зручно фіксувати прочитане, планувати нові книги, стежити за прогресом, додавати нотатки й формувати список бажаного.

Перш ніж переходити до реалізації, передбачено аналіз уже існуючих рішень. Це дасть змогу краще зрозуміти, яких саме функцій бракує користувачам, і що можна вдосконалити. Далі буде спроєктована структура програми, визначено основні можливості, і реалізовано ключові частини інтерфейсу.

Користувач зможе додавати нові книги, редагувати наявні або видаляти непотрібне. Для кожної позиції можна буде зазначити назву, автора, жанр, опис, кількість сторінок і статус читання. Також реалізується фільтрація за статусами: «Читаю», «Прочитано», «На потім» і «Закинув «. Окремий розділ буде присвячений улюбленим книгам. Вся інформація зберігатиметься локально, тож застосунок працюватиме навіть без доступу до Інтернету.

Особливу увагу в проєкті буде приділено тому, наскільки зручно користувачу взаємодіяти з додатком. Йдеться не лише про зрозумілу навігацію, а й про можливість налаштувати деякі елементи під себе. Як зазначає у статті, коли користувачі відчувають, що інтерфейс «пристосований» до них, це підвищує зацікавленість і бажання користуватись додатком [6]. А в огляді підкреслюється, що саме адаптивність і гнучкість часто стають вирішальними факторами в цифрових бібліотеках [6].

Усе це дає змогу сприймати майбутній застосунок не просто як набір функцій, а як персоналізований інструмент, що адаптується до потреб користувача. Це покликано створити приємний досвід, який викликатиме бажання повертатися до нього знову.

1.3 Порівняння існуючих мобільних рішень для управління книгами

Перш ніж приступити до розробки програми, доцільно ознайомитися з уже наявними рішеннями на ринку з подібним функціоналом. Це дозволяє краще зрозуміти, що працює ефективно, а що потребує доопрацювання в існуючих додатках, а ще визначити, як удосконалити майбутній продукт.

У цьому розділі представлено огляд найпопулярніших мобільних додатків для управління особистою бібліотекою, серед яких є Goodreads, Libib. Ми проаналізуємо функції цих додатків, а саме зручність використання, наявність офлайн-режиму, можливості створення нотаток, відстеження прогресу читання та підтримку української локалізації.

Аналіз користувацького досвіду з урахуванням реальних умов його використання допомагає виявити недоліки в наявних додатків і врахувати їх під час створення нового застосунку [7].

Результати аналізу буде представлено у вигляді порівняльної таблиці, яка наочно продемонструє різницю між додатками та підтвердить доцільність розробки нового продукту.

1.3.1 Аналіз функціональних можливостей застосунку Goodreads

Goodreads це один з найвідоміших сервісів для читачів, що дозволяє створювати свої книжкові полиці, фіксувати прочитане, ділитися відгуками та оцінками, а також спілкуватися з іншими користувачами. Сервіс належить компанії Amazon і має веб-версію та додаток як й для Android й для iOS . Його офіційний сайт містить детальний опис функціоналу та підтримуваних можливостей [8]

Основний функціонал Goodreads включає створення профілю користувача, додавання книг за назвою або ISBN, ведення власного списку прочитаного, додавання відгуків, складання списків (наприклад: «Want to

Read», «Currently Reading», «Read»), участь у літературних клубах та щорічних читальних викликах. Інтерфейс додатку побудований навколо соціального підходу – користувач бачить новини, рецензії, оцінки друзів або популярні книги. На рисунку 1.1 зображено початковий екран мобільного додатку Goodreads. Він демонструє фокус на англомовну аудиторію, інтеграцію з обліковими записами Amazon та мінімалістичний стиль оформлення з акцентом на швидкий доступ до персонального кабінету.

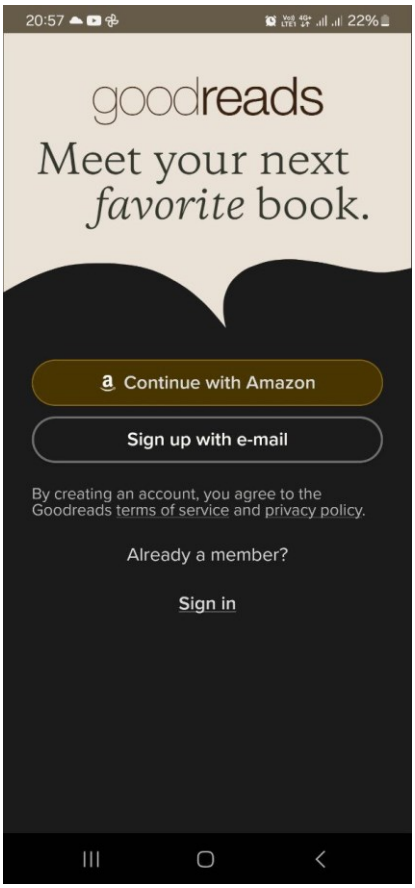

Рисунок. 1.1 – Екран входу в додаток Goodreads

Такий спосіб є зручним для базового ведення читацького обліку, однак обмежує користувача у можливості персоналізації. Відсутність підтримки нотаток, своїх категорій або розширених фільтрів може не задовольнити потреби тих, хто прагне більш гнучкий та впорядкований підхід до організації власної бібліотеки. Це важливо для користувачів, які бажають фіксувати особисті враження чи ключові думки прямо в межах додатку.

На рисунку 1.2 показано інтерфейс для додавання книги у бібліотеку. Структура взаємодії є досить простою, а саме книга просто потрапляє на одну з базових полиць без можливості гнучкого структурування, добавлення нотаток чи створення своїх статусів.

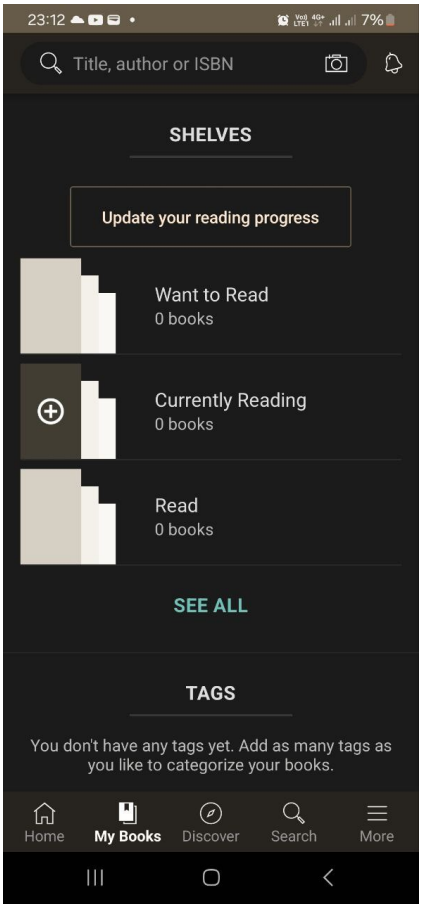

Рисунок 1.2 – Додавання книги до бібліотеки в Goodreads

Водночас платформа Goodreads має кілька обмежень. Сервіс призначений на англомовну аудиторію та книги. Хоча є наявність базового функціоналу для відстеження прогресу читання, він реалізований у вигляді окремих оновлень без гнучкого відображення статистики в інтерфейсі або системи нагадувань. Додаток також не підтримує офлайн режим, не дозволяє додавати нотатки до книги як цілісного об’єкта, а ще не дає змоги персоналізувати структуру бібліотеки під індивідуальні потреби. Згідно з UX-дослідженням Télécom Paris, користувачі відзначали складну навігацію, застарілий інтерфейс, нестачу фільтрів і персоналізації [9].

1.3.2 Аналіз функціональних обмежень застосунку Libib

Libib – мобільний додаток для створення каталогів книг, фільмів та іншого медіа. Його функціонал орієнтований переважно на архівне зберігання, а не щоденне користування. Безкоштовна версія підтримує до 5000 записів і 100 колекцій, тоді як функції на кшталт редагування даних чи розширеного керування колекціями доступні лише у платних версіях [10]. На рис. 1.3 показано головний екран із мінімалістичним інтерфейсом, без списку книг і з обмеженою інтерактивністю. Додаток не підтримує статуси читання, нотатки чи трекінг прогресу, що обмежує його функціональність для активних читачів

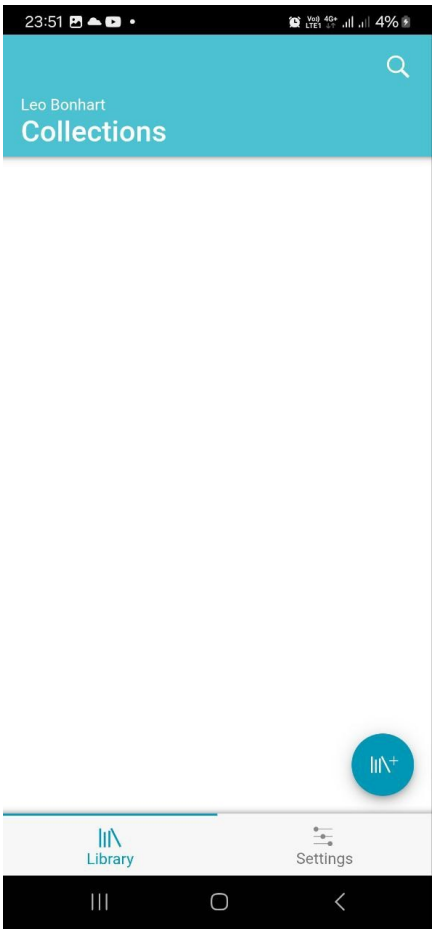

Рисунок 1.3 – Екран порожньої бібліотеки у Libib

На рис. 1.4 показано форму ручного додавання книги. Щоб додати новий запис, користувачеві потрібно самостійно ввести назву, автора, видавництво,

дату публікації, ISBN та кількість сторінок. Додаток не має відстеження прогресу читання, добавлення нотаток чи рейтингу книги. Після збереження користувач бачить лише базову картку з можливістю редагування або видалення.

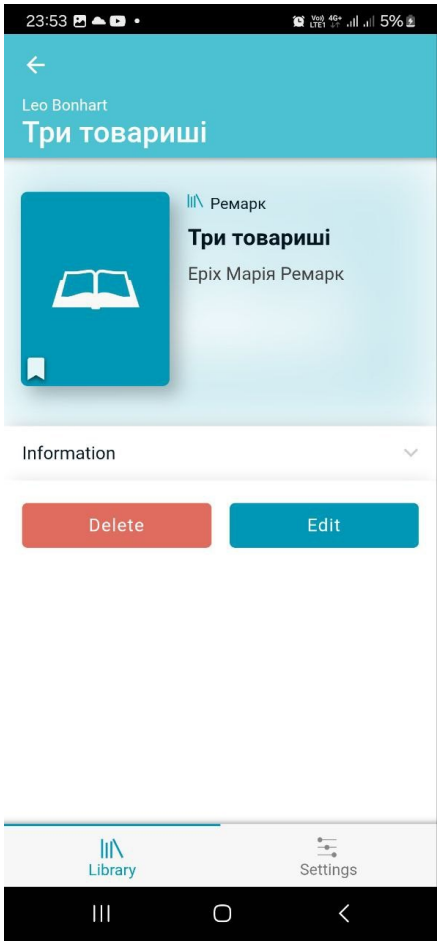

Рисунок 1.4 – Інтерфейс доданої книги у Libib

Серед основних недоліків додатку можна виокремити:

- Відсутні статуси читання («Читаю», «Прочитано» тощо).
- Відсутність відслідкованості прочитаних сторінок.
- Не можна додати оцінку або рейтинг.
- Відсутні нотатки до книги.
- Немає категорій «улюблені» або «бажані».
- Немає україномовного інтерфейсу.
- Не підтримується персоналізація бібліотеки.

Отже хоча Libib і пропунує базову каталогізацію, його функціоналі можливості не відповідають потребам сучасного користувача, який прагне взаємодії з контентом, зокрема можливість вести читацькі записи, відстеження прогресу читання та записувати власні думки. Це підкреслює актуальність створення таких додатків, як BookLoop, які поєднують зручність, адаптивність та розширені можливості управління особистою бібліотекою.

1.3.3 Порівняльна характеристика функціоналу Goodreads, Libib та BookLoop

Для кращого розуміння переваг і недоліків існуючих рішень було зроблено порівняння найбільш популярних мобільних додатків, призначених для організації особистих бібліотек. У таблиці нижче показано зіставлення можливостей Goodreads, Libib і розробленого в межах даного проєкту додатку BookLoop. В таблиці 1.1. наведено порівняльну характеристику функціоналу додатків.

Таблиця 1.1 – Порівняння функціоналу книжкових додатків

Функціонал/Додаток	**Goodreads**	**Libib**	**BookLoop**
1	**2**	**3**	**4**
Статуси читання (читаю, прочитано тощо)	Так	Ні	Так
Прогрес читання (сторінки / %)	Обмежено	Ні	Так
Оцінка книги (рейтинг)	Так	Ні	Так
Нотатки до книги	Ні	Ні	Так
Додавання до улюбленого	Ні	Ні	Так
Список бажаних книг (wishlist)	Так	Ні	Так
Можливість додавання вручну	Так	Так	Так
Сканування ISBN	Так	Частково	не реалізовано

Продовження таблиці 1.1

1	2	3	4
Офлайн-режим	Ні	Так	Так
Локалізація українською	Ні	Ні	Так
Простота інтерфейсу	Складний	Застарілий	Мінімалістичний
Категорії / кастомні фільтри	Обмежено	Ні	Так
Повна безкоштовність	Ні(є преміум)	Так	Так

Goodreads надає певний функціоналу безкоштовно, проте розширені можливості, як-от аналітичні функції або інтеграції з іншими сервісами, доступні лише за підпискою або через додаткові платформи (наприклад, Amazon Kindle) [8] . Натомість від цього, BookLoop створюється як повністю безкоштовний інструмент із фокусом на доступність і простоту.

