ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МІСЬКОГО ГОСПОДАРСТВА ІМЕНІ О. М. БЕКЕТОВА
Навчально-науковий інститут енергетичної, інформаційної
та транспортної інфраструктури

Кафедра комп'ютерних наук та інформаційних технологій

Пояснювальна записка
до кваліфікаційної роботи бакалавра

на тему: «Проєктування мобільного застосунку для персоналізованого читання книг "BookHive"»

Виконала: здобувачка вищої освіти
4 курсу, групи КН 2021-1
спеціальності
122 Комп'ютерні науки
(шифр і назва спеціальності)

Валерія ПОЛТОРАК
(ім'я та прізвище)

Керівник: Борис БОЧАРОВ
(ім'я та прізвище)

Рецензент Наталія БРАТЕРСЬКА
(ім'я та прізвище)

м. Харків – 2025 рік

Харківський національний університет міського господарства імені О. М. Бекетова
(повне найменування закладу вищої освіти)
Навчально-науковий Інститут енергетичної, інформаційної та транспортної інфраструктури
Кафедра комп'ютерних наук та інформаційних технологій
Рівень вищої освіти перший (бакалаврський)
Спеціальність 122 Комп'ютерні науки
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри КНтаІТ

Марина НОВОЖИЛОВА
«_23__» ____06_____2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Полторак Валерії Віталіївни

(прізвище, ім'я, по батькові)

1. Тема роботи «Проєктування мобільного застосунку для персоналізованого читання книг "BookHive"»

керівник роботи Бочаров Борис Петрович, к. т. н., доцент, доцент кафедри комп'ютерних наук та інформаційних технологій
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «09» травня 2025 р. № 341-03

2. Термін подання студентом роботи 23.06.2025

3. Вихідні дані до роботи: Аналіз потреб користувачів у сфері цифрового читання, рекомендації щодо функціональності мобільних застосунків для читання, індивідуальне завдання на розробку мобільного застосунку BookHive.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): 1. Аналітична частина (Загальні положення) 2. Інформаційно-математична частина 3. Програмно-технічна частина 4. Охорона праці

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):

Презентація – 21 слайд.

6. Консультанти розділів роботи

Розділ	Ім'я та Прізвище, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Борис БОЧАРОВ, к.т.н., доцент кафедри комп'ютерних наук та інформаційних технологій	30.04.2025	04.05.2025
2	Борис БОЧАРОВ., к.т.н., доцент кафедри комп'ютерних наук та інформаційних технологій	05.05.2025	11.05.2025
3	Борис БОЧАРОВ, к.т.н., доцент кафедри комп'ютерних наук та інформаційних технологій	12.05.2025	20.05.2025
4	Вікторія МАЛИШЕВА, к. т. н., доцент кафедри безпеки життєдіяльності	21.05.2025	06.06.2025

7. Дата видачі завдання 12.05.2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Вибір теми дипломної роботи	26.04.2025	Виконано
2	Затвердження тем, наукових керівників, завдань та календарного плану до дипломної роботи	28.04.2025	Виконано
3	Написання I розділу	04.05.2025	Виконано
4	Написання II розділу	11.05.2025	Виконано
5	Написання III розділу	20.05.2025	Виконано
6	Написання IV розділу	06.06.2025	Виконано
7	Подання дипломної роботи керівнику	08.06.2025	Виконано
8	Робота по усуненню зауважень керівника, уточнення і доповнення практичного матеріалу, оформлення додатків до роботи	14.06.2025	Виконано
9	Подання доопрацьованого варіанту роботи керівнику	16.06.2025	Виконано
10	Подання роботи на кафедру, передзахист кваліфікаційної роботи	21.06.2025	Виконано
11	Офіційний захист матеріалів дипломної роботи на засіданні екзаменаційної комісії	24.06.2025	Виконано

Студент _____ (підпис) Валерія ПОЛТОРАК (прізвище та ініціали)

Керівник роботи _____ (підпис) Борис БОЧАРОВ (прізвище та ініціали)

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка кваліфікаційної роботи бакалавра студентки групи КН 2021-1 спеціальності 122 «Комп'ютерні науки» Полторак Валерії Віталіївни на тему «Проєктування мобільного застосунку для персоналізованого читання книг BookHive» складається з 4 розділів, містить 52 рисунки, 4 таблиці, 22 бібліографічних джерела.

Кваліфікаційну роботу присвячено розробці мобільного застосунку для персоналізованого читання електронних книг з розширеним функціоналом аналітики, рекомендацій і соціальної взаємодії. Застосунок підтримує популярні формати файлів (.epub, .fb2, .txt, .pdf), дозволяє відслідковувати прогрес читання, брати участь у читацьких викликах, зберігати нотатки й закладки, а також спілкуватися з іншими користувачами.

У розділі «Загальні положення» виконано аналіз предметної області, вивчено існуючі аналоги, сформульовано мету, завдання, об'єкт і предмет дослідження.

У розділі «Інформаційне та математичне забезпечення» описано методи, підходи та технології, що використовувалися при розробці, наведено приклади рекомендаційних алгоритмів та структури читання.

У розділі «Програмне та технічне забезпечення» подано опис архітектури системи, структури застосунку, функціональних модулів, наведено фрагменти коду й керівництво користувача. Реалізація виконана на основі React Native + Expo, Firebase, Expo Router.

У розділі з охорони праці розглянуто аспекти безпечної роботи з мобільними пристроями, організацію середовища розробки та психологічні навантаження на користувача.

Ключові слова: МОБІЛЬНИЙ ЗАСТОСУНОК, ЕЛЕКТРОННІ КНИГИ, ІНФОРМАЦІЙНА СИСТЕМА, ПРОГРЕС ЧИТАННЯ, FIREBASE, РЕКОМЕНДАЦІЇ, ЧИТАЦЬКІ ВИКЛИКИ, UX/UI.

ANNOTATION

The explanatory report of the bachelor's qualification work by student Poltorak Valeriia Vitaliivna, group KN 2021-1, specialty 122 "Computer Science", on the topic " Creating a mobile application for personalized reading of books “BookHive”", consists of 4 chapters, contains 52 figures, 4 tables, and 19 bibliographic sources.

This qualification work is dedicated to the development of a mobile application for personalized electronic book reading with extended functionality for analytics, recommendations, and social interaction. The application supports popular file formats (.epub, .fb2, .txt, .pdf), allows users to track reading progress, participate in reading challenges, save notes and bookmarks, and communicate with other users.

The chapter "General Provisions" provides an analysis of the subject area, a review of existing analogues, and formulates the purpose, objectives, object, and subject of the research.

The chapter "Information and Mathematical Support" describes the methods, approaches, and technologies used in the development, including examples of recommendation algorithms and reading structure.

The chapter "Software and Technical Support" contains a description of the system architecture, application structure, functional modules, code snippets, and user guide. The implementation is based on React Native + Expo, Firebase, and Expo Router.

The chapter on occupational safety examines aspects of safe interaction with mobile devices, organization of the development environment, and psychological workload on the user.

Keywords: MOBILE APPLICATION, E-BOOKS, INFORMATION SYSTEM, READING PROGRESS, FIREBASE, RECOMMENDATIONS, READING CHALLENGES, UX/UI.

ЗМІСТ

ВСТУП

На сьогодні, в умовах стрімкого розвитку інформаційних технологій, цифрове читання набуває все більшої популярності. Люди дедалі частіше віддають перевагу електронним книгам, які доступні з будь-якого пристрою, не займають фізичного простору та дозволяють читати у зручний час. У зв’язку з цим зростає попит на зручні та функціональні мобільні застосунки для читання. Однак аналіз існуючих рішень на ринку показує, що більшість таких застосунків мають обмежений функціонал. Унаслідок цього виникає потреба у створенні комплексного рішення, яке поєднує ключові можливості сучасних рідерів і додаткові функції, що сприяють залученню до регулярного читання.

Розробка мобільного застосунку BookHive є актуальною у зв’язку з необхідністю створення універсального середовища для читання електронних книг, яке включатиме не лише базові функції, а й інтелектуальні можливості – рекомендації, читацьку статистику, соціальну взаємодію та мотиваційні механіки. Такий застосунок дозволить зробити процес читання не лише зручним, а й захоплюючим, сприяючи формуванню стабільної читацької звички. Метою цієї дипломної роботи є розробка мобільного застосунку, що забезпечує повноцінний процес персоналізованого читання.

Завдання дослідження:

- провести аналіз предметної області та визначити недоліки існуючих рішень;
- розробити логічну та функціональну архітектуру мобільного застосунку;
- обрати інструментарій та середовище розробки;
- реалізувати основні модулі, екрани та компоненти застосунку;
- забезпечити збереження даних за допомогою хмарної бази;

- провести тестування, моделювання типових сценаріїв взаємодій: обґрунтувати функціональність застосунку та оцінити результат.

Теоретична цінність полягає у використанні сучасних принципів архітектури мобільних застосунків, структуризації компонентів та інтеграції сторонніх сервісів. Прикладна значущість проєкту полягає у тому, що створений застосунок може бути використаний у повсякденному житті для читання та організації читацької активності, а також легко масштабуватися з додаванням нових функцій.

РОЗДІЛ 1
ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

1.1.1 Опис процесу діяльності

У світі, що базується на знаннях, значення читання як психічної та навчальної діяльності залишається вирішальним. Тим не менш, звичайні методи читання розвиваються через цифрові досягнення. Сьогодні багато користувачів читають книги з мобільних пристроїв, електронних книг та планшеті. Цей тренд особливо помітний серед молоді, студентів, фахівців у галузі ІТ та гуманітарних наук, які прагнуть оптимізувати власний час та зберігати мобільність. У той же час, зі зростанням кількості доступних книг та платформ, існує потреба в інструментах для організації процесу читання, підтримки прогресу, планування бажаних списків та отримання персоналізованих рекомендацій.

Традиційних процес читання передбачає послідовне ознайомлення з текстом книги, часто без можливості залишати інтерактивні нотатки або позначки. Коли читач має багату кількість прочитаних або запланованих книг, то він змушений вести записи у сторінках додатка (нотатки, календарі і т.ін.) або окремо у кожній книзі робить помітки – це ускладнює управління власною бібліотекою. Крім того, при читанні електронних книг виникають складнощі з перенесенням прогресу на інші пристрої або відновленням даних після втрати доступу до облікового запису.