Таким чином, існуючі сервіси мають значні обмеження або перевантаженість, що ускладнює їхнє застосування для простих щоденних задач, пов’язаних із читанням і нотатками. У результаті виникла ідея розробити власний додаток BookLoop, який поєднує найнеобхідніші можливості в одному інтуїтивному інтерфейсі [10].

1.4 Порівняння технології розробки мобільних додатків

У сучасному цифровому середовищі розробка мобільних додатків набула стратегічного важливого значення у сфері програмного забезпечення [11]. Залежно від обраного підходу до розробки, можна виділити нативні, гібридні та кросплатформені рішення.

Нативна розробка передбачає створення окремих додатків для платформ Android та iOS із застосуванням таких мов програмування таких як Kotlin або Swift. Цей підхід забезпечує високу продуктивність, але вимагає більше часу, ресурсів й витрат на створення та підтримку окремих кодових баз [12].

Гібридний підхід до розробки, що базується на використанні HTML, CSS та JavaScript, дозволяє створювати додатки, які функціонують у веб-оболонці. Цей підхід сприяє зниженню витрат на розробку, проте зазвичай має недоліки у швидкодії та обмеженому доступі до нативного функціоналу пристрою. Яскравими прикладами таких технологій є Cordova та Ionic [13].

Кросплатформена розробка поєднує переваги попередніх підходів. Завдяки таким фреймворкам, як RN, Flutter, Xamarin тощо, розробники можуть створювати єдиний код для кількох платформ. RN розроблений на JavaScript і дозволяє створювати нативоподібний інтерфейс користувача (UI) з високою продуктивністю [14]. Серед ключових переваг є підтримка компонентного підходу, гнучкість, значна спільнота та наявність численних готових модулів, зокрема для роботи з API [15]. React Native поєднує знайому зручність веброзробки з доступом до нативних можливостей пристрою, що робить його особливо привабливим для створення сучасних мобільних застосунків.

У межах цієї роботи використовується саме RN, оскільки він дає змогу створити зручний UI з можливістю адаптації для двох ключових мобільніх платформ. При цьому завдяки використанню HTML та CSS у компонентних стилях можливо забезпечити персоналізацію інтерфейсу без необхідності дублювання коду [16].

Важливою перевагою React Native є підтримка нативних модулів, що дозволяє взаємодіяти з камерами, сповіщеннями, геолокацією та іншими API пристрою. Завдяки цьому можна реалізовувати функції, як-от сканування ISBN або надсилання локальних повідомлень без втрати продуктивності [11].

Окрім цього, велика спільнота React Native та активна підтримка бібліотек помітно прискорюють процес розробки й значно полегшують розв'язання технічних питань.

Отже, з огляду на баланс між продуктивністю, зручністю розробки, гнучкістю UI та доступом до API, вибір React Native стає оптимальним рішенням для реалізації персонального органайзера читання.

Таблиця 1.2 – Порівняння підходів до мобільної розробки

Технологія	**Тип розробки**	**Продуктивність**	**Можливості та обмеження**
Нативна	Платформозалежна	Висока	Повний доступ до API, найвища швидкодія, окремий код для кожної ОС
React Native	Кросплатформна	Середньо-висока	Один код для Android/iOS, доступ до більшості нативних функцій
Flutter	Кросплатформна	Висока	Власний рушій інтерфейсу, висока продуктивність, мова Dart
Cordova Ionic	Гібридна (WebView)	Низька-середня	Веб-інтерфейс, обмежений доступ до API, повільніша робота

Як бачимо з таблиці, нативна розробка демонструє найвищий рівень продуктивність, проте вимагає значно більше ресурсів, оскільки потребує створення окремих рішень для кожної платформи.

Гібридні фреймворки, хоч і дозволяють пришвидшити розробку, часто поступаються в ефективності роботи додатків та у доступі до нативного функціоналу. У цьому контексті React Native постає як оптимальний компроміс, оскільки поєднує переваги кросплатформності з можливістю гнучкого налаштування інтерфейсу та підключення до API [17].

У наступному підпункті буде наведено аргументацію на користь вибору саме React Native для реалізації мобільного додатку.

1.5 Обґрунтування вибору React Native

Серед фреймворків для кросплатформеної розробки мобільних додатків помітне місце займає RN, розроблений компанією Meta (колишньою Facebook), який дозволяє створювати додатки під Android та iOS з використанням JavaScript і вебтехнологій (HTML, CSS).

RN забезпечує оптимальний баланс між продуктивністю та зручністю розробки, підтримуючи орієнтований на користувацький інтерфейс із використанням нативних компонентів платформи. На відміну від гібридних рішень наприклад, Cordova, він дозволяє безпосередньо взаємодіяти з API платформи, що підвищує продуктивність і адаптивність інтерфейсу [15].

У дослідженні 2020 року порівнювали створення двох схожих додатків за допомогою React Native та Flutter. React Native виділявся кращою інтеграцією з нативними функціями пристрою, зручністю використання UI-компонентів і нижчим порогом входу для веброзробників. Водночас Flutter демонстрував трохи вищу продуктивність під час запуску [18].

На рисунку 1.5 представлено порівняльну характеристику Flutter і React Native за такими критеріями: продуктивність, спільнота, мова програмування, доступ до нативних API, UI-компоненти та інші параметри.

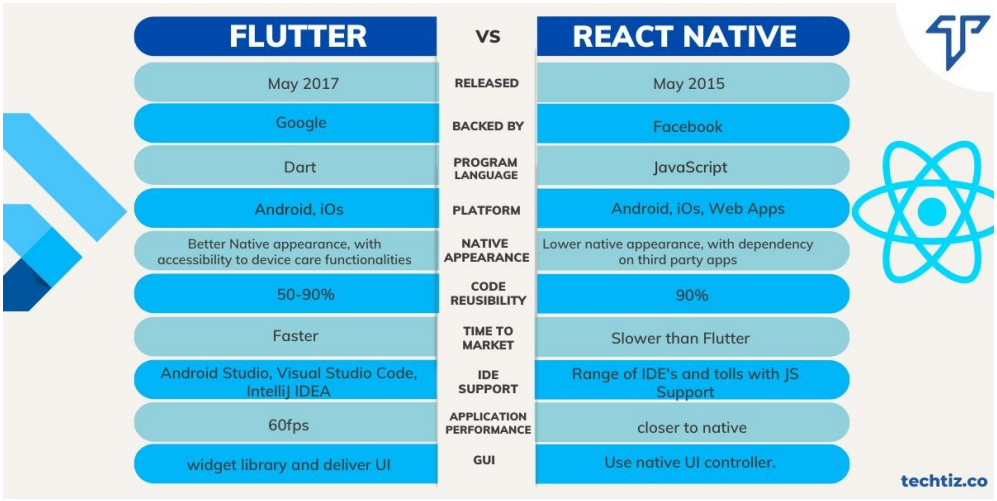

Рисунок 1.5 – Порівняльна характеристика Flutter та React Native

У статті за 2023 рік підкреслюється, що RN характеризується широкою спільнотою, активною екосистемою бібліотек та вдосконаленою підтримкою CI/CD процесів, що суттєво полегшує процес розробки і сприяє швидкому випуску продукту. Окрім того, використання JavaScript надає веброзробникам простіший старт у порівнянні з використанням Dart у Flutter [19].

Отже, беручи до уваги необхідність швидкої реалізації, доступ до нативного функціоналу, можливість використання знайомих інструментів (Visual Studio Code, Android Studio) та високий рівень підтримки з боку спільноти, вибір React Native є обґрунтованим рішенням, зокрема в контексті гнучких підходів до розробки програмних систем [17].

1.6 Висновок до першого розділу

У першому розділі кваліфікаційної роботи здійснено ґрунтовний аналіз сучасних тенденції розвитку мобільних, що дозволило обґрунтувати актуальність розроблення застосунку для управління персональною віртуальною бібліотекою. Було визначено мету, завдання, об'єкт та предмет дослідження, а також проведено аналіз функціональних можливостей існуючих сервісів для обліку прочитаної літератури Goodreads та Libib. Здійснене порівняння показало, що наявні рішення не повністю задовольняють потреби україномовних користувачів, особливо в контексті офлайн роботи, персоналізації, створення нотаток і гнучкої фільтрації книг.

У межах розділу було також досліджено різні підходи до розробки мобільних застосунків – нативний, гібридний і кросплатформений та здійснено порівняльний аналіз фреймворків. На основі результатів обґрунтовано вибір React Native як оптимального інструменту для реалізації проєкту, що забезпечує баланс між продуктивністю, швидкістю розробки та зручністю підтримки. У наступному розділі буде приділено детальну увагу структурі мобільного додатку, архітектуру, реалізовану функціональність та елементи UI/UX-дизайну.

РОЗДІЛ 2. ПРАКТИЧНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ BOOKLOOP

2.1 Загальна структура додатку

Мобільний застосунок BookLoop був створений з дотриманням принципів модульності, масштабованості та розподілу відповідальностей між компонентами, що відповідає принципам сучасної архітектури додатків на базі React Native [20]. Така структура не лише робить проєкт зрозумілішим для подальшої підтримки, а й спрощує розширення функціоналу та багаторазове використання вже реалізованих рішень. У застосунку було використано платформу Expo, яка спрощує процес розробки, тестування та збірки мобільного застосунку.

Файлова структура проєкту побудована за функціональним принципом: ключові частини застосунку винесено в окремі папки відповідно до їхніх завдань. Це відповідає поширеним практикам організації коду в React Native-проєктах і допомагає підтримувати порядок та зрозумілу логіку в архітектурі додатку [21] . На рисунку 2.1 показано, як виглядає основна структура директорій, що відображає функціональний поділ компонентів у проєкті BookLoop.

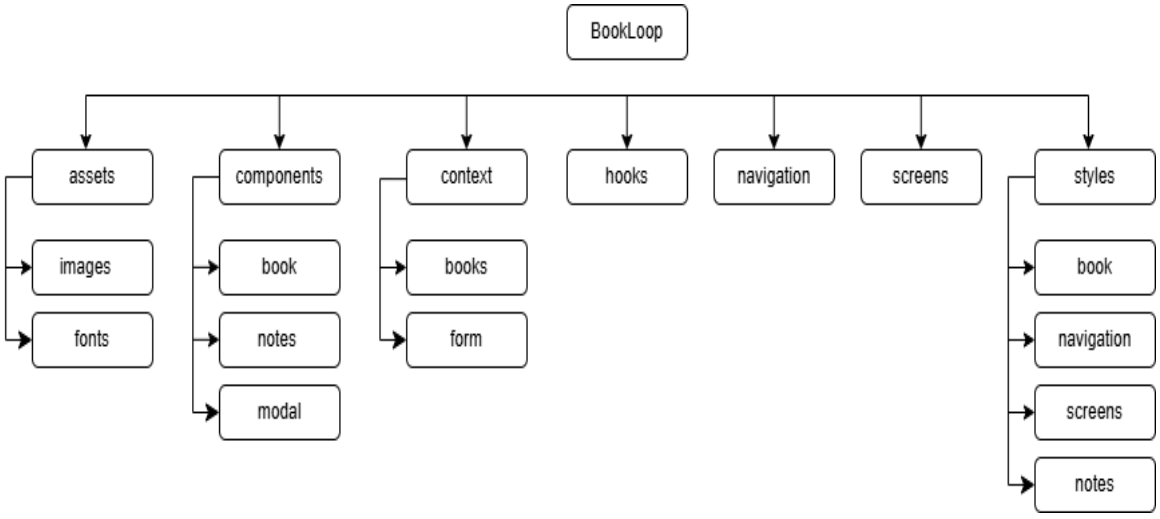

Рисунок 2.1 – Деревоподібна структура папок і модулів проєкту BookLoop

Опис основних папок проєкту:

- assets – папка де зберігаються статичні ресурси, наприклад зображення, шрифти та іконки. Вони імпортуються до компонентів для візуального оформлення додатку.
- components – папка, в якій зібрано багаторазові елементи інтерфейсу користувача. Вони групуються за функціональністю в окремі підпапки (наприклад, book/, modal/, notes/ тощо). Тут є реалізації карток книг, кнопок, модальних вікон, свайп-елементів і фільтрів. Ці компоненти використовуються в різних частинах додатку, що дозволяє зменшити повторення коду й полегшити його підтримку.
- context – папка, що відповідає за управління глобальним станом застосунку через Context API. Тут розміщені провайдери, які зберігають і обробляють дані, пов’язані з книгами, нотатками, бажанками та формами введення. Такий підхід дозволяє централізовано реагувати на оновлення, видалення записів, фільтрацію та інші дії, що спрощує керування станом у межах усього проєкту..
- hooks – містить користувацькі React-хуки для винесення повторюваної логіки. Наприклад, useBookDetails отримує дані про книгу за ID, а useFormContext обробляє введення у формах. Використання хуків робить архітектуру чистішою, сприяє повторному використанню логіки та підвищує гнучкість у розробці функціоналу..
- navigation – містить логіку, пов’язану з переходами між екранами додатку. Вона включає окремі конфігурації для стекової навігації (StackNavigator) та навігації з вкладками (TabNavigator). Усе реалізовано з використанням бібліотеки @react-navigation/native, з урахуванням стилістичних налаштувань і параметрів переходів, які адаптовані до потреб інтерфейсу..
- screens – Містить основні екрани додатку, кожен з яких відповідає за окрему частину функціоналу. Серед них – головна бібліотека, перегляд деталей книги, додавання й редагування, робота з нотатками, бажанками, перегляд статистики тощо. Детальніше про кожен з екранів йдеться у підрозділі 2.3.

- styles – папка для зберігання стилів інтерфейсу. Кожен файл стилів має назву, що відповідає компоненту або екрану (bookDetailsStyles.js, noteCardStyles.js тощо). Винесення стилів в окремі файли дозволяє централізовано керувати візуальним оформленням та підтримувати єдиний дизайн.

Така структура проєкту дозволяє швидко орієнтуватися у коді, легко додавати нову функціональність та забезпечує стабільність упродовж усього циклу розробки. Вона відповідає сучасним підходам до розробки мобільних додатків на базі React Native [22]. Та принципам адаптивного вибору архітектури програмного забезпечення в умовах гнучкої розробки [23].

2.2 Інтерфейс користувача та навігація

Інтерфейс користувача мобільного додатку BookLoop розроблений з орієнтацією на простоту, інтуїтивність та комфортність використання. Головна ціль – забезпечити швидкий доступ до ключових розділів: віртуальної бібліотеки, нотаток, бажаного списку та обраних книг.

Під час проєктування враховано основні принципи мобільної ергономіки, зокрема:

- можливість керування додатком однією рукою.
- адаптивність інтерфейсу до різних розмірів екранів.
- збереження контексту при переході між екранами.
- використання контрастної палітри та мінімалістичного дизайну, відповідно до рекомендацій щодо кросплатформеного UI в React Native [24].

Для реалізації навігації між екранами використано бібліотеку @react-navigation/native, яка дозволяє створювати гнучкі та масштабовані структури переходів [25].

На рисунку 2.2. зображено вміст папки navigation, яка містить конфігурації навігаційних структур:

Ці структури забезпечують зручний перехід між екранами та логічну організацію інтерфейсу мобільного додатку.

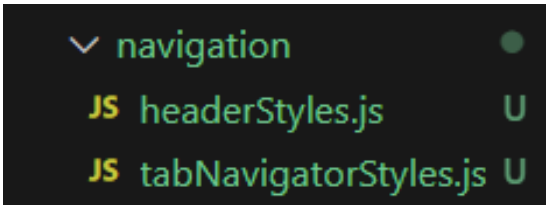

Рисунок 2.2 – Вміст папки navigation

У програмі використано дві головні схеми для навігації, які забезпечують зручний перехід поміж екранами та логічну структуру інтерфейсу.

- Stack-навігація (StackNavigator.js) – забезпечує плавні переходи між екранами, з можливістю повернення до попередніх сторінок.
- Tab-навігація (TabNavigator.js) – відповідає за відображення нижньої панелі з вкладками, що забезпечують доступ до основних розділів додатку: «Головна», «Додати», «Категорії».

Кожна вкладка таб-навігації являє собою окремий стек (Stack Navigator), що забезпечує можливість створювати переходи всередині одного розділу. Наприклад, у розділі «Головна» користувач може перейти на екран деталей книги, а потім до екрана редагування чи нотаток, зберігаючи контекст поточного розділу.

На рисунку 2.3 представлено нижню навігаційну панель (tab-bar), яка завжди доступна для користувача:

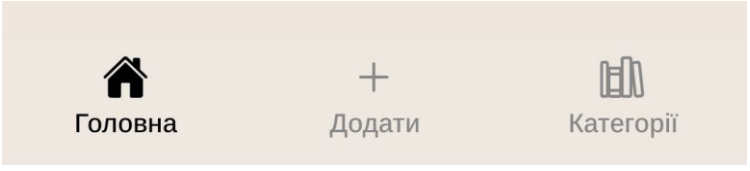

Рисунок 2.3 – Нижня навігаційна панель (tab-bar)

Ця панель є основним засобом навігації між головними розділами застосунку.

Такий підхід до поєднання вкладок і стеків дозволяє раціонально організувати маршрутизацію, забезпечуючи гнучкість і логічність переходів [26]. Нижня навігаційна панель (tab-bar) надає постійний доступ до ключових екранів додатка. Кожна вкладка представлена відповідною іконкою та текстовим підписом, а для активної вкладки застосовується кольорове виділення, що сприяє підвищенню зручності орієнтації для користувачів.

2.3 Екрани додатку та їхнє призначення

Цей підрозділ аналізує ключові екрани мобільного застосунку BookLoop, які становлять основу взаємодії користувача з його персональною віртуальною бібліотекою. У відповідності до сучасних принципів UX – дизайну, кожен екран реалізує конкретну функцію – від перегляду та фільтрації книг до створення нотаток. Такий поділ функціоналу на логічно окремі екрани сприяє підвищенню зручності користування і підвищує ефективність навігації у мобільному застосунку [27].

2.3.1 Головний екран (HomeScreen)

Головний екран є центральним елементом мобільного додатку BookLoop і з’являється першим після запуску програми. Тут користувач може швидко отримати доступ до своєї віртуальної бібліотеки. Інтерфейс цього екрана розроблено з акцентом на зручність та ефективність, що дозволяє легко орієнтуватися та оперативно користуватися ключовими функціями додатку [14].

Головний екран дозволяє:

- Переглядати всі додані книги.
- Шукати книги за ключовими словами (назва або автор).
- Фільтрувати книги за статусом читання.

- Сортувати список книг за різними критеріями.
- Переглядати візуальний прогрес читання.

На рисунку 2.4 представлено приклад головного екрана мобільного додатку. Тут відображені різноманітні книги з актуальним станом читання та показниками прогресу.

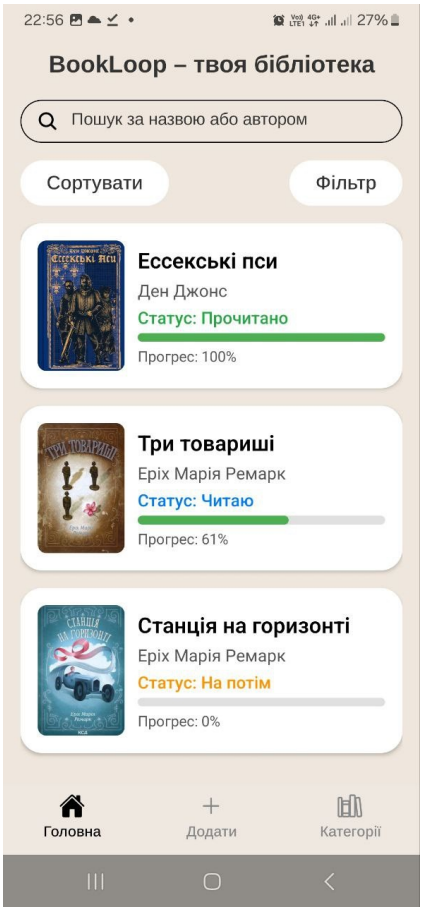

Рисунок 2.4 – Головний екран додатку

Даний екран слугує головною точкою доступу для виконання більшості дій користувача, включаючи перегляд книг, зміну їхнього статусу або перехід до розділу із нотатками. Інтерфейс головного екрана містить наступні складові елементи:

- Заголовок додатку – фіксований напис «BookLoop – твоя бібліотека».
- Поле пошуку – розміщене під заголовком, дозволяє шукати книги за назвою або автором. Пошук працює в режимі реального часу.

- Кнопки «Сортувати» та «Фільтр» – відкривають відповідні модальні вікна, де користувач може обрати критерії сортування або статус фільтрації.
- Список книг представлений у вигляді окремих карток, кожна з яких містить зображення обкладинки, назву книги, ім’я автора, статус читання (наприклад, «Читаю» або «Прочитано») та індикатор прогресу у відсотках.

На даному екрані представлено три книги, кожна з яких має різний статус: «Прочитано» (позначено зеленим кольором), «Читаю» (синій колір) та «На потім» (помаранчевий колір). Кожна картка книги оснащена індикатором прогресу, який відображає відсоток завершеності читання, що дозволяє користувачеві легко визначити поточний етап своєї читацької активності. У верхній частині інтерфейсу розміщені інструменти пошуку і кнопки фільтрації, тоді як нижній блок займає панель навігації з виділеною вкладкою «Головна».

За натисканням на картку книги відкривається екран деталей, де можна побачити всю інформацію про книгу, змінити статус, додати нотатку тощо.

2.3.2 Екран деталей книги (BookDetailsScreen)

Після натискання на книгу зі списку на головному екрані відкривається екран деталізованої інформації про неї. Його головне завдання – надати користувачу всю необхідну інформацію про обрану книгу та забезпечити різні опції для взаємодії, такі як зміна статусу, додавання до улюблених чи ведення нотаток.

Основні функції екрана:

- Ознайомлення з описом книги, її автором, жанром і видавництвом.
- Зміна статусу книги (наприклад, «Читаю», «Прочитано»).
- Відображення кількості прочитаних сторінок і загального прогресу
- Виставлення рейтингу.
- Додавання й перегляд нотаток.
- Редагування або видалення книги зі списку.

Екран поділяється на три ключові зони: верхню (загальна інформація), середню (функціональні показники) і нижню (дії користувача). У прикладі на рисунку 2.5 зображено верхню частину екрана з деталями книги. У цій зоні користувач знайомиться з вибраною книгою завдяки візуальному та текстовому представленню, що включає обкладинку, назву, автора, жанр, видавництво та короткий опис.

Рисунок 2.5 – Відображення загальної інформації про книгу та опису

На екрані представлено зображення обкладинки, під яким структуровано ключові дані: назва книги, ім'я автора, жанр та інформація про видавництво. Функціональна кнопка «Додати до улюблених» забезпечує можливість збереження книги у спеціальну категорію, що сприяє швидкому доступу до неї в майбутньому. Короткий опис твору надається у згорнутому форматі з опцією розгортання, що дозволяє користувачеві зручно ознайомитися зі змістом перед початком читання.

На рисунку 2.6 зображено центральну частину екрана книги, де розташовано функціональні елементи: статус, кількість сторінок, індикатор прогресу, рейтинг та приклад доданої нотатки.

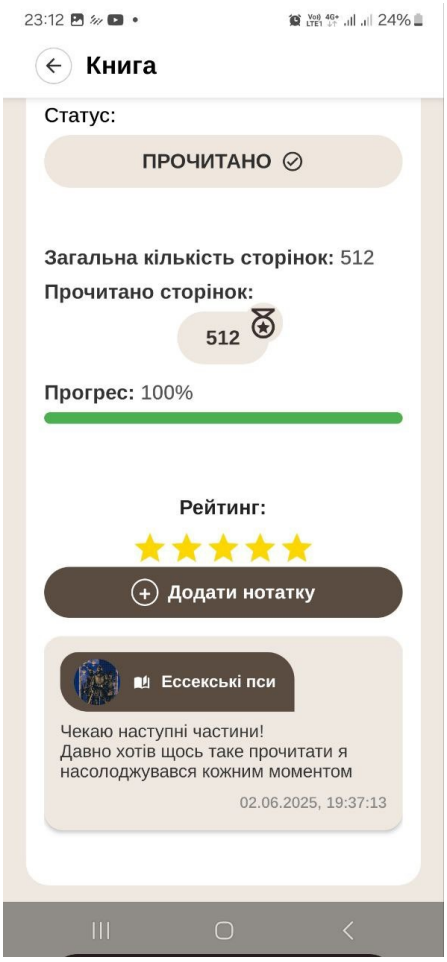

Рисунок 2.6 – Статус, прогрес читання, рейтинг та нотатка до книги

Цей фрагмент екрана дозволяє переглядати:

- Статус читання книги (наприклад, «Прочитано»).
- Загальну й прочитану кількість сторінок.
- Індикатор прогресу у відсотках (візуально та чисельно).
- Рейтинг книги, представлений у вигляді зірок.
- Додану нотатку, відображену у вигляді картки з датою, текстом і мініатюрою книги.

На рисунку 2.7 показано нижню частину екрана, яка містить основні кнопки для взаємодії з книгою: зміна статусу, редагування, видалення.

Ці елементи забезпечують користувачеві швидкий доступ до основних дій з книгою.

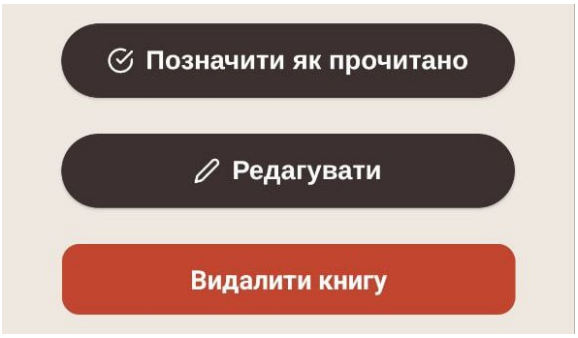

Рисунок 2.7 – Кнопки для взаємодії з книгою: статус, редагування, видалення

У цьому блоці користувач може:

- Позначити книгу як прочитану (за потреби).
- Відкрити екран редагування, щоб змінити дані про книгу.
- Видалити книгу – дія підтверджується вручну, а сама кнопка стилізована червоним кольором для виділення.