Згідно з дослідженням [1], сучасні користувачі очікують від програмного забезпечення не лише зручного відображення тексту, а й розширеного функціоналу: можливість відстеження прогресу, інтегрований словник, генерація рекомендацій на основі уподобань та функції соціальної взаємодії. На практиці ж більшість платформ зосереджена лише на продажу

книг або базових функціях «читалки», ігноруючи зростаючий запит на інтелектуальні книжкові сервіси [2].

Процес організації читання в умовах сучасної діджиталізації включає наступні етапи:

– вибір книги для читання (зазвичай із особистих записів або рекомендації однодумців та друзів);

– початок читання книги (без автоматичного збереження прогресу в більшості випадків);

– переривання читання та необхідність вручну пам’ятати або фіксувати місце зупинки;

– бажання знайти схожі книги або продовження – що потребує додаткового пошуку, що може займати багато часу.

Отже, очевидною є потреба у створенні мобільного застосунку, який дозволить централізовано керувати всіма аспектами процесу читання – від вибору книг до оцінювання прочитаного та обміну враженнями з іншими користувачами.

Крім того, важливо забезпечити гнучкість взаємодії із зовнішніми сервісами, що надають доступ до відкритих електронних бібліотек, а також впровадити інструменти аналізу читацької активності для формування персоналізованих порад, що базуються не лише на жанрових уподобаннях, а й на поведінкових даних користувача.

1.1.2 Опис функціонування моделі

Функціональна модель застосунку «BookHive» базується на принципі персоналізованого підходу до цифрового читання. Застосунок об’єднує у собі функції класичної читалки, органайзера читання, соціальної платформи та системи рекомендацій.

Основні функції системи можна згрупувати за п’ятьма напрямками (рис. 1.1): функції читання, обліку, нагадування та мотивації, соціальної взаємодії й інтелектуальної персоналізації.

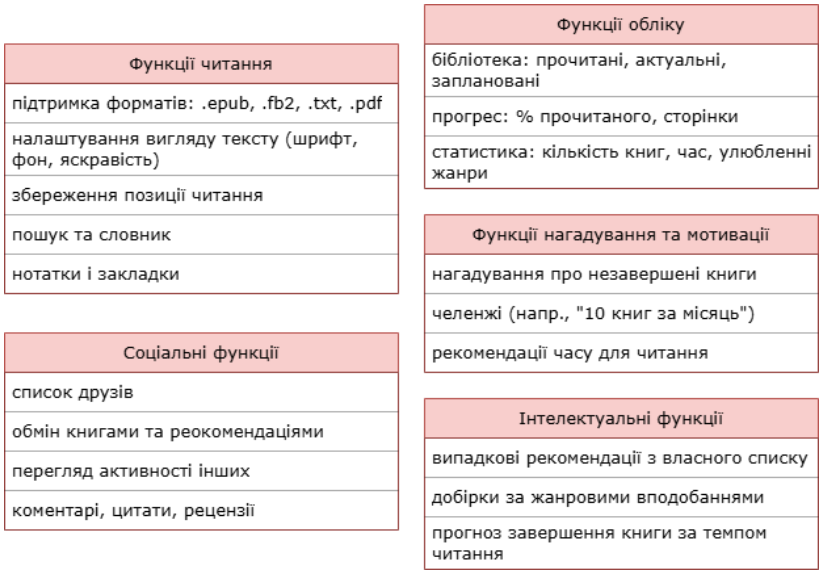

Рисунок 1.1 – Основні функції системи

Перш за все, функції читання забезпечують комфортну взаємодію користувача з електронним текстом. Система повинна підтримувати популярні формати книг, такі як: .epub, .fb2, .txt, .pdf. Також дозволяти користувачеві налаштовувати зовнішній вигляд сторінки – змінювати шрифти, кольори фону та яскравість. Потрібні функції збереження позиції, пошуку по тексту, інтеграції з вбудованим словником, а також можливість додавати закладки і нотатки.

Функції обліку включають ведення персональної бібліотеки, розділеної на прочитані, актуальні та заплановані книги. Система автоматично відстежує прогрес читання – кількість прочитаних сторінок, відсоток завершення тощо. Користувач може переглядати загальну статистику: кількість прочитаних книг, витрачений час, улюблені жанри.

До функцій нагадування та мотивації входять механізми, які підтримують регулярне читання, мотивують. Застосунок сповіщає про незавершені книги, пропонує тематичні челенджі та рекомендації щодо часу для читання – на основі користувацької активності.

Соціальні функції дозволяють додавати друзів, обмінюватися книгами та рекомендаціями, переглядати читацьку активність інших користувачів.

Реалізовано можливість залишати коментарі, ділитися улюбленими цитатами та публікувати міні-рецензії.

Нарешті, інтелектуальні функції використовують алгоритми для персоналізації контенту. Система генерує рекомендації на основі жанрових вподобань, оцінок і читацької історії. Крім того, вона здатна прогнозувати орієнтовний час завершення книги відповідно до темпу читання конкретного користувача [3].

Усі ці функції формують взаємопов’язану систему, яка не лише підтримує процес читання, але й стимулює мотивацію, соціальну взаємодію та збагачення читацького досвіду.

Функціональну модель можна представити як множину взаємопов’язаних сервісів, які взаємодіють із користувачем через зручний мобільний інтерфейс. Вона забезпечує зворотній зв’язок, накопичення даних та їхнє подальше використання для персоналізації сервісу.

Ця модель повинна враховувати специфіку мобільного читання: короткі сесії, потребу в збереженні контексту, адаптивний інтерфейс залежно від типу пристрою. Додатково до основного функціоналу застосунок може містити модуль аналітики, який дозволить користувачеві переглядати статистику, формувати персональні добірки літератури та автоматично отримувати поради на основі його читацького стилю.

Варто зазначити, що для реалізації такої моделі важливою є підтримка хмарної синхронізації з метою збереження прогресу даних про книги, а також локальне зберігання (для офлайн-режиму). З точки зору архітектури, найбільш підходящім є модульних підхід використанням шаблону MVVM або Clean Architecture для поділу логіки інтерфейсу, бізнес-логіки та роботи з даними.

1.2 Огляд наявних аналогів

У цифровому середовищі існує велика кількість мобільних застосунків, що надають користувачам можливості читати електронні книги, вести облік

прочитаного, створювати списки бажаного та взаємодіяти з іншими читачами. Проте ці функції, як правило, реалізовані розрізнено: одні застосунки орієнтовані виключно на читання, інші – на соціальну взаємодію або рекомендаційні механізми. Деякі з них використовують лише базові алгоритми, не враховуючи індивідуальні читацькі звички користувача. Розглянемо деякі з найпопулярніших сервісів та платформ, які мають схожий функціонал. Мета цього огляду – визначити переваги та недоліки існуючих рішень, а також з’ясувати, на які функції варто зробити акцент під час проєктування власної системи (табл. 1.1).

Таблиця 1.1 – Порівняльна характеристика існуючих застосунків для читання книг

Назва застосунку	Основні функції	Переваги	Недоліки
1	2	3	4
Goodreads	Каталогізація книг, створення книжкових полиць (прочитане, бажане), оцінка та рецензії, обговорення в групах, соціальні функції (друзі, стрічка активності)	Велика база даних (понад 2 млрд книжок), активне книжкове ком'юніті, інтеграція з Amazon Kindle, можливість створення власних списків	Відсутність вбудованої читалки, низький рівень персоналізації рекомендацій, застарілий інтерфейс
Rork	Мінімалістична платформа з інтелектуальними підказками, створення списків, рекомендації за стилем читання, базові аналітичні звіти	Пріоритет на персоналізацію, інноваційний підхід до інтерфейсу, простота використання	Обмежена підтримка форматів, відсутність читалки, відсутність синхронізації та розширених функцій обліку
Libby	Читання та прослуховування електронних та аудіокниг через бібліотеки, синхронізація прогресу, таймер для аудіо	Безкоштовний доступ до великої кількості книг через бібліотечні картки, підтримка Kindle	Потрібна бібліотечна картка, обмежений доступ до книг залежно від бібліотеки
Moon+ Reader	Професійна читалка з розширеними налаштуваннями, підтримка великої кількості форматів, ведення статистики	Підтримка TTS (читання вголос), дуже гнучкі налаштування, режим нічного читання	Висока складність інтерфейсу для новачків, платна Pro-версія для повного функціоналу

Продовження таблиці 1.1

1	2	3	4
Librera	Читання файлів у форматах PDF, EPUB, MOBI, FB2, підтримка озвучування (TTS), авто-скролінг, нічний режим	Легкий, швидкий, підтримує велику кількість форматів, TTS, режим для водіїв	Менш інтуїтивний інтерфейс, відсутні соціальні функції, обмежена бібліотечна організація
FullReader	Читалка з аудіоплеєром, підтримка PDF, EPUB, MOBI, TTS, хмарне збереження, розділення за категоріями	Вбудований аудіоплеєр, Google Drive синхронізація, стильний інтерфейс	Присутня реклама в безкоштовній версії, платні функції у Premium
Lithium	Автоматичне виявлення EPUB-книг, простий інтерфейс, підтримка нічного режиму, збереження прогресу	Дуже легкий та інтуїтивний, без реклами, автоматична бібліотека	Підтримує лише формат EPUB, мінімум функцій, немає хмарної синхронізації
PocketBook Reader	Читання PDF, EPUB, DJVU, FB2 та ін., хмарна синхронізація, озвучування, вбудовані словники	Висока підтримка форматів, багатомовні словники, зручний перегляд PDF	Іноді повільно працює з великими файлами, обмежена інтерактивність інтерфейсу

Тепер зосередимо увагу на трьох застосунках з таблиці 1.1: Rork, Moon+ Reader та Goodreads. Вони представляють різні підходи до реалізації систем. Кожен з них демонструє унікальний концептуальний напрям, проте не охоплює повного функціонального спектра.

Rork – інноваційний мобільний застосунок, розроблений в Україні як стартап, орієнтований на персоналізований підхід до рекомендацій книг. Основна концепція полягає в мінімалістичному інтерфейсі та фокусі на читацькому досвіді, що базується на аналізі вподобань користувача. Застосунок дозволяє створювати списки бажаного, отримувати підказки, а також переглядати базову аналітику щодо стилю читання [4].

Попри інноваційний підхід до персоналізації, Rork не має вбудованої читалки, що змушує користувача звертатися до сторонніх засобів для безпосереднього читання. Крім того, застосунок не підтримує імпорт файлів, не має синхронізації з хмарними сховищами та не реалізує механізми обліку

прогресу. Отже, його потенціал зосереджено на рекомендаційній системі, але при цьому відсутні базові функції організації процесу читання як такого.

Moon+ Reader – це професійна читалка, розроблена незалежними розробниками з Китаю, яка вже понад десять років підтримується та активно використовується по всьому світу. Застосунок має надзвичайно гнучке налаштування інтерфейсу, підтримує понад 10 популярних форматів електронних книг, включно з .epub, .pdf, .fb2, .djvu, .mobi, .chm тощо. У програмі реалізовано функції нічного режиму, озвучування (TTS), статистики читання, а також потужну систему закладок і анотацій [5].

Незважаючи на потужний функціонал читалки, Moon+ Reader практично не містить соціальних елементів, не підтримує рекомендацій або обміну книгами. Тим паче, інтерфейс програми досить перевантажений і може здатися складним для новачків. Частина функцій доступна лише у платній Pro-версії, що також обмежує її повну доступність.

Goodreads – це найвідоміший у світі соціальний книжковий сервіс, розроблений у США та придбаний компанією Amazon у 2013 році. Застосунок дозволяє створювати власні книжкові полиці, залишати оцінки, рецензії, вести читацькі челенджі, комунікувати з друзями та ділитися враженнями від прочитаного. Його база даних містить понад 2 мільярди книжкових записів, і він активно інтегрований з Amazon Kindle, що забезпечує широку популярність серед англомовних користувачів [6]. Однак Goodreads не має власного модуля для читання книг – він більше схожий на книжкову соцмережу, ніж на застосунок для безпосереднього читання. Крім того, функції персоналізації рекомендацій є досить обмеженими, а інтерфейс виглядає застарілим і неадаптованим до сучасних стандартів UX/UI.

Аналіз існуючих платформ дозволяє зробити висновок, що жоден із розглянутих сервісів не поєднує всі функціональні можливості, які закладаються в основу розробки застосунку BookHive. Переважна більшість сервісів або зосереджена на читанні без глибокої інтеграції інтелектуальних функцій, або навпаки – пропонує розвинені соціальні механізми, проте не

підтримує роботу з різними книжковими форматами. Окремі застосунки мають високий рівень персоналізації, як-от Rork, однак поки що не демонструють повноцінної системи обліку та аналітики. Таким чином, при створенні BookHive варто акцентувати увагу на комплексному підході: поєднанні зручного та функціонального інтерфейсу читалки з персоналізованими рекомендаціями, гнучкою системою обліку, мотиваційними елементами та соціальною взаємодією користувачів.

1.3 Постановка задачі

Як вже було сказано раніше, що у сучасних умовах швидкого розвитку інформаційних технологій спостерігається активне зростання інтересу до саме цифрового читання. Це можна пояснити не лише тільки зручністю доступу до різного контенту, а й можливістю читати з будь-якого пристороюю у зручний час. Однак, незважаючи на наявність великої кількості мобільних застосунків для читання, вони задовольняють лише окремі функціональні потреби. Наприклад, як було згадано раніше, деякі з них пропонують лише зручний інтерфейс, інші – на читання файлів або на відслідковування прогресу. У зв’язку з цим виникає потреба у створенні комплексного рішення, яке буде об’єднувати найкращі функціональні можливості в одному середовищі.

Мета розробки мобільного застосунку BookHive полягає у створенні інтегрованої платформи для персоналізованого читання електронних книг, яка дозволить користувачам повноцінно взаємодіяти з усіма етапами читацького процесу – від вибору книг до обговорення. Також застосунок має забезпечувати читання наступних форматів: .epub, .fb2, .txt, .pdf. Додатково необхідна функція збереження місця зупинку, зміна оформлення інтерфейсу, додавання закладок та нотаток. Особливий акцент треба приділити також на інтелектуальні функції: рекомендації на основі вподобань, активності та жанрових переваг, ще створення викликів, які будуть мотивувати користувачів

до регулярного читання. Крім цього, система повинна вміти нагадувати про незавершене читання чи заплановані книги.

Однією з ключових переваг застосунку має стати можливість соціальної взаємодії: додавання друзів, обмін книжками, обговорення, цитування та перегляд активності однодумців. Важливо, щоб взаємодія із застосунком була не лише функціональною, а й естетично привабливою – відповідно до сучасних принципів UX/UI дизайну.

З технічної точки зору, система має бути побудована, наприклад, на базі фреймворку React Native з використанням платформи Expo, що дозволить створити застосунок для Android та iOS одночасно. Для збереження даних, синхронізації та реалізації обліку користувачів передбачається використання Firebase як основної хмарної платформи. Важливими завданнями також є забезпечення стабільної роботи застосунку в офлайн-режимі, підтримка локального кешування та налаштування гнучкої структури бази даних.

Логічну структуру застосунку BookHive можна уявити як багаторівневу модель, яка складається з чотирьох основних рівнів, що тісно взаємодіють між собою (рис. 1.2). До таких рівнів належать: інтерфейс користувача, логіка додатку, робота з даними та сторонні сервіси з ресурсами. Така архітектура дозволяє забезпечити чітке розмежування функціональних обов’язків між компонентами системи, що полегшує масштабування, тестування та супровід застосунку в майбутньому.

Рисунок 1.2 – Логічна структура застосунку

Проглянемо також структуру інтерфейсу застосунку BookHive (рис. 1.3). Центральним елементом навігаційної структури застосунку є головна сторінка, яка служить доступом до основних функціональних модулів. З неї користувач може швидко перейти до будь-якого з підрозділів, відповідно до своїх цілей: читання, організація бібліотеки, перегляд статистики, соціальна взаємодія або використання інтелектуальних рекомендацій.

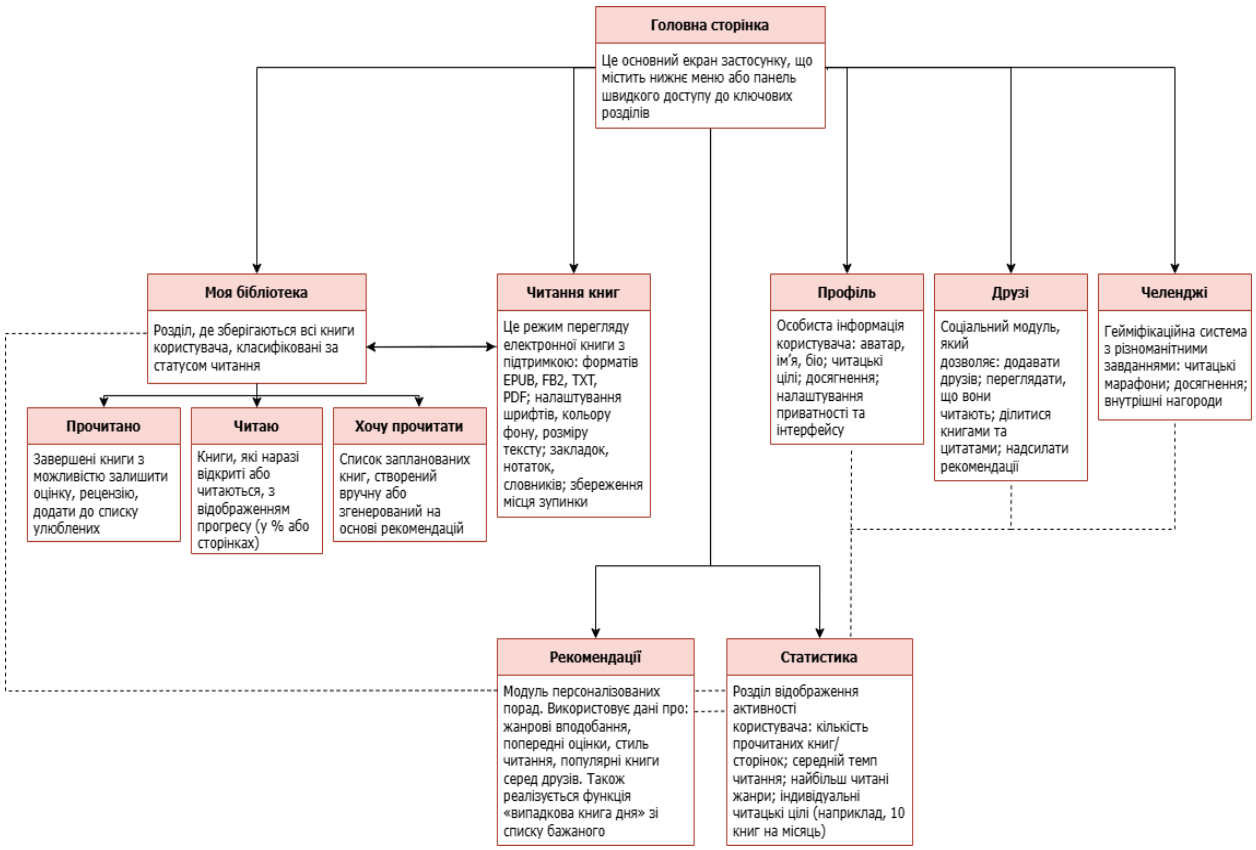

Рисунок 1.3 - Структура інтерфейсу застосунку

Отже, основними завданнями проєкту є: формування функціональної та інформаційної моделі застосунку; визначення структури інтерфейсу користувача; розробка рекомендаційного алгоритму; проєктування архітектури системи з урахуванням багаторівневої взаємодії між модулями; забезпечення синхронізації даних; реалізація соціального функціоналу та тестування застосунку та оцінка його зручності у реальних сценаріях використання.

Висновки до розділу

У розділі було проаналізовано предметну область, вивчено процес організації цифрового читання та виокремлено ключові проблеми, з якими стикаються користувачі під час взаємодії з традиційними електронними платформами. Також запропоновано концепцію застосунку BookHive як

універсального рішення, що поєднує читалку, систему обліку прогресу, персоналізовані рекомендації, механізми соціальної взаємодії та мотиваційні елементи.

Огляд існуючих аналогів засвідчив, що жодна з платформ не поєднує всі необхідні для сучасного читача функції. Це підтверджує актуальність і доцільність створення нового продукту, здатного задовольнити попит на комплексний і персоналізований сервіс читання. У результаті сформульовано мету проєкту, технічні засади реалізації та основні функціональні блоки системи, які в подальшому слугуватимуть основою для моделювання, програмної реалізації та тестування застосунку BookHive.