Таким чином, екран BookDetailsScreen є повноцінною інформаційною й функціональною сторінкою книги, де користувач може не лише переглядати дані, а й активно взаємодіяти з ними.

2.3.3 Екран додавання книги (AddBookScreen)

Екран додавання книги надає користувачам можливість створювати нові записи у своїй віртуальній бібліотеці. Його головна мета – забезпечити зручний та інтуїтивно зрозумілий процес введення інформації про книгу. Поля впорядковані за тематичними групами, а інтерфейс виконано в мінімалістичному стилі для максимально швидкої та комфортної взаємодії.

Основні можливості:

- Введення основної інформації: назва, автор, жанр.
- Додавання опису твору.

- Вказання кількості сторінок, видавництва та ISBN.
- Вибір або зміна обкладинки книги.
- Збереження книги натисканням кнопки «Додати книгу».

Нижче на рисунку 2.8 показано першу частину екрана, де користувач заповнює текстові поля: назву книги, ім’я автора, жанр та опис.

Рисунок 2.8 – Поля введення основної інформації про книгу

Всі поля оформлені чітко і структуровано, із виразно виділеними заголовками, такими як «Назва книги», «Автор», «Жанр», що значно полегшує процес навігації та заповнення інформації. Поле для опису здатне підтримувати введення довгих текстів і відображає весь зміст без потреби у прокрутці, що є суттєвою перевагою для ознайомлення з анотацією перед її збереженням.

На рисунку 2.9 представлено нижню частину екрана для додавання книги. Тут користувач вводить кількість сторінок, назву видавництва, 13-значний ISBN, а також додає або змінює обкладинку книги.

Ці дані становлять важливу частину бібліографічної інформації та забезпечують можливість точного визначення книги.

Рисунок 2.9 – Вибір обкладинки, технічні поля та кнопки дій

Після заповнення текстових полів можна вибрати відображення обкладинки за допомогою кнопки «Обрати обкладинку», яка надає доступ до галереї пристрою. Завершення заповнення форми здійснюється через кнопку «Додати книгу», що зберігає всі дані в систему. Усі кнопки оформлені зі збереженням чіткої стилістики та ієрархії – важливі дії виділяються темним фоном для акцентування уваги.

Таким чином, екран AddBookScreen поєднує простоту введення з усіма ключовими параметрами, необхідними для створення повноцінного запису книги

2.3.4 Екран категорій (CategoriesScreen)

Екран категорій виконує роль навігаційного інструменту, що надає користувачам швидкий доступ до різноманітних розділів їхньої бібліотеки. Його основне завдання – організувати матеріали у логічні категорії, щоб спростити пошук і систематизацію книг відповідно до їхнього статусу, важливості чи пов'язаних дій, таких як редагування нотаток. Зокрема, з цього екрана можна:

- Здійснювати перехід до.
- Списку улюблених книг.
- Розділу нотаток.
- Бажанок (списку книг, які користувач хоче прочитати або придбати).
- Повного списку усіх книг.

На рисунку 2.10 показано вигляд цього екрана з кнопками для переходу до відповідних розділів.

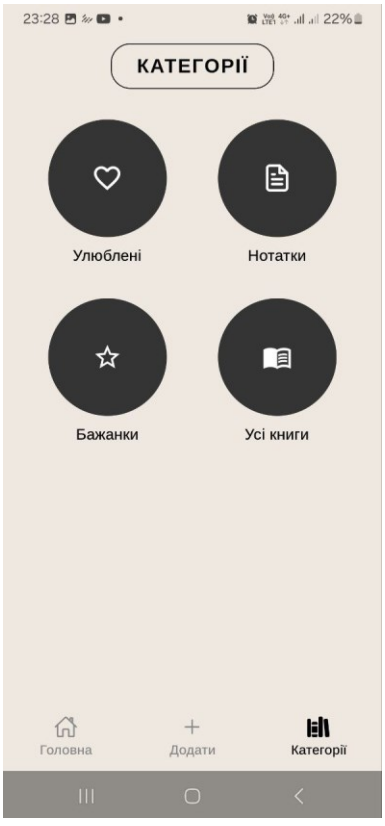

Рисунок 2.10 – Екран категорій із доступом до основних розділів

Кожна категорія представлена у вигляді круглої кнопки з іконкою та підписом, що забезпечує інтуїтивність навігації. Елементи розташовані симетрично, а активна вкладка «Категорії» підсвічується на нижній панелі навігації. Завдяки цьому користувач може легко орієнтуватися у структурі додатку та комфортно взаємодіяти з великим обсягом контенту.

2.3.5 Екран улюблених книг (FavoritesScreen)

Цей екран містить перелік книг, які користувач позначив як улюблені. Він є елементом функціоналу категорій і забезпечує зручний доступ до найважливіших або найцінніших творів.

Призначення екрана:

- Перегляд списку улюблених книг.
- Швидкий перехід до деталей книги.
- Візуальне відображення статусу читання та прогресу.
- Повідомлення про порожній список, якщо ще нічого не додано.

Нижче на рисунку 2.11 зображено екран у випадку, коли користувач ще не додав жодної книги до категорії «Улюблені». У центрі екрана відображається зображення емодзі з текстом «Немає улюблених книг», що візуально підкреслює порожній статус розділу.

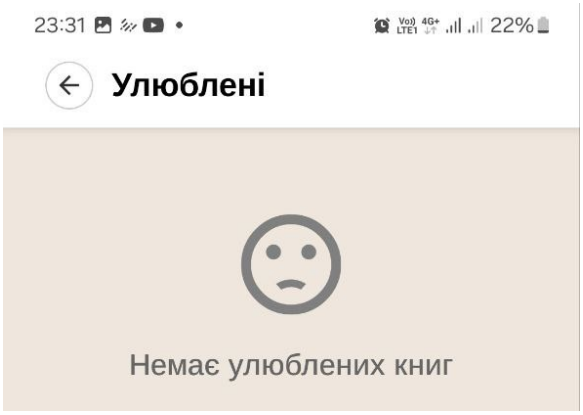

Рисунок 2.11 – Повідомлення про відсутність улюблених книг

Це допомагає запобігти непорозумінням і мотивує користувача додати книги до свого списку.

На рисунку 2.12 продемонстровано приклад відображення улюблених книг, якщо вони вже додані. Кожна книга представлена у вигляді окремої картки, подібно до головного екрана.

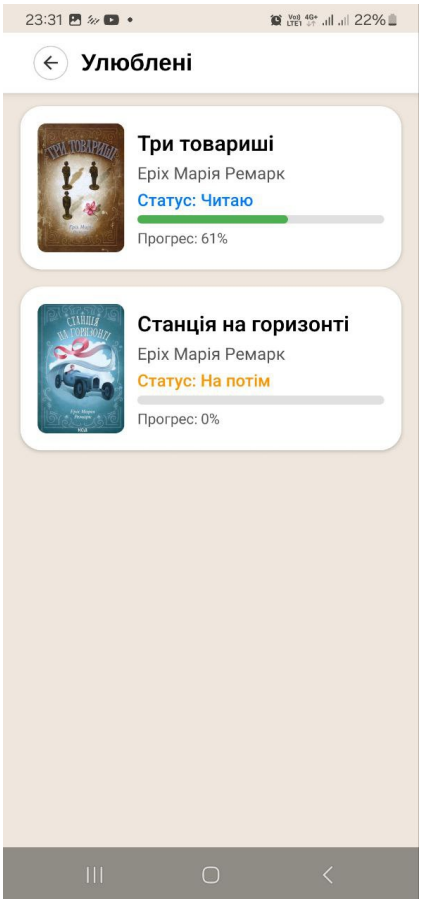

Рисунок 2.12 – Список улюблених книг з індикаторами статусу та прогресу

Кожна картка містить:

- Обкладинку книги.
- Назву й автора.
- Статус читання (наприклад, «Читаю», «На потім») з відповідним кольоровим маркуванням.
- Індикатор прогресу у відсотках.

Цей формат дозволяє швидко зорієнтуватися в тому, які книги є найважливішими для користувача, та в якому стані перебуває їх прочитання.

2.3.6 Екран бажанок (WishlistScreen)

Екран бажань передбачає окрему категорію для збереження книг, які користувач поки не має, але планує прочитати або придбати. Це свого роду список для майбутнього, який дозволяє впорядковувати цікаві видання та повертатися до них у зручний час.

.Функціональність:

- Перегляд списку бажаних книг.
- Додавання нової бажанки вручну.
- Перехід до деталей бажаної книги (у разі редагування).
- Відображення стану «порожній список».

На рисунку 2.13 показано повідомлення про відсутність бажанок. Цей стан відображається автоматично, якщо користувач ще не додав жодної книги до категорії. У центрі екрана відображається лаконічне повідомлення з іконкою, яке повідомляє, що список бажаних книг поки що порожній.

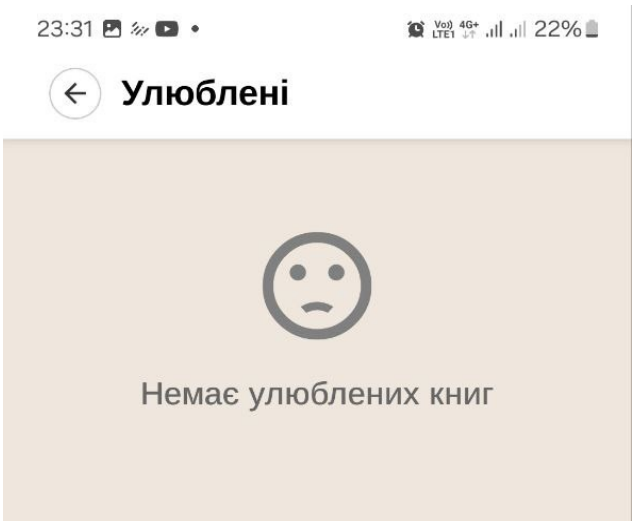

Рисунок 2.13 – Повідомлення про порожній список бажанок

Це заохочує користувача розпочати створення власного переліку книг, які планується прочитати.

На рисунку 2.14 показаний заповнений список бажаних книг. Кожна з «бажанок» представлена у вигляді компактної картки з назвою, автором та обкладинкою.

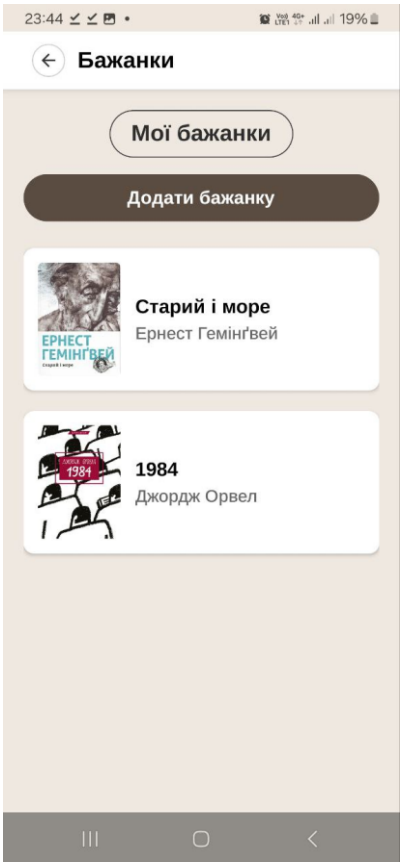

Рисунок 2.14 – Відображення списку бажаних книг

У верхній частині екрана знаходиться заголовок «Мої бажанки» та кнопка «Додати бажанку», яка відкриває форму для створення нової книги, аналогічну формі, використаній у AddBookScreen. Дизайн карток бажанок спрощений: вони не містять індикатора прогресу та статусу читання, адже ці книги поки залишаються непрочитаними.

При натисканні кнопки «Додати бажанку» відкривається стандартна форма, де користувач заповнює наступні дані: назву книги, автора, жанр, ISBN, кількість сторінок та додає зображення обкладинки. Уся введена інформація зберігається в окремому списку бажанок, який не залежить від основної

бібліотеки книг. Після придбання або прочитання книги її можна перемістити до основної бібліотеки за допомогою функції редагування записів.

2.3.7 Екран усіх книг (AllBooksScreen)

Екран «Усі книги» слугує загальним оглядом усіх книжок, які користувач фактично додав до своєї бібліотеки, виключаючи бажані. Це означає, що тут представлені лише ті книги, які вже перебувають у користувача – незалежно від їхнього статусу читання (наприклад, «Читаю», «Прочитано», «На потім» тощо). На рисунку 2.15 представлений екран «Усі книги» у вигляді сітки, де відображені обкладинки, назви, автори, статуси та рейтинги.

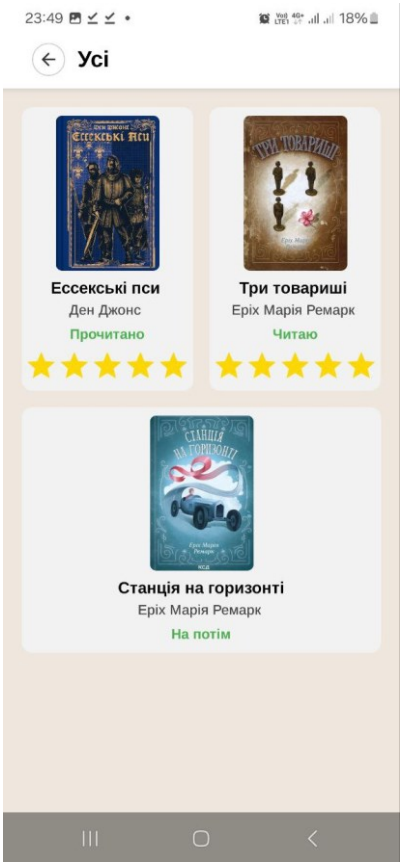

Рисунок 2.15 – Екран «Усі книги» з рейтингами, статусами і обкладинками

Цей екран функціонує як своєрідна візуальна галерея, що забезпечує швидкий доступ до особистої колекції у зручному й компактному форматі.