РОЗДІЛ 2
ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз предметної області

Аналіз предметної області – це початковий етап проєктування інформаційної системи. Він дозволяє систематизувати знання про досліджувану галузь , виявити основні сутності та зв’язки між ними, вхідні й вихідні дані, правила обробки інформації та обмеження. Також створює підґрунтя для проєктування системи, зокрема побудови інформаційної, логічної та функціональної моделе. Концептуальне моделювання є актуальнішим для галузі інформаційних систем, ніж будь-коли раніше, і вимагає нових методів і граматик для представлення знань [7].

Предметна область визначає сферу застосування майбутнього програмного забезпечення та окреслює межі його функціонування. У межах цього проєкту предметною областю є цифрова організація процесу читання електронних книжок, яка має у собі управління персональною бібліотекою, рекомендаційні механізми, відстеження читацького прогресу та соціальну взаємодію. Програмний продукт – мобільний застосунок BookHive – має на меті не лише підтримати звичку до читання, але й зробити цей процес більш ефективним, привабливим та зручним.

У результаті аналізу предметної області було:

1. Виділено основні об’єкти дослідження, до яких належать:

– користувач – основна сутність, що взаємодіє із системою;

– книга – об’єкт читання, що має метадані (автор, жанр, прогрес);

– рекомендація – автоматично згенеровані або надані іншими користувачами поради щодо читання;

– статистика – дані щодо активності користувача;

– читацькі виклики (челенджі);

– інтерфейс – вхідна точка доступу до функціоналу застосунку.

2. Побудовано інформаційну модель (рис. 2.1)

Об'єкт	Вхідні дані	Вихідні дані
Користувач	Логін, особисті дані, цілі читання	Профіль, читацька історія, статистика
Книга	Назва, автор, жанр, формат	Відображення у бібліотеці, статус читання, прогрес
Рекомендації	Оцінки, жанрові вподобання, поведінкова аналітика	Персоналізований список книг, книга дня
Статистика	Події читання, кількість сторінок, жанр, час	Таблиці, графіки прогресу, звіти
Виклики	Завдання, цілі, дата початку/завершення	Нагороди, досягнення, мотиваційні елементи

Рисунок 2.1 – Інформаційна модель системи

Модель демонструє взаємодію між об’єктами системи та типами даних, що входять і генеруються в процесі експлуатації. Далі розглянемо більш детально цю інформацію.

3. Визначено ключові обмеження на використання системи.

2.1.1 Вхідні дані

Вхідні дані – інформація, що надходить від користувача або зовнішніх джерел і використовується для ініціації, налаштування, персоналізації та супроводу основних процесів функціонування застосунку. Згідно з підходом аналізу вводу-обробки-виводу (IPO), правильне формування вхідних даних забезпечує надійну роботу всієї системи [8]. Розглянемо вхідні дані для застосунку BookHive більш детально та приклад структури вхідних даних (рис. 2.2).

Категорія	Приклад
Ідентифікаційні	Email: example@gmail.com; Ім'я: Валерія; Аватар: avatar01.jpg
Книга	Назва: *1984*; Автор: Дж. Орвелл; Формат: .epub; Жанр: антиутопія
Поведінкові	Прогрес: 45%; Оцінка: 5; Жанри: фантастика, історія; Рецензія: «Сильна книга»
Зовнішні джерела	Запит до Google Books API за ISBN: 9780451524935

Рисунок 2.2 – Приклад структури вхідних даних

Залежно від джерела, вхідні дані умовно поділяються на:

1. Персональні дані користувача. Ці дані вводяться користувачем під час реєстрації або редагування профілю:

– Ім'я користувача – використовується для персоналізації інтерфейсу;

– Email-адреса – для автентифікації, синхронізації та відновлення доступу;

– Аватар (фото профілю) – зберігається у базі даних або у хмарному середовищі

– Пароль або ключ доступу – зберігається у зашифрованому вигляді (не використовується напряму у логіці застосунку, але є частиною бекенду).

2. Контентні дані – дані, що стосуються читацького процесу:

– файли книг у підтримуваних форматах: .epub – основний формат з підтримкою розмітки та масштабування; .fb2, .txt – легкі текстові формати без складного форматування; .pdf – для збереження структури друкованого макету.

– метадані книги (можуть бути імпортовані, або введені вручну): назва, автор, рік видання, жанр, короткий опис, мова.

3. Поведінкові дані користувача – ці дані автоматично генеруються під час використання застосунку:

– прогрес читання (номер сторінки або відсоток завершення);

– оцінки книг або рейтин;

– рецензії та нотатки (текстові дані);

– уподобання (вибрані жанри, улюблені автори, історії вибору книг);

– активність у викликах (виконані або заплановані завдання);

– соціальні дії – додавання друзів, взаємодія з чужим профілем або рекомендаціями.

4. Зовнішні джерела (API-запити). Для розширення функціональності застосунку передбачена можливість використання відкритих або приватних API:

– обкладинки книг – можливе завантаження через метадані;

– рекомендації – на основі запитів до рекомендаційного модуля, що враховує жанри, історію перегляду, активність;

– додаткова інформація (біографія автора або подібні видання);

– тематичні виклики – завантаження з сервера або створення користувачем вручну.

Сукупність вхідних даних у застосунку BookHive дозволяє сформувати адаптивне середовище для читання, яке враховує індивідуальні потреби користувача.

2.1.2 Вихідні дані

Вихідні дані – це результат обробки вхідної інформації, що генерується системою у відповідь на дії користувача або логіку, закладену у програмний модуль. Ці дані відображають поточний стан системи, персоналізований контент, статистичні звіти, рекомендації, соціальну активність, а також визначають якість взаємодії користувача зі застосунком.

У межах проєктування застосунку BookHive до вихідних даних належать:

1. Динамічні списки книг. Залежно від статусу книги система автоматично розподіляє їх на такі категорії:

– «Прочитано» – книги, які завершені користувачем (вихідні дані включають: назву, автора, дату завершення, оцінку, рецензію, цитати);

– «Читаю зараз» – поточні книги з відображенням прогресу у відсотках або сторінках, поточне місце зупинки, таймер сесії;

– «Хочу прочитати» – список із відкладених книг, створений вручну.

2. Персоналізовані рекомендації, які формуються на основі: жанрових уподобань, читацької історії, активності в челенджах та соціальної взаємодії, популярних книг серед друзів.

Типові результати включають: «Книги, які можуть Вас зацікавити», «Новинки жанру, який Ви читаєте найчастіше», «Рекомендовані друзями».

Формат представлення рекомендацій: списки з картками книг (назва, автор, зображення обкладинки, кнопка «додати до бібліотеки» або «почати читати».

3. Статистичні дані користувача. Вони використовуються для візуалізації прогресу, мотивації та аналізу:

– загальна кількість прочитаних книг за період;
– середній темп читання;
– популярні жанри;
– час, витрачений на читання;
– динаміка активності.

4. Соціальні дії. Система підтримує вихідні дані, пов’язані з соціальною активністю користувача:

– стрічка друзів – показує, хто що читає, рецензії;
– челенджі – участь користувача у викликах (назва, прогрес, дедлайн);
– цитати та рецензії – короткі текстові пости, якими ділиться користувач;
– нотифікації (сповіщення).

Сформовані вихідні дані забезпечують зворотній зв’язок із користувачем, дозволяють візуалізувати його активність, мотивувати та аналізувати.

2.1.3 Обмеження

У процесі проєктування мобільного застосунку BookHive необхідно враховувати низку обмежень, які стосуються технічної реалізації, збереження

та обробки даних, а також забезпечення безпечної інформації. До ключових обмежень належать:

– формати файлів. Система підтримає лише визначені формати електронних книг: .epub, .fb2, .txt, .pdf. Інші формати не обробляються та потребують попередньої конвертації.

– обсяг збереження. У зв’язку з обмеженістю мобільної пам’яті, реалізовано лімітований обсяг для збереження книг у локальному кеші (наприклад, не більше 500 МБ без хмарної синхронізації).

– структурні обмеження бази даних. Через обрану модель даних (реляційну або документоорієнтовану) виникає необхідність дотримання чіткої схеми зв’язків між об’єктами. Це впливає на гнучкість масштабування.

– конфіденційність і безпека. Дані користувача підлягають захисту згідно з політикою приватності та міжнародними стандартами (наприклад, GDPR). Це вимагає шифрування, автентифікації та обмеженого доступу до чутливої інформації.

– обмеження API. Інтеграція з зовнішніми джерелами (наприклад, книжкові бази або обкладинки) залежить від умов ліцензування, квот на запити та стабільності сторонніх сервісів.

2.2 Проєктування системи

Проєктування системи передбачає створення логічної та структурної основи майбутнього застосунку BookHive. На цьому етапі формується архітектура, визначаються ключові компоненти, взаємозв’язки між ними, а також готується технічна база для реалізації функціоналу. Основними завданнями є побудова база даних, об’єктно-орієнтованої моделі, а також початкове планування структури інтерфейсу користувача.

2.2.1 Проєктування бази даних

Одним із ключових етапів створення інформаційної системи є проєктування бази даних, яка забезпечує ефективне зберігання, доступ та обробку структурованих даних. Вона є фундаментом для реалізації логіки мобільного застосунку BookHive, що надає користувачам функціонал для збереження книг, відстеження прогресу читання, ведення статистики, участі в читацьких викликах тощо. Для опису логічної структури бази даних було розроблено ER-модель (рис. 2.3), яка відображає сутності системи, їхні атрибути та взаємозв'язки.

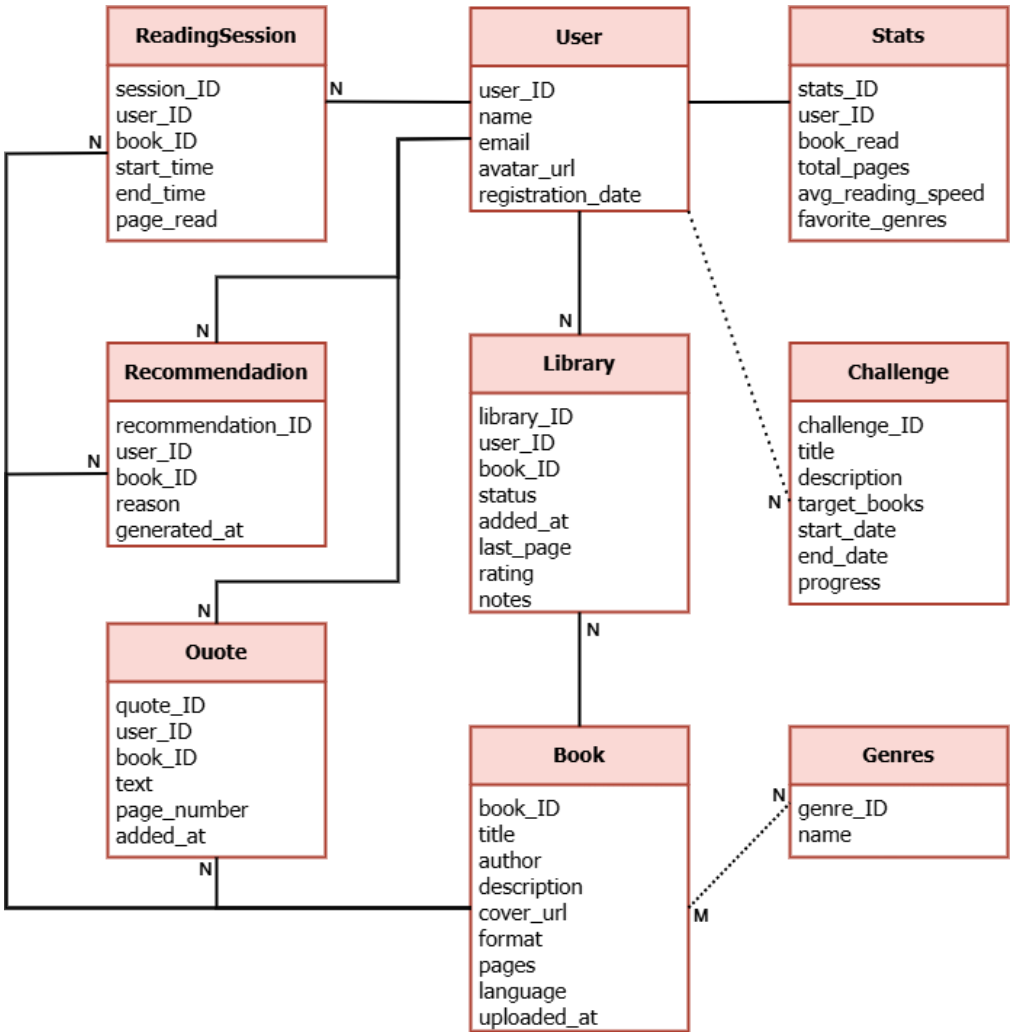

Рисунок 2.3 – ER-модель бази даних інформаційної системи BookHive

Нижче у таблиці 2.1 подано перелік основних сутностей, які використовуються в моделі бази даних, разом із відповідними атрибутами, що

визначають їхню функціональність та інформаційне наповнення. Ця структура слугує основою для побудови взаємозв'язків між сутностями та реалізації функціональних можливостей застосунку.

Таблиця 2.1 – Сутності моделі та атрибути

Сутності	Атрибути
1	2
User (Користувач) – сутність описує зареєстрованого користувача застосунку	user_id – унікальний ідентифікатор користувача; name – ім'я; email – електронна пошта (унікальна); avatar_url – зображення профілю; registration_date – дата реєстрації;
Book (Книга) – описує електронну книгу, яка доступна в системі	book_id – унікальний ідентифікатор; title – назва книги; author – автор(и); description – короткий опис; cover_url – обкладинка; format – формат файлу (EPUB, FB2, PDF, TXT); pages – кількість сторінок; language – мова книги; uploaded_at – дата додавання до системи
Library (Бібліотека) – містить інформацію про зв'язок між користувачем та гнигою	library_id – унікальний ідентифікатор запису; user_id – посилання на користувача; book_id – посилання на книгу; status – стан: «читаю», «прочитано», «у планах»; added_at – дата додавання до бібліотеки; last_page – остання прочитана сторінка; rating – оцінка користувача; notes – особисті нотатки (опціонально).
ReadingSession (Сесія читання) – фіксує дії користувача під час читання	session_id – унікальний ідентифікатор; user_id – ідентифікатор користувача; book_id – ідентифікатор книги; start_time – час початку сесії; end_time – час завершення; pages_read – кількість прочитаних сторінок.
Stats (Статистика) – агрегована інформація про читацьку активність	stats_id – унікальний ідентифікатор; user_id – посилання на користувача; books_read – загальна кількість прочитаних книг; total_pages – загальна кількість сторінок; avg_reading_speed – середній темп читання; favorite_genres – найчастіше обрані жанри.
Genres (Жанри) – категорія книжок	genre_id – унікальний ідентифікатор жанру; name – назва жанру (наприклад, «Фентезі», «Наукова фантастика»).
Quote (Цитата) – користувач може зберігати улюблені цитати з книг	quote_id – унікальний ідентифікатор; user_id – посилання на автора цитати; book_id – посилання на джерело; text – текст цитати;

Продовження таблиці 2.1

1	2
	page_number – номер сторінки; added_at – дата додавання.
Challenge (Виклик/Челенджі) – мотиваційні завдання для користувачів	challenge_id – унікальний ідентифікатор; title – назва челенджу; description – умови участі; target_books – кількість книг для виконання; start_date, end_date – тривалість; progress – кількість завершених книг користувачем.
Recommendation (Рекомендація) – інтелектуальні поради для користувача на основі аналітики	recommendation_id – унікальний ідентифікатор; user_id – кому призначена; book_id – рекомендована книга; reason – підстава для рекомендації (напр., «схоже на прочитане»); generated_at – дата створення рекомендації.

Структура бази даних побудована з урахуванням модульності та можливості масштабування. У разі потреби можуть бути додані нові сутності, такі як Activity (дії користувача), Notification (сповіщення), Achievement (досягнення), не порушуючи цілісності моделі.

Завдяки чітко побудованим зв’язкам забезпечується:

- цілісність даних через зовнішні ключі;
- можливість складної аналітики (через агрегування статистики);
- гнучкість у розширенні функціоналу.

Таким чином, проєктування бази даних є фундаментом для логіки інформаційної системи та забезпечує її узгоджену та надійну роботу.

2.2.2 Побудова об’єктно-орієнтованої моделі

Об’єктно-орієнтовне програмування (ООП) (рис. 2.4) – парадигма програмування, яка базується на концепції «об’єктів» – абстрактних сутностей, що об’єднують стан (дані) і поведінку (методи). Згідно з сучасними підходами, ООП є одним із найпоширеніших методів створення програмного забезпечення, особливо в мобільній розробці, оскільки забезпечує повторне використання коду, модульність та спрощує масштабування проєктів [9].

ООП-моделювання є важливим етапом у проєктуванні системи, оскільки дозволяє:

– розподілити систему на модулі, що відповідають окремим сутностям;

– чітко визначити відповідальність кожного модуля;

– забезпечити повторне використання коду через наслідування;

– легко підтримувати та масштабувати систему.

Як зазначають дослідники, об’єктно-орієнтоване програмування зосереджується на модульності, повторному використанні коду та масштабованості, що робить його надзвичайно придатним для складних програмних систем [10]. Крім того, воно заохочує до розбиття системи на менші керовані модулі, спрощує тестування й обслуговування та підтримує легку адаптацію до змін у вимогах [11].

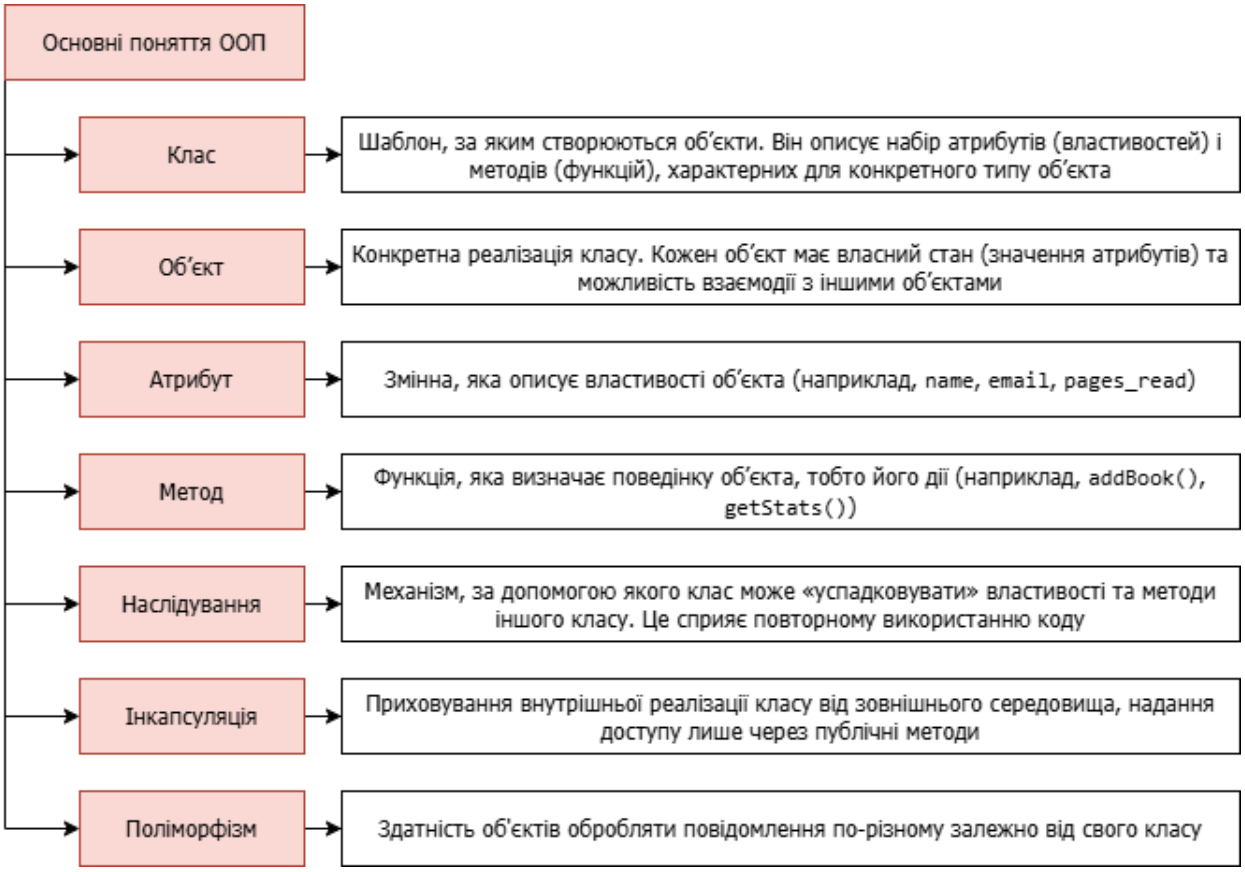

Рисунок 2.4 – Основні поняття ООП

У контексті застосунку BookHive об’єкти – це користувачі, книги, челенджі, рекомендації, які мають свої властивості та поведінку. На основі попередньо розробленої ER-моделі (див. рис. 2.3) та структури бази даних, для

побудови об’єктно-орієнтованої моделі системи виділено набір ключових класів. Кожен клас реалізує відповідну сутність системи та відповідає за певний аспект функціонування системи, забезпечує зберігання, обробку або відображення відповідних даних. Також класи мають власні атрибути (властивості) та методи (функції), які реалізують поведінку об’єкта. Розберемо класи системи більш детально:

User (рис. 2.5) – представляє зареєстрованого користувача. Містить дані профілю, історію активностей, зв’язки з іншими класами (читання, цитати, рекомендації). Має методи автентифікації, оновлення профілю, перегляд статистики.