Основні функції екрана:

- Відображення всіх книг, незалежно від статусу.
- Виведення основної інформації: обкладинка, назва, автор.
- Відображення статусу читання та рейтингу, якщо він встановлений.
- Швидкий доступ до детальної інформації по кожній книзі (через натискання).

Цей екран зручний для перегляду бібліотеки та оперативного доступу до будь-якої книги.

2.3.8 Екран нотаток до книг (BookNotesScreen)

Екран «Нотатки» забезпечує зручний доступ до всіх коментарів, які користувач залишив до книг. Цей функціонал реалізовано як окремий розділ, що дозволяє зосередитись саме на читацьких враженнях, думках і цитатах, незалежно від статусу книги.

Призначення:

- Перегляд усіх створених нотаток у зручному списку.
- Фільтрація нотаток за конкретною книгою.
- Швидкий перехід до книги, з якою пов’язана нотатка.
- Відображення дати створення.

Для досягнення більшої інформативності та логічності в поданні інформації, кожна нотатка подана у форматі окремої картки, оснащеної основними атрибутами. Такий спосіб організації сприяє швидкому сприйняттю змісту та полегшує навігацію між взаємопов'язаними книжковими записами.

На картці нотатки представлено:

- Мініатюру обкладинки книги.
- Назву книги.
- Текст нотатки.
- Дату й час створення.

На рисунку 2.16 представлений екран нотаток у розширеному вигляді. У верхній частині розташовані кнопки для переходу до всіх нотаток і селектор фільтрації за книгою.

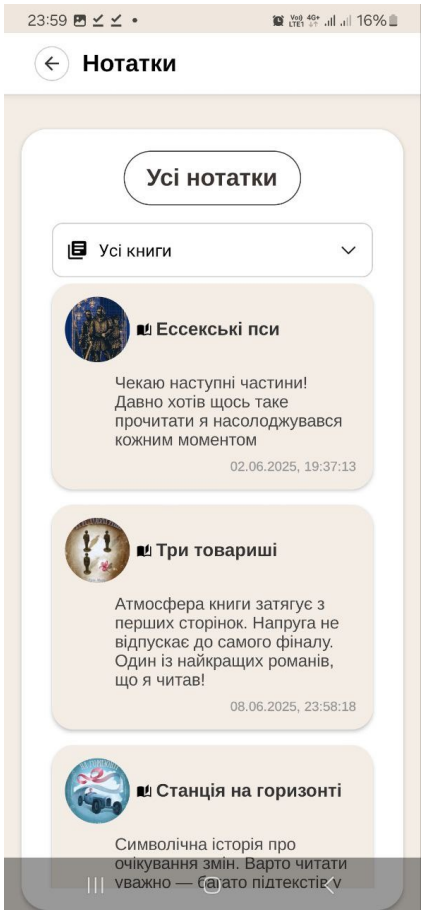

Рисунок 2.16 – Екран нотаток з фільтром і прикладами коментарів до книг

Цей інтерфейс забезпечує зручну роботу з читацькими нотатками, дозволяючи переглядати, редагувати та застосовувати їх у подальшому аналізі прочитаного матеріалу. Для вибору конкретної книги реалізований фільтр у вигляді випадаючого списку, що допомагає зосередитися на певному творі.

2.4 Реалізація основної функціональності

На поточному етапі розробки впроваджено основну логіку взаємодії користувача з додатком BookLoop. Основний акцент зроблено на функціоналі для додавання книг, зміні їх статусів, роботі з нотатками, а також

удосконаленні користувацького досвіду за рахунок використання анімацій, модальних вікон і жестів, відповідаючи сучасним тенденціям у мобільній розробці [16].

Однією з ключових функцій є можливість додати нову книгу, що реалізовано у файлі AddBookScreen.js. Користувач вводить основну інформацію про книгу, після чого дані обробляються й книга додається до контексту. Залежно від вибраного режиму, доданий елемент потрапляє або до основної бібліотеки, або до списку бажань. Використання LayoutAnimation і scrollToEnd забезпечує плавну анімацію під час прокрутки до кінця списку після додавання нової книги, що значно підвищує рівень зручності для користувача [16].На рисунку 2.17 зображено функцію handleAddBook, яка відповідає за цю логіку.

```
const handleAddBook = () => {
  const newBook = {
    title,
    author,
    genre,
    description,
    image,
    totalPages: parseInt(totalPages) || 0,
    currentPage: 0,
    publisher,
    isbn13
  };

  if (isWishlistMode) {
    addToWishlist(newBook);
    navigation.navigate('Бажанки');
  } else {
    addBook(newBook);
    navigation.navigate('Головна');
  }

  resetForm();
  setSearch('');
};
```

Рисунок 2.17 – Додавання книги до бібліотеки або бажанок

Для забезпечення зручності користувачів інтегровано функцію вибору обкладинки книги. Після вибору зображення із галереї автоматично виконується плавна прокрутка до нижньої частини форми. Це рішення дає змогу одразу побачити результат взаємодії з інтерфейсом, суттєво покращуючи

досвід користувача (UX) [16]. На рисунку 2.18 продемонстровано процес обробки обкладинки та запуску відповідної анімації.

```
const handlePickImage = async () => {
  const uri = await pickImage();
  if (uri) {
    LayoutAnimation.configureNext(LayoutAnimation.Presets.easeInEaseOut);
    setImage(uri);

    //Прокрутити вниз, щоби було видно кнопки
    setTimeout(() => {
      scrollRef.current?.scrollToEnd({ animated: true });
    }, 300);
  }
};
```

Рисунок 2.18 – Завантаження обкладинки з анімацією прокрутки

Додатково впроваджено можливість модального сортування книг за назвою або прогресом читання через компонент ModalSort.js. Під час вибору потрібної опції список автоматично оновлюється завдяки використанню функції setSortOption у контексті. Такий підхід сприяє декларативному управлінню інтерфейсом [23]. На рисунку 2.19 відображено застосування цієї функціональності у компоненті сортування.

```
{options.map((option) => (
  <Pressable
    key={option.value}
    onPress={() => onSelect(option.value)}
    accessibilityLabel={`Сортувати за ${option.label}`}
  >
    <Text
      style={[
        styles.modalItem,
        selected === option.value && styles.selectedItem
      ]}
    >
      {option.label}
    </Text>
  </Pressable>
))}
```

Рисунок 2.19 – Реалізація модального сортування книг

Система управління нотатками базується на використанні Swipeable з react-native-gesture-handler. Компонент SwipeableNote.js дозволяє видаляти або редагувати нотатки за допомогою свайпів, відповідаючи сучасним принципам

дизайну мобільних інтерфейсів [28]. Анімації для додавання нотаток створені за допомогою Animated.View, що додає динамічності та інтерактивності. На рисунку 2.20 показано функції renderLeftActions і renderRightActions, які виконують ці операції.

```
const renderRightActions = () => (
  <TouchableOpacity onPress={() => onEdit(note)} style={styles.swipeEdit}>
    <Feather name="edit-2" size={20} color="#fff" />
  </TouchableOpacity>
);

const renderLeftActions = () => (
  <TouchableOpacity onPress={() => onDelete(note)} style={styles.swipeDelete}>
    <MaterialCommunityIcons name="trash-can-outline" size={20} color="#fff" />
  </TouchableOpacity>
);
```

Рисунок 2.20 – Свайп-дії для редагування та видалення нотатки

Також важливу роль відіграє функція updateBookStatus, яка забезпечує можливість змінювати статус книги між такими категоріями, як: Читаю, На потім, Прочитано, Закинув. Для оптимізації процесу оновлення була застосована useCallback, що дозволяє зменшити кількість зайвих перерендерів компонентів і покращити загальну продуктивність [23]. На рисунку 2.21 представлено реалізацію цієї функції.

```
const updateBookStatus = useCallback((id, status) => {
  setBooks(prev =>
    prev.map(book =>
      book.id === id ? { ...book, status } : book
    )
  );
}, []);
```

Рисунок 2.21 – Функція оновлення статусу книги (updateBookStatus)

Реалізація кожної функції будується на основі практик автоматизованого створення UI, реактивної архітектури та статичного аналізу. Це забезпечує високу підтримуваність і можливість масштабування додатка в перспективі [16].

2.5 Висновки до другого розділу

У другому розділі кваліфікаційної роботи було детально розкрито послідовну реалізацію основних етапів розробки мобільного застосунку BookLoop, призначеного для створення персональної віртуальної бібліотеки.

Насамперед, у підрозділі 2.1 було проаналізовано структуру проєкту. Розроблений логічний поділ програми на екрани, компоненти, стилі та контексти сприяв зручності у масштабуванні та підтримці коду. Надано схему архітектури із ґрунтовним поясненням функціональних ролей її ключових частин.

У підрозділі 2.2 зосереджено увагу на системі навігації. Було розроблено нижню панель навігації з вкладками, верхній заголовок із функціями фільтрації та пошуку, а також реалізовано використання спеціальних шрифтів для забезпечення візуальної узгодженості дизайну.

Підрозділ 2.3 стосувався компонування окремих екранів інтерфейсу. Детально описано дизайн головного екрана бібліотеки, сторінки деталей книги, форм для додавання й редагування книг, а також перегляду списку бажанок. Включено аспекти створення блоків розмітки, стилізації та забезпечення інтуїтивного користувацького досвіду.

Підрозділ 2.4 зосереджувався на реалізації ключових функцій застосунку. Серед них такі можливості, як додавання книг, завантаження обкладинок із застосуванням анімацій, сортування контенту в модальному вікні, свайпи для роботи з нотатками та оновлення статусу книг. Усі функції супроводжувались прикладами коду із відповідними технічними поясненнями.

Таким чином, у другому розділі було сформовано фундаментальне функціональне ядро мобільного застосунку BookLoop – від проєктування структури та дизайну до побудови логіки взаємодії з книжковим контентом. Це створює передумови для переходу до фінального етапу розробки, який включає тестування, перевірку надійності збереження даних і оцінку отриманих результатів, що будуть докладно висвітлені у наступному розділі.

РОЗДІЛ 3. ПРАКТИЧНІ АСПЕКТИ РЕЛІЗАЦІЇ ТА ПЕРЕВІРКИ ФУНКЦІОНУВАННЯ ДОДАТКУ

3.1 Тестування додатку BookLoop

Мета тестування полягала у перевірці стабільної роботи додатку BookLoop, коректності збереження даних, зручності користування інтерфейсом і можливості функціонування без підключення до інтернету. Основна увага була приділена аналізу поведінки додатку в умовах реального використання, включаючи додавання та редагування книг, роботу з нотатками, налаштування фільтрів та доступ до функцій у режимі офлайн.

Таблиця 3.1 – Результати ручного тестування додатку BookLoop

Сценарій тестування	**Дії користувача**	**Очікувана поведінка**	**Фактичний результат**
1	**2**	**3**	**4**
Додавання книги	Заповнення форми, натиск «Зберегти»	Книга з’являється у списку з усіма даними	Книга збережена правильно
Редагування та видалення книги	Зміна опису або жанру, подальше видалення	Дані оновлюються, книга зникає після видалення	Оновлення й видалення працюють
Додавання нотаток	Створення нотатки до книги	Нотатка зберігається, видно після перезапуску	Нотатка зберігається коректно
Робота фільтрів	Вибір статусу читання	Виводяться лише книги з вибраним статусом	Фільтрація працює точно

Продовження таблиці 3.1

1	2	3	4
Офлайн-режим	Перезапуск без інтернету	Дані доступні, зміни зберігаються локально	Повна робота без мережі

Тестування проводилося вручну на кількох фізичних пристроях із різними характеристиками (зокрема, на телефонах Samsung та Xiaomi). Для кожного ключового сценарію було описано типову дію користувача, очікувану поведінку додатку та фактичний результат.

Особливу увагу приділяли послідовності виконання дій та стабільності їх результатів при повторному відтворенні сценаріїв.Було проведено перевірку наступних сценаріїв: додавання нової книги з основними даними, зміну статусу книги, редагування та видалення записів, створення нотаток, перегляд статистики прогресу, використання фільтрів і сортування, а також тестування правильного функціонування елементів інтерфейсу, таких як кнопки, поля введення та свайпи.

Усі ці дії перевірялися як за наявності інтернет-з'єднання, так і в автономному режимі, щоб оцінити роботу додатку в умовах відсутності доступу до мережі.

Такий підхід є типовою практикою серед мобільних розробників. Які часто віддають перевагу ручному тестуванню, навіть за наявності автоматизованих інструментів. Результати масштабного дослідження, проведеного на основі аналізу понад 1600 мобільних додатків, свідчать про те, що ручне тестування продовжує залишатися ключовим методом виявлення помилок у процесі розробки мобільних додатків, особливо на етапі фінальної перевірки функціональності в умовах реального використання [29].