Атрибут	Тип	Опис
userId	Intager	Унікальний індефікатор
name	String	Ім'я користувача
email	String	Елетронна пошта
avatarUrl	String	Посилання на аватар
registrationDate	Date	Дата реєстрації

Метод	Опис
login()	Авторизація користувача
logout()	Вихід з облікового запису
updateProfile()	Зміна аватару або імені
getStats()	Отримання читацької статистики

Рисунок 2.5 – Основні атрибути та методи класу User

Book (рис. 2.6) – описує електронну книгу. Містить інформацію про назву, автора, формат, обкладинку, кількість сторінок. Реалізує функції перегляду опису, запуску читання, додавання до бібліотеки.

Атрибут	Тип	Опис
bookId	Integer	Унікальний індефікатор
title	String	Назва книги
author	String	Автор
description	Text	Короткий опис
coverUrl	String	Посилання на обкладинку
format	String	Формат файлу
pages	Integer	Кількість сторінок
language	String	Мова книги
uploadedAt	Date	Дата додавання до системи

Метод	Опис
getInfo()	Отримати повну інформацію про книгу
download()	Завантажити файли книги
preview()	Переглянути перші сторінки

Рисунок 2.6 – Основні атрибути та методи класу Book

Library (рис. 2.7) – опосередковує зв’язок між User та Book. Відображає стан книги в бібліотеці користувача: читаю/прочитано/у планах. Містить дані про оцінку, нотатки, останню прочитану сторінку.

Атрибут	Тип	Опис
libraryId	Integer	Унікальний запис у бібліотеці
userId	Integer	Посилання на користувача
bookId	Integer	Посилання на книгу
status	String	Статус ("читаю", "прочитанно" тощо)
addedAt	Date	Дата додавання до бібліотеки
lastPage	Integer	Остання прочитана сторінка
rating	Float	Оцінка користувача
notes	Text	Особисті нотатки (опціально)

Метод	Опис
updateStatus()	Оновлення статусу читання
addNote()	Додати нотатку до книги
rateBook()	Виставити оцінку

Рисунок 2.7 – Основні атрибути та методи класу Library

ReadingSession (рис. 2.8) – фіксує сесію читання – її початок, завершення та кількість прочитаних сторінок. Взаємодіє з аналітичними сервісами та класом Stats.

Атрибут	Тип	Опис
sessionId	Intager	Індефікатор сесії
userId	Integer	Користувач
bookId	Integer	Книга
startTime	DateTime	Час початку
endTime	DateTime	Час завершання
pagesRead	Integer	Прочитані сторінки

Метод	Опис
start()	Старт нової сесії
end()	Завершення сесії
getDuration()	Повертає тривалість читання

Рисунок 2.8 – Основні атрибути та методи класу ReadingSession

Stats (рис. 2.9) – Агрегує статистику користувача: кількість прочитаних книг, загальна кількість сторінок, середній темп читання. Включає методи аналітики та генерації зведення даних.

Атрибут	Тип	Опис
statsId	Integer	Унікальний індефікатор статистики
userId	Integer	Посилання на користувача
booksRead	Integer	Загальна кількість прочитаних книг
totalPages	Integer	Сума сторінок усіх прочитаних книг
avgReadingSpeed	Float	Середня швидкість читання (сторінок/день)
favoriteGenres	String	Найчастіше обрані книги

Метод	Опис
calculateStats()	Розрахунок статистики на основі сесій читання
getTopGenres()	Повертає список улюблених жанрів
updateStats()	Оновлення значень після нової прочитаної книги

Рисунок 2.9 – Основні атрибути та методи класу Stats

Quote (рис. 2.10) – Відповідає за збереження цитат. Містить текст, номер сторінки, посилання на книгу та користувача. Може містити мітки або теги.

Атрибут	Тип	Опис
quoteId	Integer	Індифікатор цитати
userId	Integer	Хто зберіг
bookId	Integer	З якої книги
text	Text	Текст цитати
pageNumber	Integer	Номер сторінки
addedAt	Date	Дата додавання

Метод	Опис
save()	Зберігає нову цитату
delete()	Видаляє цитату

Рисунок 2.10 – Основні атрибути та методи класу Quote

Genres (рис. 2.11) – Служить для категоризації книг. Містить список жанрів, доступних у системі. Дозволяє фільтрацію й побудову рекомендацій.

Атрибут	Тип	Опис
genreId	Integer	Унікальний індефікатор жанру
name	String	Назва жанру

Метод	Опис
getBookd()	Отримати список книг у цьому жанрі

Рисунок 2.11 – Основні атрибути та методи класу Genres

Challenge (рис. 2.12) – Модель читацьких викликів. Містить назву, умови, терміни, прогрес виконання. Використовується у розділі мотивації користувача.

Атрибут	Тип	Опис
challengeId	Integer	Індефікатор челенджу
title	String	Назва виклику
description	Text	Умови участі
targetBooks	Integer	Скільки книг потрібно прочитати
startDate	Date	Початок виклику
endDate	Date	Завершення
progress	Integer	Поточний прогрес користувача

Метод	Опис
updateProgress()	Оновлює кількість завершених книг у виклику
checkCompletion()	Перевіряє, чи виконано завдання

Рисунок 2.12 – Основні атрибути та методи класу Challenge

Recommendation (рис. 2.13) – Генерує та зберігає рекомендації для користувача на основі даних читання, жанрів та історії. Може бути доповнена поясненням причин рекомендації.

Атрибут	Тип	Опис
recommendationId	Integer	Індифікатор рекомендації
userId	Integer	Кому призначено
bookId	Integer	Яку книгу рекомендовано
reason	String	Причина рекомендації ("схоже на попереднє" тощо)
generatedAt	Date	Дата формування рекомендації

Метод	Опис
generate()	Генерує рекомендацію на основі читацького профілю
display()	Виводить рекомендацію користувачу

Рисунок 2.13 – Основні атрибути та методи класу Recommendation

Всі ці класи взаємодіють між собою за допомогою асоціацій (відношень), що моделюють зв’язки між об’єктами системи. Наприклад, кожен об'єкт Library асоційований з об'єктами User і Book, а Stats пов’язаний з

ReadingSession. Надалі ці класи будуть реалізовані як компоненти програмного коду, що дозволить перенести логіку з моделі на етап реалізації.

Перейдемо до UML-діаграми класів, вона є одним із ключових інструментів ООП. Діаграма дозволяє візуалізувати структуру системи, зобразити класи, їхні атрибути, методи, а також взаємозв'язки між ними. Такий підхід полегшує розуміння архітектури застосунку, забезпечує цілісність логіки та підтримує модульність при подальшій розробці.

Для мобільного застосунку BookHive було побудовано UML-діаграму класів (рис. 2.14), яка охоплює всі основні компоненти системи: User, Book, Library, ReadingSession, Stats, Genre, Quote, Challenge, Recommendation.

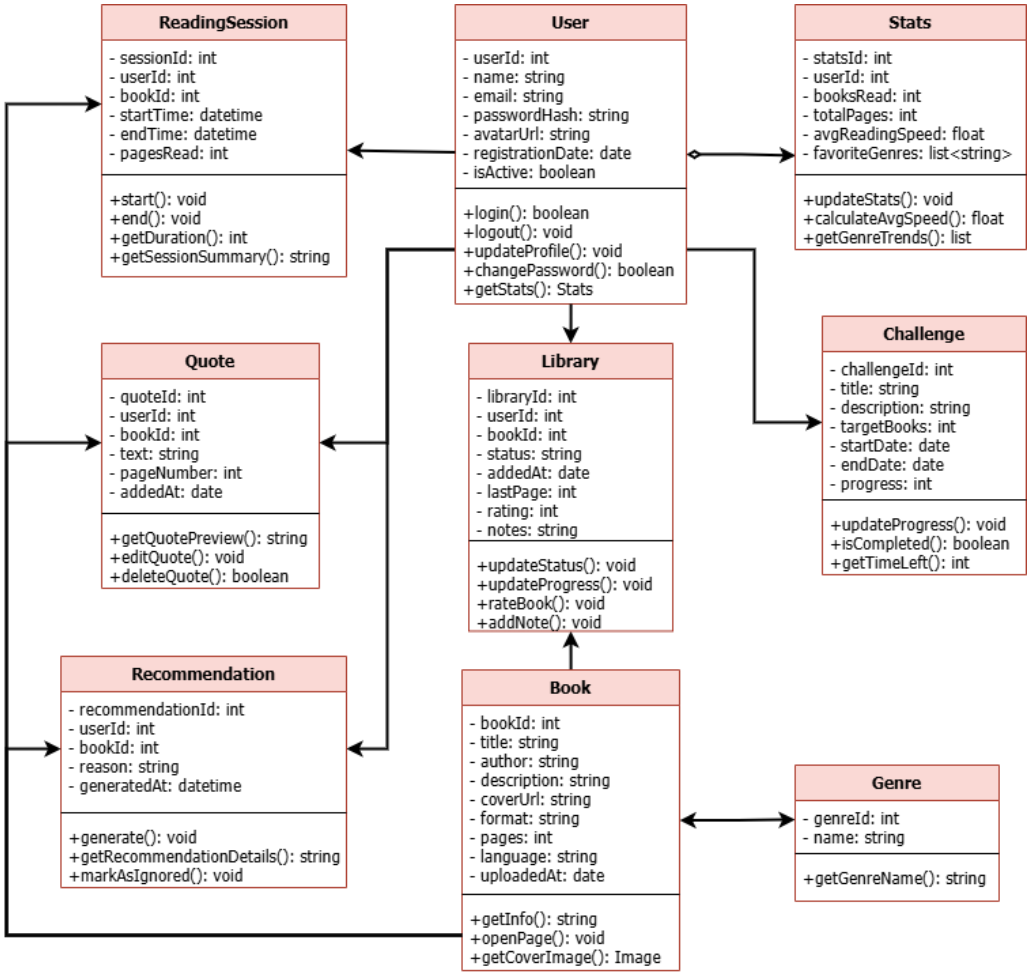

Рисунок 2.14 – UML-діаграма класів мобільного застосунку BookHive

На діаграмі класів відображено: атрибути кожного класу; основні методи, які реалізують функціональну поведінку; асоціації між класами;

агрегацію; композицію; зв’язки "один-до-багатьох" та "багато-до-багатьох", де це необхідно.

Об’єктно-орієнтована модель застосунку BookHive є гнучкою, масштабованою та зручною для супроводу. Її чітке розділення відповідальності між класами забезпечує легке розширення функціоналу, додавання нових компонентів та адаптацію під майбутні зміни. Структура підтримує модульне тестування й оновлення без порушення загальної архітектури, саме це робить модель ефективною основою для подальшої реалізації системи.

2.2.3 Проєктування інтерфейсу користувача

Інтерфейс користувача – ключовий елемент взаємодії між користувачем та самим мобільним застосунком. Його завдання – зробити функціонал інтуїтивно зрозумілим, візуально привабливим та зручним для щоденного використання. При проєктуванні інтерфейсу для застосунку BookHive було враховано сучасні вимоги UX/UI-дизайну, адаптивність до різних розмірів екранів, підтримку темної теми та забезпечення доступності. Для реалізації планового дизайну проєкта використовувався Figma.

У мобільному застосунку BookHive передбачено такі основні екрани: Головний екран, Моя бібліотека, Рекомендації, Статистика, Профіль, Челенджі та інші. Колірна палітра тепла та має приглушені кольори, які створюють атмосферу затишку – ідеально для книжкового застосунку.

У застосунку реалізовано нижню навігаційну панель із вкладками, а також свайпи між розділами та контекстне меню у вигляді трьох крапок у верхньому куті для додаткових дій (наприклад, редагування або фільтрації контенту).

Розглянемо спочатку вікна входу та реєстрації (рис. 2.15), вони були створені з урахуванням мінімалізму та зручності для користувача. Екран входу містить поля для введення імені користувача та пароля, кнопку входу та альтернативні методи автентифікації (Google, локальний профіль, Facebook). Екран реєстрації дозволяє новим користувачам створити свій обліковий запис.

Обидва екрани мають зрозумілу навігацію між собою та виконані в єдиному візуальному стилі.

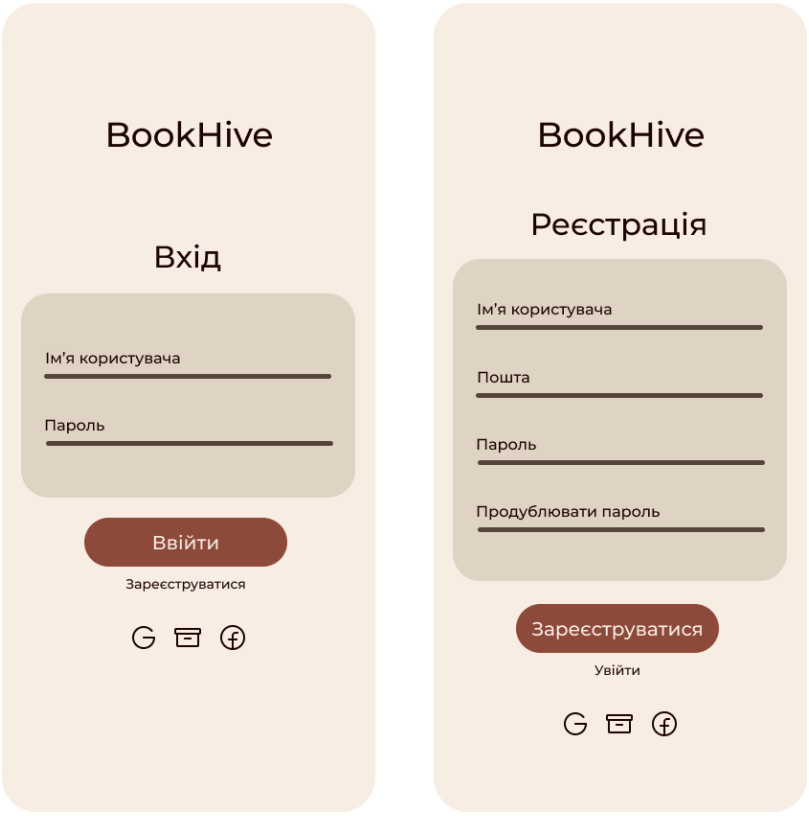

Рисунок 2.15 – Вікна входу та реєстрації

Головний екран (рис. 2.16) э основним центром навігації між іншими вікнами. Він містить блок «Продовжити», де буде показана поточна книга та прогрес. Також є секція «Жанри» у вигляді кнопок та «Рекомендації» з обкладинками книг, які були індивідуально підібрані під користувача.

У застосунку маємо є бічне меню навігації (рис. 2.16), воно відкривається через кнопку у верхньому лівому куті. У ньому користувач має доступ до профілю, бібліотеки, колекції, налаштувань, файли з пристрою, а також там можна залишити відгук на застосунок або ознайомитися з детальною інформацією.

Екран рекомендацій (рис. 2.16) показую персоналізовані добірки книг. Тут присутні секції: «Вам може сподобатися» та «Популярне зараз». Усі блоки

виконано у візуальному стилі застосунку з акцентами з фірмової палітри кольорів.

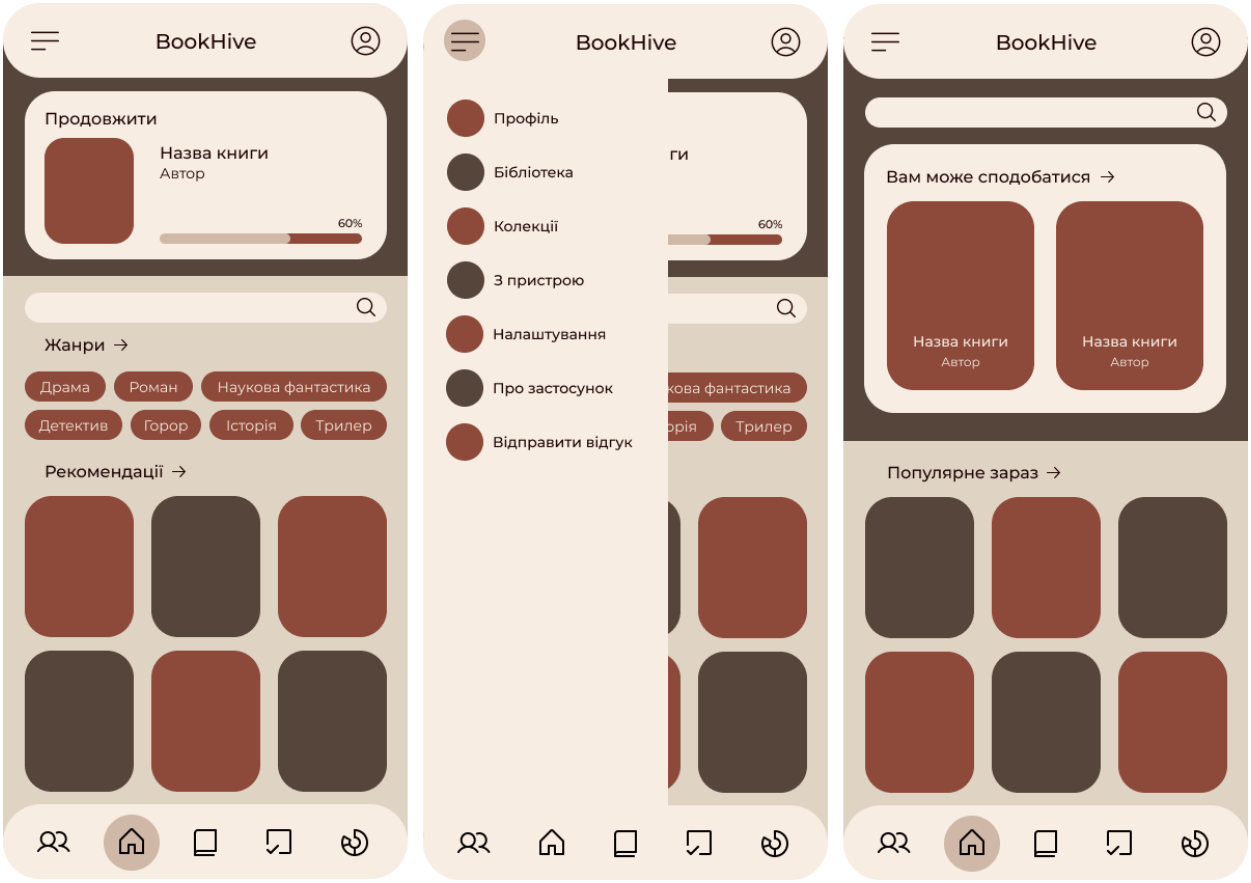

Рисунок 2.16 – Головний екран, бічне меню навігації та екран рекомендацій

Екран бібліотеки (рис. 2.17) надає доступ до усіх книг користувача, які будуть розділені за статусом: «Прочитано», «Читаю» та «Хочу прочитати». Кожна книга представлена карткою з власною обкладинкою, назвою, автором та прогресом прочитаного. Окремо у «Прочитаних» також вказано особистий рейтинг книги за 10-ти бальною шкалою. Як і всі екрани, інтерфейс виконано у фірмовій кольоровій палітрі з інтуїтивно зрозумілою структурою.