3.2 Офлайн режим і збереження даних

Одним із головних завдань при розробці мобільного додатку BookLoop було забезпечення стабільної роботи у офлайн-режимі. Це означає, що користувачі повинні мати можливість здійснювати такі дії, як перегляд, додавання, редагування та видалення книг і нотаток, навіть без доступу до Інтернету. Для реалізації цієї вимоги в додатку був інтегрований механізм локального збереження даних, на основі технології AsyncStorage. Це частина екосистеми React Native, яка підтримується спільнотою та вважається одним із рекомендованих рішень, для зберігання даних у форматі ключ-значення [29]

AsyncStorage є асинхронним механізмом локального зберігання даних, який функціонує за принципом пар «ключ–значення» та дозволяє зберігати інформацію безпосередньо на пристрої користувача. Через те, що цей інструмент підтримує виключно рядкові значення, об'єкти необхідно попередньо серіалізувати у формат JSON за допомогою методу JSON.stringify(), а для їх подальшого використання – виконувати десеріалізацію за допомогою JSON.parse(). Таке рішення виявляється ефективним для мобільних застосунків, які не залежать від серверної чи хмарної інфраструктури, але потребують стабільного та надійного зберігання даних між робочими сесіями [29].

У застосунку BookLoop використання AsyncStorage дозволяє реалізувати збереження такого роду даних:

- список книг разом із полями: назва, автор, статус, обкладинка, жанр, кількість сторінок, улюблене, прогрес читання.
- нотатки, які зберігаються як вкладені масиви усередині об’єктів книг;
- статус читання (наприклад, «Читаю», «Прочитано»).
- список бажаних книг (wishlist) – окремий масив об'єктів.

Під час запуску додатка здійснюється зчитування даних із локального сховища. На рисунку 3.1 продемонстровано реалізацію функції loadBooks, що дозволяє отримати збережений JSON-масив, перетворити його на список

об'єктів, локалізувати значення статусів і ініціалізувати відсутні поля за потреби.

```
const loadBooks = async () => {
  const saved = await AsyncStorage.getItem('books');
  if (saved) {
    const parsed = JSON.parse(saved);
    const updated = parsed.map(book => ({
      ...book,
      status:
        book.status === 'reading' ? 'Читаю' :
        book.status === 'later' ? 'На потім' :
        book.status === 'finished' ? 'Прочитано' :
        book.status === 'abandoned' ? 'Закинув' :
        book.status === 'не вказано' ? 'На потім' :
        book.status,
      isFavorite: book.isFavorite || false,
      notes: book.notes || []
    }));
    setBooks(updated);
  }
};
```

Рисунок 3.1 – Завантаження книг з локального сховища

Система також забезпечує автоматичне збереження внесених змін. Усі оновлення списку книг або списку бажань моментально серіалізуються у формат JSON і записуються в AsyncStorage. На рисунку 3.2 показано, як відбувається процес збереження змін у масиві книг.

```
useEffect(() => {
  AsyncStorage.setItem('books', JSON.stringify(books));
}, [books]);
```

Рисунок 3.2 – Збереження змін до локального сховища

Для оцінки надійності механізму зберігання було проведено тестування в різноманітних умовах:

- Після примусового закриття застосунку через системну панель Android всі книги та нотатки були збережені.
- Після перезапуску пристрою програма успішно зчитала дані з AsyncStorage.

- У повністю офлайн-режимі, без доступу до Wi-Fi чи мобільних даних, додаток зберігав усі свої функції: можна було додавати, редагувати, фільтрувати книги й нотатки без жодних обмежень.

Завдяки використанню AsyncStorage забезпечується стабільне локальне збереження даних, що дозволяє працювати з додатком у будь-який час, незалежно від наявності Інтернету. Це значно підвищує надійність і зручність користування BookLoop.

3.3 Використані бібліотеки та інструменти

У цьому розділі наведено список основних бібліотек, використаних під час створення мобільного додатку. Назви зазначено згідно з офіційними іменами npm-пакетів, причому деякі з них мають префікс @. Цей префікс вказує на простір імен (scope), до якого належить відповідна бібліотека в системі управління пакетами Node.js. Наприклад, назва @react-navigation/native означає, що пакет native входить до складу простору імен react-navigation.

Середовище Expo значно полегшило початковий етап налаштування проєкту, запуск на реальних пристроях, перевірку функціональності та створення APK-збірок. Використання інструменту Expo CLI дозволило уникнути складних нативних налаштувань, що дало змогу зосередитися на розробці бізнес-логіки та інтерфейсу для користувача [17]. У рамках цього середовища були використані такі модулі:

- expo-image-picker – для вибору зображень обкладинок із галереї.
- expo-font – для підключення кастомних шрифтів, зокрема Arimo.
- expo-device і expo-status-bar – для взаємодії з системними параметрами пристрою.

Для навігації між екранами застосовано стек бібліотек:

- @react-navigation/native.
- @react-navigation/native-stack.
- @react-navigation/bottom-tabs.

Ці бібліотеки були використані для реалізації функціоналу вкладок, організації переходів між основними екранами (такими як головний екран, додавання книги, перегляд, редагування та список бажань). Унікальні ідентифікатори для книг, нотаток і списків бажань створювалися за допомогою:

- react-native-uuid.
- react-native-get-random-values.

Це гарантувало правильну обробку кожного об'єкта, незалежно від кількості та послідовності дій користувача. Для візуалізації інтерфейсу було використано:

- lucide-react-native – бібліотека сучасних SVG-іконок з легким дизайном.
- @expo/vector-icons – набір класичних іконок, зокрема Ionicons, які використовувалися для вкладок і кнопок дій.

React-native-gesture-handler, яка забезпечує створення свайп-жестів над нотатками для їх редагування чи видалення. Для відображення модальних вікон з функцією редагування нотаток застосовувалася бібліотека react-native-modal, яка підтримує анімації та дає змогу налаштовувати вміст вікон.

Серед інших UI-компонентів застосовувались:

- @react-native-community/slider – повзунок для візуального відображення прогресу читання книги.
- @react-native-picker/picker – для вибору статусу книги.
- react-native-safe-area-context та react-native-screens – для адаптації елементів до розміру екрана та врахування безпечних зон.

Для забезпечення плавних анімацій появи нотаток і загальної інтерактивності інтерфейсу використано такі підходи:

- react-native-reanimated – що дозволило зробити взаємодію сучасною та приємною для користувача.
- react-native-svg – необхідна залежність для рендерингу SVG-іконок.

Використання зазначених бібліотек і модулів сприяло створенню стабільного, адаптивного, візуально привабливого та зручного для користувача

мобільного додатка, уникаючи при цьому труднощів, пов'язаних із нативним налаштуванням. Такий підхід відповідає принципам проєктування архітектури програмного забезпечення, де вибір бібліотек має зберігати цілісність і якість системи [30]. .Водночас React Native має певні обмеження у функціональності, зокрема щодо підтримки вбудованих нативних можливостей, таких як забезпечення доступності, що вимагає додаткових рішень для розширення базового функціоналу. Це акцентує увагу на важливості ретельного вибору сторонніх бібліотек, які не лише забезпечують повноцінну інтеграцію з платформними рішеннями, але й зберігають переваги кросплатформеного підходу [29].

3.4 Підсумкова оцінка релізації до третього розділу

У процесі розробки мобільного застосунку BookLoop були успішно впроваджені ключові функції, визначені на етапі формулювання завдань (див. підрозділ 1.2). Цей застосунок пропонує користувачам зручний і функціональний інструмент для організації їхньої особистої бібліотеки, забезпечуючи всі необхідні засоби для ведення обліку прочитаних книг, тих, що плануються до читання, а також літератури в списку бажань. Окрім цього, користувачам надається можливість додавати нотатки та відслідковувати динаміку свого читацького прогресу.

До переліку реалізованих функціональних елементів входять:

- Додавання нової книги з повною інформацією: назва, автор, жанр, опис, кількість сторінок, обкладинка.
- Редагування книги, включаючи зміну статусу, оновлення тексту та кількості сторінок.
- Видалення книги зі збереженням цілісності списку.
- Ведення персональних нотаток: додавання, редагування, видалення.
- Категоризація книг за статусами: «Читаю», «На потім», «Закинув», «Прочитано».

- Автоматичний розрахунок прогресу читання на основі прочитаних сторінок.
- Візуальний індикатор прогресу (progress bar) у деталях книги.
- Відображення кількості прочитаних книг на головному екрані.
- Список бажаних книг (wishlist): окреме додавання, редагування, видалення.
- Фільтрація за статусами та пошук за назвою/автором.
- Позначення книг як улюблених з відповідним візуальним маркером.
- Анімована поява елементів інтерфейсу (через react-native-reanimated).
- Свайп-дії для нотаток (редагування/видалення) з використанням react-native-gesture-handler.
- Адаптивний дизайн зі світлою та темною темами.
- Повна офлайн-підтримка через локальне збереження даних в AsyncStorage.

Інтерфейс розроблений із врахуванням принципів зручності, доступності та логічної структури, що забезпечує позитивний досвід користувачів (UX) . Це дозволяє легко орієнтуватися в додатку, працювати з книгами, відстежувати прогрес і зберігати важливі нотатки.

Під час тестування (див. підрозділ 3.1) було підтверджено, що додаток стабільно працює як у режимі онлайн, так і офлайн. Усі дані надійно зберігаються навіть у разі примусового завершення роботи чи перезавантаження програми.

У подальших планах – розширення функціоналу, зокрема:

- Push-сповіщення для нагадувань.
- Хмарне резервне копіювання та синхронізацію між пристроями.
- Інтеграцію з базами ISBN для автоматичного заповнення даних про книги.
- Розширену статистику та аналітику читання.

Програма BookLoop успішно реалізує поставлені завдання та слугує робочим прототипом для особистого управління бібліотекою. Її розробка базувалася на принципах поступового впровадження [31]. Такий підхід також відповідає гнучкому проєктуванню [32].

3.5 Висновки до третього розділу

У третьому розділі кваліфікаційної роботи акцент зроблено на практичному впровадженні та функціонуванні мобільного додатка BookLoop. Детально розглянуто процес тестування основних функцій, організацію зберігання даних, а також інструменти й технології, застосовані в ході розробки. На основі отриманих даних сформульовано висновки щодо ефективності та доцільності втілених рішень.

Ручне тестування підтвердило стабільність роботи додатка: інформація про книги, записки та прогрес читання зберігалася коректно, а сам застосунок функціонував без збоїв навіть в офлайн-режимі. Для локального збереження даних використовувалася бібліотека @react-native-async-storage/async-storage, яка надає можливість доступу до інформації навіть за відсутності підключення до Інтернету, що є важливим фактором для багатьох користувачів.

Особливу увагу приділено аналізу технологій, що стали основою розробки, зокрема фреймворку React Native та пов’язаних із ним бібліотек. Розглянуто реалізацію навігації, анімацій, візуального контенту та побудову інтуїтивно зрозумілого й комфортного інтерфейсу. Завдяки інтеграції цих елементів було досягнуто високого рівня адаптивності додатка, його функціональної насиченості та приємності користувацького досвіду.

Зрештою, додаток цілком відповідає поставленим завданням: він пропонує зручний інструмент для управління особистою бібліотекою та створює базу для подальшого вдосконалення функціоналу та покращення інтерфейсу.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Працездатність людини – оператора

У сучасному цифровому світі поняття «оператор» набуло значно ширшого значення, ніж у минулому. Якщо раніше це стосувалося лише людей, які керують технікою або машинами у виробничих умовах, то сьогодні до цієї категорії відносять і тих, хто працює з інформаційними системами. Програмісти, аналітики, тестувальники, адміністратори – усі вони виконують функції операторів, адже їхня діяльність пов’язана з керуванням складними процесами, обробкою інформації та ухваленням рішень у взаємодії з технікою.

У такій роботі працездатність – це не просто фізична присутність або можливість натискати клавіші. Йдеться про здатність тривалий час залишатися уважним, мислити логічно, не втрачати фокус і приймати точні рішення. Зокрема, у сфері програмування одна помилка може призвести до збоїв у роботі цілої системи, що робить питання підтримання працездатності надзвичайно важливим.

Для цього важливо розуміти, які чинники впливають на ефективність праці. Частину з них можна побачити неозброєним оком – це організація робочого місця, освітлення, температура, наявність перерв. Проте інші, менш помітні, можуть мати ще більший вплив. Йдеться про психофізіологічні особливості людини: емоційний стан, рівень втоми, наявність стресу або тривожності [33].

Іноді працездатність знижується через постійні фактори – наприклад, хронічні захворювання або неврологічні порушення. Такі стани можуть непомітно, але стабільно знижувати здатність до концентрації, викликати швидке виснаження або дратівливість. У результаті – зниження точності, більше помилок, складнощі з логічним мисленням. У сфері ІТ це може мати серйозні наслідки, адже навіть незначна помилка у коді здатна вплинути на цілісність системи або безпеку даних.

Крім того, є чинники тимчасові – втома наприкінці робочого дня, нестача сну, емоційне виснаження. Вони не є постійною проблемою, але можуть суттєво знижувати працездатність у конкретний момент [33]. Працівник, який працює у стані емоційної напруги, часто діє повільніше, менш уважно і гірше справляється з навантаженням. Це також створює додаткові ризики – як для нього самого, так і для результату роботи.

Особливо чутливими до цих чинників є працівники, які лише починають професійну діяльність. Через брак досвіду вони витрачають більше ресурсів на виконання навіть простих завдань, швидше втомлюються, більше хвилюються. З досвідом приходить вміння раціонально розподіляти сили, реагувати спокійніше, тримати увагу на головному.

Разом із тим, не варто недооцінювати роль емоційного стану навіть у досвідчених працівників. Навіть фахівець високого рівня може втратити пильність після конфлікту, відчуття тривоги чи просто через накопичену втому. Усе це – природні реакції організму, які впливають на працездатність незалежно від компетенцій. І чим вища відповідальність завдань, тим більше значення мають такі, здавалося б, «непомітні» речі.

Зважаючи на це, підтримка працездатності має стати не лише особистою справою працівника, а й частиною корпоративної культури. Команда, яка розуміє цінність здорового психофізіологічного стану, створює сприятливе середовище для стабільної та якісної роботи [33]. Це передбачає не лише комфортні фізичні умови, а й можливість відпочинку, розвантаження, зміну діяльності, а іноді й просто спокійну розмову з колегою чи керівником.

Кожному з нас знайома втома, коли навіть просте завдання здається складним – саме тому важливо вміти вчасно зупинитися й подбати про себе.

Зрештою, працездатність – це не сталість, а баланс, який потребує постійної уваги. І коли цей баланс збережено, навіть напружена робота не буде виснажливою, а стане процесом, у якому є місце і для результату, і для людини.

4.2 Стан та значення охорони праці

У сучасному світі питання безпеки на робочому місці набувають особливого значення. Це реальний інструмент, що безпосередньо впливає на економічну стабільність країни, добробут громадян і загальний рівень розвитку суспільства. Суттєву роль у створенні безпечного виробничого середовища відіграє охорона праці, галузь, яка поєднує правові, соціально економічні, технічні, гігієнічні та профілактичні заходи, спрямовані на збереження життя, здоров’я та працездатності людини під час її професійної діяльності

Головна мета охорони праці – створити такі умови, за яких працівник може виконувати свої обов’язки без загрози для здоров’я. Йдеться не лише про загальні знання чи інструкції – охорона праці передбачає реальні дії, які допомагають запобігати нещасним випадкам, а також професійним хворобам, аваріям і пожежам.

Базові уявлення про цю сферу формуються ще на етапі навчання. Майбутні фахівці повинні добре розуміти принципи правового регулювання, техніки безпеки, санітарні норми та, що найважливіше, – усвідомлювати пріоритет збереження життя і здоров’я людини над будь-якими виробничими показниками

Як галузь знань, охорона праці зосереджується насамперед на умовах праці. У центрі її уваги перебуває виробнича система, яка включає три взаємопов’язані компоненти: людину, техніку та середовище, в якому вона працює. Методологічна основа цієї сфери базується на системному підході, що враховує вплив фізичних, хімічних, біологічних, психологічних та соціальних чинників на безпеку людини.

На жаль, високий рівень виробничого травматизму і професійних захворювань досі залишається серйозною проблемою. Це не лише питання здоров’я окремих працівників – йдеться про суттєві економічні втрати для держави. За підрахунками, щорічні витрати на компенсації постраждалим, а також на супутні виплати, перевищують один мільярд гривень

У цьому контексті важливу роль відіграє Закон України «Про загальнообов'язкове державне соціальне страхування від нещасного випадку», який забезпечує правові та фінансові гарантії для осіб, що втратили працездатність через виробничі травми або професійні хвороби

Ефективна система охорони праці ґрунтується на комплексному підході з урахуванням постійної взаємодії між людиною, технікою та навколишнім середовищем. Такий підхід дозволяє своєчасно виявляти потенційні загрози, оцінювати допустимі навантаження, визначати зони ризику та планувати профілактичні заходи, що мінімізують небезпеку

Сучасні умови праці змінюються швидко, і це породжує нові виклики. З'являються ризики, які не можна ігнорувати, вони вимагають оновлення засобів захисту, удосконалення механізмів контролю та адаптації до нових реалій. Охорона праці сьогодні – це не просто набір технічних норм, а складна система, що включає організаційні, психологічні, правові та інформаційні аспекти, спрямовані на збереження життя і здоров'я людини. Бо врешті-решт, ідеться не лише про правила, а про реальне життя людей, які щодня працюють у різних умовах.

Попри наявність законодавчої бази та державних програм, стан охорони праці в Україні потребує серйозного покращення. Основні проблеми залишаються незмінними: застаріле обладнання, нестача фінансування, низький рівень відповідальності роботодавців і, що не менш важливо, – недостатня культура безпеки. Формування свідомого ставлення до безпеки праці серед усіх учасників виробничого процесу є одним з першочергових завдань сьогодення.

Забезпечення належних умов праці – це не лише вимога закону, а й спільна відповідальність держави, роботодавців та самих працівників. Безпека життєдіяльності – це турбота не тільки про теперішнє, а й про майбутнє. Її рівень прямо впливає на здоров'я людей, ефективність підприємств і розвиток суспільства в цілому.

4.3 Висновок до четвертого розділу

У четвертому розділі акцент зроблено на тому, що часто залишається непоміченим – на людині, яка виконує розумову працю. Розкрито, як на працездатність впливають не лише навички чи досвід, а й втома, емоційний стан, рівень стресу та загальне навантаження. Описано як постійні, так і тимчасові чинники, що можуть знижувати ефективність, незалежно від рівня підготовки працівника. Особливу увагу приділено важливості підтримки внутрішнього балансу, своєчасного відпочинку та розумінню того, що працездатність – це не фіксована величина, а стан, який потребує уваги та підтримки. Підкреслюється, що турбота про психофізіологічний стан має бути не лише особистим завданням, а й частиною командної культури.

Охорона праці – це не просто про правила чи інструкції. Насамперед, це про людину, її безпеку, здоров’я і спокій на роботі. Коли створено належні умови, працюється не лише ефективніше, а й із більшим відчуттям стабільності та підтримки. Щоб це справді працювало, важливо, аби до питань безпеки відповідально ставилися всі – і держава, і роботодавці, і самі працівники. Лише разом можна створити середовище, де праця буде не загрозою, а простором розвитку.

ВИСНОВКИ

Перший розділ присвячений загальному огляду теми та аналізу її актуальності в умовах стрімкого розвитку цифрових технологій. Розглянуто наявні мобільні додатки для управління книжковими бібліотеками, зокрема їхні сильні та слабкі сторони. У цьому контексті було порушено питання вибору технології розробки: порівняно різні підходи та пояснено, чому саме React Native став основною платформою для реалізації проєкту.

У другому розділі мова йде вже про безпосередню реалізацію застосунку. Описано, як побудовано його структуру, як організовано зв’язки між різними частинами інтерфейсу та які завдання виконує кожен з екранів. Також показано, яким чином додаток реагує на дії користувача та які можливості він надає для взаємодії з персональною книжковою бібліотекою.

Третій розділ зосереджений на технічному боці роботи додатку. Тут докладно описано, як реалізовано збереження даних на пристрої та функціонування в умовах відсутності інтернету. Розкрито підхід до тестування: що саме перевірялося, які ситуації моделювалися та які труднощі виникали в процесі. Крім того, представлено огляд використаних бібліотек і технологій, що стали основою технічної реалізації, та зроблено висновки щодо якості й надійності отриманого результату.

Четвертий розділ присвячений працездатності людини в умовах інтелектуальної праці. Наголошено, що на ефективність впливають не лише професійні навички, а й загальний стан організму — рівень втоми, стресу, емоційного виснаження. Розглянуто чинники, що впливають на здатність концентруватися й приймати рішення. Підкреслено важливість відпочинку, психоемоційної рівноваги та підтримки з боку колективу. У цьому контексті охорона праці виступає як засіб створення безпечного й стабільного середовища, орієнтованого на добробут людини.

ПЕРЕЛІК ДЖЕРЕЛ

1. Digital 2025: Global Overview Report – DataReportal – Global Digital Insights. DataReportal – Global Digital Insights. URL: https://datareportal.com/reports/digital-2025-global-overview-report (дата звернення: 14.06.2025).

2. Okonkwo C. Assessment of User Experience (UX) Design Trends in Mobile Applications. Journal of Technology and Systems. 2024. Vol. 6, no. 5. P. 29–41. URL: https://doi.org/10.47941/jts.2147 (дата звернення: 14.06.2025).

3. Annual number of mobile app downloads worldwide 2023| Statista. *Statista*. URL: https://www.statista.com/statistics/271644/worldwide-free-and-paid (дата звернення: 14.06.2025).

4. Best Reading and Literature Apps to Discover and Enjoy Books in 2025. Software House. URL: https://softwarehouse.au/blog/best-reading-and-literature-apps-to-discover-and-enjoy-books-in-2025/ (date of access: 14.06.2025).

5. Majumder A. S. The Influence of UX Design on User Retention and Conversion Rates in Mobile Apps. arXiv. 2025. 2501.13407. URL: https://doi.org/10.48550/arXiv.2501.13407 (date of access: 14.06.2025).

6. A Systematic Literature Review of User Behavior and Personalization in Digital Libraries / M. Marzuki et al. International Journal of Research and Innovation in Social Science. 2025. Vol. IX, no. I. P. 4830–4842. URL: https://doi.org/10.47772/ijriss.2025.9010372 (date of access: 15.06.2025).

7. Lu G., Qu S., Chen Y. Understanding user experience for mobile applications: a systematic literature review. Discover Applied Sciences. 2025. Vol. 7, no. 6. URL: https://doi.org/10.1007/s42452-025-07170-3 (date of access: 15.06.2025)..

8. About Goodreads. Goodreads | Meet your next favorite book. URL: https://www.goodreads.com/about/us (date of access: 15.06.2025).

9. Evaluating the Usability of Goodreads – Quant UX • HSS_0EL42_TP. Quant UX • HSS_0EL42_TP – A course at Télécom Paris about the use of

quantitative methods for user-experience research. URL: https://quantux.telecom-paris.fr/2025/03/07/evaluating-the-usability-of-goodreads/ (date of access: 15.06.2025).

10. FAQs – Libib Help Center. Libib Help Center – You need help and we're here to give it!. URL: https://support.libib.com/faqs/ (date of access: 15.06.2025).

11. Goli V. R. react native evolution, native modules, and best practices. international journal of computer engineering and technology. 2021. Vol. 12, no. 2. P. 73–85. URL: https://doi.org/10.34218/ijcet_12_02_009 (date of access: 15.06.2025).

12. Zohud T., Zein S. Cross-Platform Mobile App Development in Industry: A Multiple Case-Study. International Journal of Computing. 2021. Vol. 20, no. 1. P. 46–54. URL: https://doi.org/10.47839/ijc.20.1.2091 (date of access: 15.06.2025).

13. Zarichuk O. Comparative analysis of frameworks for mobile application development: Native, hybrid, or cross-platform solutions. Вісник Черкаського державного технологічного університету. 2023. Vol. 28, no. 4. P. 19–27. URL: https://doi.org/10.62660/2306-4412.4.2023.19-27 (date of access: 15.06.2025).

14. Goli V. R. Cross-Platform Mobile Development: Comparing React Native and Flutter, and Accessibility in React Native. International Journal of Innovative Research in Computer and Communication Engineering. 2023. Vol. 11, no. 03. URL: https://doi.org/10.15680/ijircce.2023.1103002 (date of access: 12.06.2025).

15. React Native vs Native Development: Pros and Cons. AWS Consulting Partner | Gen AI | Product Engineering. URL: https://www.brilworks.com/blog/react-native-vs-native-development/ (date of access: 15.06.2025).

16. Безверхий О. І., Куценко О. І. Шляхи оптимізації кросплатформенних додатків із використанням бібліотеки react та фреймворку react native. Systems and Technologies. 2024. Т. 67, № 1. С. 30–35. URL: https://doi.org/10.32782/2521-6643-2024-1-67.5 (дата звернення: 12.06.2025).

17. Kharchenko, A., Halay, I., Zagorodna, N., & Bodnarchuk, I. (2015). Trade-off optimal decision of the problem of software system architecture choice. 2015 Xth International Scientific and Technical Conference» Computer Sciences and Information Technologies»(CSIT), 198–205.

18. Markowski M., Smołka J. A comparative analysis of the Flutter and React Native frameworks. Journal of Computer Sciences Institute. 2023. Vol. 29. P. 346–351. URL: https://doi.org/10.35784/jcsi.3794 (date of access: 15.06.2025).

19. Khan S. M., Nabi A. u., Bhanbhro T. H. Comparative Analysis between Flutter and React Native. International Journal of Artificial Intelligence & Mathematical Sciences. 2022. Vol. 1, no. 1. P. 16–29. URL: https://doi.org/10.58921/ijaims.v1i1.19.

20. Optimize React apps using a multi-layered structure - LogRocket Blog. LogRocket Blog. URL: https://blog.logrocket.com/optimize-react-apps-using-a-multi-layered-structure/ (date of access: 15.06.2025).

21. Prasad N. React Native folder structure. Medium. URL: https://medium.com/@nitishprasad/react-native-folder-structure-e9ceab3150f3 (date of access: 15.06.2025).

22. Project Structure. React Native Express. URL: https://www.reactnative.express/app/project_structure (date of access: 11.06.2025).

23. Bodnarchuk, I., Lisovyi, V., Kharchenko, O., & Galai, I. (2018). Adaptive method for assessment and selection of software architecture in flexible techniques of design. 2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), 1, 292–297.

24. Soral R. Five Android and iOS UI Design Guidelines for React Native. InfoQ. URL: https://www.infoq.com/articles/ios-android-react-native-design-patterns/ (date of access: 11.06.2025).

25. Getting started | React Navigation. React Navigation | React Navigation. URL: https://reactnavigation.org/docs/getting-started/ (date of access: 11.06.2025).

26. Uzayr S. b. Mastering React Native: A Beginner's Guide. Boca Raton : CRC Press, 2022. 354 p. URL: https://doi.org/10.1201/9781003310440.

27. Sandesara M., Bodkhe U., Tanwar S. Design and Experience of Mobile Applications: A Pilot Survey. Mathematics. 2022. Vol. 10, no. 14. 2380. URL: https://doi.org/10.3390/math10142380 (date of access: 12.06.2025).

28. React Native Gesture Handler: Create Swipeable Gestures. Jscrambler. URL: https://jscrambler.com/blog/creating-swipeable-gestures-with-react-native-gesture-handler (date of access: 15.06.2025).

29. AsyncStorage · React Native. React Native · Learn once, write anywhere. URL: https://reactnative.dev/docs/asyncstorage (date of access: 13.06.2025).

30. Волович, В., Береженко, Б. М., & Боднарчук, І. О. (2022). Задача проєктування програмної архітектури в процесах забезпечення якості. Матеріали X Науково-Технічної Конференції «Інформаційні Моделі, Системи Та Технології», 104–106.

31. Боднарчук, І., Харченко, О., Хоміцький, Б., & Шимчук, Г. (2019). Проектування архітектури програмних систем в проектах з гнучкими методами управління. Матеріали XXI Наукової Конференції Тернопільського Національного Технічного Університету Імені Івана Пулюя, 46–48.

32. Харченко, О., Яцишин, В., & Боднарчук, І. (2013). Експертна система проектування архітектури програмного забезпечення. Комп'ютерні Технології Друкарства, (29), 10–26.

33. Драган . І., Рудова А. Я., Бергер А. Д. Визначення стресових чинників у роботі працівників за допомогою нормування праці. Східна Європа: Економіка, бізнес та управління. 2022. № 1(34). С. 77–81. URL: https://doi.org/10.32782/easterneurope.34-13.

34. Automated algorithm for determining surface's oil capacity based on the analysis of the Abbot-Firestone diagram's parameters. Iaroslav Lytvynenko, Volodymyr Dzyura, Pavlo Maruschak CEUR Workshop Proceedings, 2024, 3896, pp. 74–79

35. Development of an algorithm for identification of damage types on the surface of sheet metal Palianytsia, Y., Lytvynenko, I., Menou, A., Shymchuk, G., Dubchak, A. CEUR Workshop Proceedings, 2024, 3742, pp. 84–96

36. Methodology of the Formation of Sports Matches Statistical Information Using Neural Networks Sorokivska, O., Lytvynenko, I., Sorokivskyi, O., Kozbur, H., Strutynska, I. CEUR Workshop Proceedings, 2023, 3628, pp. 389–403

37. The Method of Computer Modeling of Heart Rhythm based on the Vector of Stationary and Stationary-related Random Sequences Onyskiv, P., Lytvynenko, I., Oleksandr, V., Shymchuk, G., Hotovych, V. CEUR Workshop, 2023, 3468, pp. 223–232

38. Computer modeling of cardiac rhythm based on vector of stationary random sequences. Serhii Lupenko, Iaroslav Lytvynenko, Petro Onyskiv, Anatolii Lupenko, Oleksandr Volianyk, Olena Tsitsiura // Scientific Journal of TNTU. Tern.: TNTU, 2023. Vol 108. No 4. P. 131–143.

ДОДАТКИ

Додаток А

Код компонента SwipeableNote.js (свайпи та відображення нотаток)

```
import React from 'react';
import { View, Text, Image, TouchableOpacity } from 'react-
native';
import { Swipeable } from 'react-native-gesture-handler';
import { MaterialCommunityIcons, Feather } from '@expo/vector-
icons';
// Імпортуємо два стилі
import stylesDark from '../../styles/notes/noteCardStyles';
import stylesLight from '../../styles/notes/noteCardStylesLight';
export default function SwipeableNote({ note, onEdit, onDelete,
variant = 'dark' }) {
  const styles = variant === 'light' ? stylesLight : stylesDark;
  const renderRightActions = () => (
    <TouchableOpacity onPress={() => onEdit(note)}
style={styles.swipeEdit}>
      <Feather name=«edit-2» size={20} color=«#fff» />
    </TouchableOpacity>
  );
  const renderLeftActions = () => (
    <TouchableOpacity onPress={() => onDelete(note)}
style={styles.swipeDelete}>
      <MaterialCommunityIcons name=«trash-can-outline» size={20}
color=«#fff» />
    </TouchableOpacity>
  );
  return (
    <Swipeable
      renderLeftActions={renderLeftActions}
      renderRightActions={renderRightActions}
      overshootLeft={false}
      overshootRight={false}
    >
      <View style={styles.noteCard}>
        <View style={styles.noteHeader}>
          {note.bookImage && (
            <Image source={{ uri: note.bookImage }}
style={styles.bookImage} />
          )}
          <MaterialCommunityIcons
            name=«book-open-page-variant»
            size={14}
            color={variant === 'light' ? '#000' : '#fff'}
            style={{ marginRight: 4 }}
          />
          <Text style={styles.bookTitle}>{note.bookTitle}</Text>
        </View>

        <Text style={styles.noteText}>{note.text}</Text>
```

```
        <Text style={styles.noteDate}>
          {note.date ? new Date(note.date).toLocaleString('uk-UA')
: ''}
        </Text>
      </View>
    </Swipeable>
  );
}
```

Додаток Б

BookDetailsScreen.js (екран деталі книги)

```
import React from 'react';
import { View, ScrollView, Text, SafeAreaView } from 'react-
native';
import styles from '../styles/screens/bookDetailsStyles';
import { useNavigation, useRoute } from '@react-
navigation/native';
import { useBooksContext } from
'../context/books/useBooksContext';

import BookHeader from '../components/book/BookHeader';
import BookProgress from '../components/book/BookProgress';
import BookNotes from '../components/notes/BookNotes';
import BookFooter from '../components/book/BookFooter';
import RatingBlock from '../components/book/RatingBlock';
import DeleteBookButton from
'../components/book/DeleteBookButton';

export default function BookDetailsScreen() {
  const navigation = useNavigation();
  const route = useRoute();
  const { bookId } = route.params;

  const {
    books,
    wishlist,
    updateBookStatus,
    updateCurrentPage,
    toggleFavorite,
    editBook
  } = useBooksContext();

  const book = books.find(b => b.id === bookId) || wishlist.find(b
=> b.id === bookId);
  const isWishlistBook = wishlist.some(b => b.id === bookId);

  if (!book) {
    return (
      <View style={styles.container}>
        <Text>Книгу не знайдено.</Text>
      </View>
    );
  }

  const status = book.status;
  const totalPages = book.totalPages || 0;
  const currentPage = book.currentPage || 0;
  const isFavorite = book.isFavorite || false;
```

```
  const progress = totalPages > 0 ? Math.round((currentPage /
totalPages) * 100) : 0;
  const notes = book.notes || [];
  const rating = book.rating || 0;

  return (
    <SafeAreaView style={{ flex: 1, backgroundColor: '#EFE8E1' }}>
      <ScrollView
        contentContainerStyle={{ paddingBottom: 40 }}
        style={{ backgroundColor: '#EFE8E1' }}
      >
        <View style={styles.container}>
          {/* Заголовок книги */}
          <BookHeader
            image={book.image}
            title={book.title}
            author={book.author}
            genre={book.genre}
            publisher={book.publisher}
            description={book.description}
            status={status}
            isFavorite={isFavorite}
            onToggleFavorite={() => toggleFavorite(bookId)}
            onChangeStatus={
              isWishlistBook ? undefined : (val) =>
updateBookStatus(bookId, val)
            }
          />

          {/* Прогрес, рейтинг, нотатки */}
          {!isWishlistBook && (
            <View style={styles.cardBlock}>
              <BookProgress
                currentPage={currentPage}
                totalPages={totalPages}
                progress={progress}
                onUpdatePage={(val) => updateCurrentPage(bookId,
val)}
              />

              {status === 'Прочитано' && (
                <RatingBlock
                  rating={rating}
                  onChange={(newRating) => editBook(bookId, {
rating: newRating })}
                />
              )}

              <BookNotes
                notes={notes}
                bookId={bookId}
                bookTitle={book.title}
                bookImage={book.image}
              />
```

```
            </View>
          )}

          {/* Додаткова інформація */}
          <BookFooter
            id={bookId}
            title={book.title}
            author={book.author}
            image={book.image}
            description={book.description}
            genre={book.genre}
            status={status}
            progress={progress}
            publisher={book.publisher}
            publishedDate={book.publishedDate}
            language={book.language}
            totalPages={totalPages}
            isbn13={book.isbn13}
            onMarkAsRead={
              isWishlistBook
                ? undefined
                : () => {
                    updateBookStatus(bookId, 'Прочитано');
                    updateCurrentPage(bookId, totalPages);
                    navigation.goBack();
                  }
            }
          />

          {/* Кнопка видалення */}
          <DeleteBookButton
            bookId={bookId}
            bookTitle={book.title}
            isWishlistBook={isWishlistBook}
          />
        </View>
      </ScrollView>
    </SafeAreaView>
  );
}
```

Додаток В

BooksProvider.js (логіка збереження та обробки даних)

```
import React, { createContext, useState, useEffect, useCallback,
useMemo } from 'react';
import AsyncStorage from '@react-native-async-storage/async-
storage';
import { v4 as uuidv4 } from 'uuid';
import { pickImage } from './booksHelpers';

export const BooksContext = createContext();

export const BooksProvider = ({ children }) => {
  const [books, setBooks] = useState([]);
  const [wishlist, setWishlist] = useState([]);
  const [search, setSearch] = useState('');

  useEffect(() => {
    const loadBooks = async () => {
      const saved = await AsyncStorage.getItem('books');
      if (saved) {
        const parsed = JSON.parse(saved);
        const updated = parsed.map(book => ({
          ...book,
          status:
            book.status === 'reading' ? 'Читаю' :
            book.status === 'later' ? 'На потім' :
            book.status === 'finished' ? 'Прочитано' :
            book.status === 'abandoned' ? 'Закинув' :
            book.status === 'не вказано' ? 'На потім' :
            book.status,
          isFavorite: book.isFavorite || false,
          notes: book.notes || []
        }));
        setBooks(updated);
      }
    };

    const loadWishlist = async () => {
      const saved = await AsyncStorage.getItem('wishlist');
      if (saved) {
        const parsed = JSON.parse(saved);
        setWishlist(parsed);
      }
    };

    loadBooks();
    loadWishlist();
  }, []);

  useEffect(() => {
```

```
    AsyncStorage.setItem('books', JSON.stringify(books));
  }, [books]);

  useEffect(() => {
    AsyncStorage.setItem('wishlist', JSON.stringify(wishlist));
  }, [wishlist]);

  const addBook = useCallback((newBook) => {
    setBooks(prev => [...prev, { id: uuidv4(), ...newBook }]);
  }, []);

  const addToWishlist = useCallback((newBook) => {
    const bookToAdd = {
      id: uuidv4(),
      ...newBook,
      status: 'wishlist',
      isFavorite: false,
      notes: []
    };
    setWishlist(prev => [...prev, bookToAdd]);
  }, []);

  const deleteFromWishlist = useCallback((id) => {
    setWishlist(prev => prev.filter(book => book.id !== id));
  }, []);

  const editWishlistBook = useCallback((id, updated) => {
    setWishlist(prev =>
      prev.map(book => (book.id === id ? { ...book, ...updated } :
book))
    );
  }, []);

  const editBook = useCallback((id, updated) => {
    setBooks(prev =>
      prev.map(book => (book.id === id ? { ...book, ...updated } :
book))
    );
  }, []);
  const deleteBook = useCallback((id) => {
    setBooks(prev => prev.filter(book => book.id !== id));
  }, []);

  const updateBookStatus = useCallback((id, status) => {
    setBooks(prev =>
      prev.map(book =>
        book.id === id ? { ...book, status } : book
      )
    );
  }, []);
  const updateCurrentPage = useCallback((id, page) => {
    setBooks(prev =>
```

```
      prev.map(book => {
        if (book.id === id) {
          const newProgress =
            book.totalPages && book.totalPages > 0
              ? Math.round((page / book.totalPages) * 100)
              : 0;
          return {
            ...book,
            currentPage: page,
            progress: newProgress
          };
        }
        return book;
      })
    );
  }, []);

  const toggleFavorite = useCallback((id) => {
    setBooks(prev =>
      prev.map(book =>
        book.id === id ? { ...book, isFavorite: !book.isFavorite }
: book
      )
    );
  }, []);

  const addNoteToBook = useCallback((bookId, newNote) => {
    setBooks(prevBooks =>
      prevBooks.map(book =>
        book.id === bookId
          ? {
              ...book,
              notes: [...(book.notes || []), newNote]
            }
          : book
      )
    );
  }, []);

  const filteredBooks = useMemo(() => {
    return books.filter(book =>
      (book.title?.toLowerCase() ||
'').includes(search.toLowerCase()) ||
      (book.author?.toLowerCase() ||
'').includes(search.toLowerCase())
    );
  }, [books, search]);

  return (
    <BooksContext.Provider
      value={{
        books:filteredBooks,
```

```
        wishlist,
        setSearch,
        search,
        addBook,
        addToWishlist,
        deleteFromWishlist,
        editBook,
        editWishlistBook,
        deleteBook,
        updateBookStatus,
        updateCurrentPage,
        toggleFavorite,
        addNoteToBook,
        pickImage
      }}
    >
      {children}
    </BooksContext.Provider>
  );
};
```