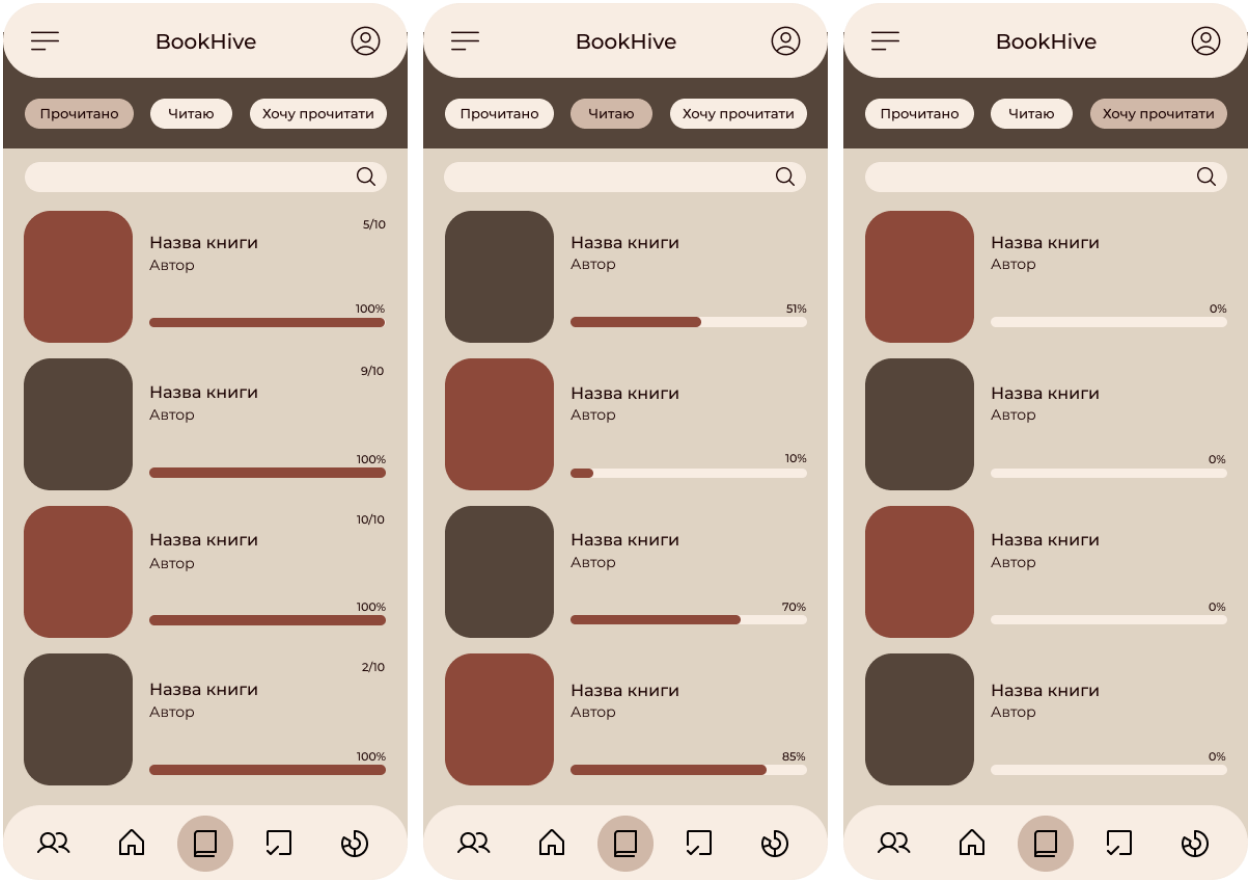

Рисунок 2.17 – Екрани бібліотеки: «Прочитано», «Читаю», «Хочу прочитати»

Екран перегляду книги (рис. 2.18) складається з трьох головних частин: опис самої книги, області для читання та блок налаштування.

На першому екрані відображається обкладинка до книги, статус прочитання, назва, автор та короткий опис книги. Це дозволяє користувачу швидко зорієнтуватися в інформації. Додатково є змога додати рейтин, побачити відсоток прочитаного та перейти до обговорення книги. Можна ще подивитися свої виділені цитати та закладки у книзі.

Основна частина читання має зручний розмір шрифту та гарну контрастність, що дозволяє менше навантажувати очі. Зверху показана сторінка на якій зараз знаходиться користувач, а знизу – прогрес читання.

Налаштування читання дозволяють змінювати шрифт, розмір тексту, тип листання сторінок (горизонтальне або вертикальне), міжрядковий інтервал. Така гнучкість забезпечує комфортне користування та більш індивідуальний підхід,

Рисунок 2.18 – Екран перегляду книги та додаткові налаштування

Екран профілю користувача (рис. 2.19) відображає загальну інформацію про активність читача: кількість прочитаних книг, витрачений час та загальна кількість прочитаних сторінок.

Також на екрані відображено улюблені жанри користувача, здобуті винагороди за пройдені челенджі та індивідуальні рекомендації книг. Це сприяє персоналізації досвіду користувача та підвищує його залучення до застосунку.

Екран друзів (рис. 2.19) дає змогу переглядати список контактів, додавати нових друзів і керувати заявками. Для кожного користувача передбачено аватар, статус і кнопку для перегляду детальної інформації, що спрощує соціальну взаємодію в межах додатку.

Рисунок 2.19 – Екран профілю та списку друзів

Екран статистики (рис. 2.20) надає користувачу візуальне представлення прогресу читання у вигляді кругової діаграми та графіка активності за днями тижня. Також виводиться ключові показники, такі як: кількість прочитаних книг, середній час читання на день та кількість днів підряд, коли користувач читав книги. Це дозволяє користувачу відстежувати свої досягнення та зберігати мотивацію

Екран викликів (челенджів) (рис. 2.20) відображає список активностей для підвищення залученості. Користувач може долучатися до нових челенджів або стежити за прогресом поточних, може додавати і свої виклики. Кожен виклик має назву, короткий опис, ілюстрацію та індикатор виконання, що сприяє гейміфікації процесу читання.

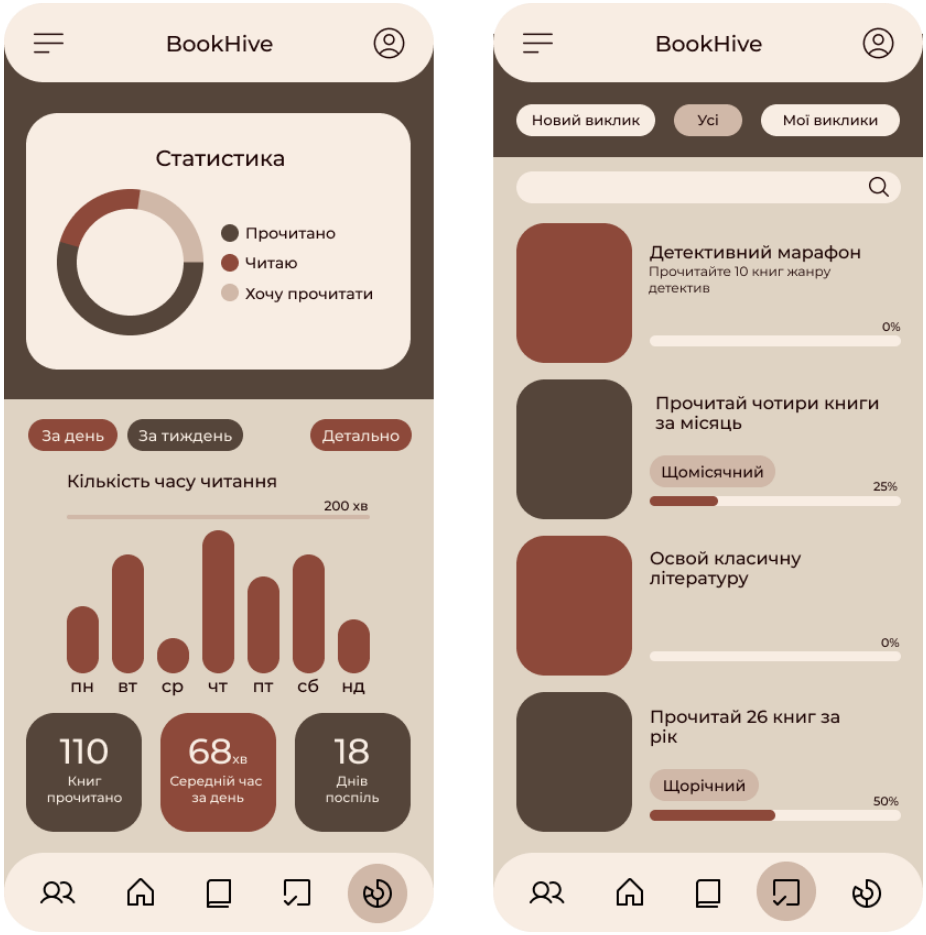

Рисунок 2.20 – Екран статистики та викликів

UX/UI підходи:

– дотримання принципу мінімалізму (інтерфейс без перевантаження);
– адаптивність до різних типів екранів;
– підтримка темної теми для зменшення навантаження на очі;
– використання чіткої типографії і зрозумілої іконографії;
– інтерактивні елементи мають візуальний зворотній зв’язок.

Проєктування інтерфейсу було виконано з урахуванням цільової аудиторії – користувачів, які активно читають і прагнуть зручно керувати своєю електронною бібліотекою

2.3 Математичне та алгоритмічне забезпечення

Математичне та алгоритмічне забезпечення визначає логіку функціонування основних функцій застосунку та забезпечують точність та ефективність його роботи. Воно об’єднує формалізацію процесів, які відповідають за аналітику, персоналізацію та обробку даних.

У мобільному застосунку BookHive алгоритмічне забезпечення реалізує:

- Побудову рекомендацій книг;
- Обчислення прогресу читання;
- Прогноз завершення читання книги;
- Аналіз читацької активності.

Кожен з алгоритмів описаний текстово та схематично для подальшої реалізації в середовищі React Native з використанням Firebase.

Однією з ключових функцій застосунку BookHive є генерація персоналізованих рекомендацій книг для користувача. Він має на меті запропонувати користувачу книги, які можуть його зацікавити, на основі жанрових уподобань, історії читання та активності в застосунку. Основна логіка, тобто математична модель, реалізується через контент-орієнтований підхід, у якому використовується аналіз жанрів, авторів та статистики користувача. Формула для зваженої оцінки буде виглядати наступним чином:

$$R = \alpha \times \text{genre\_score} + \beta \times \text{author\_score} + \gamma \times \text{rating\_score} \quad (2.1)$$

де: genre_score – відсоток жанрової відповідності;

author_score – частота переглядів/читань автора;

rating_score – середній рейтинг схожих книг;

α, β, γ – вагові коефіцієнти, які можна налаштовувати (наприклад, 0.4 / 0.3 / 0.3).

Представимо цей алгоритм у вигляді блок-схеми (рис. 2.21)

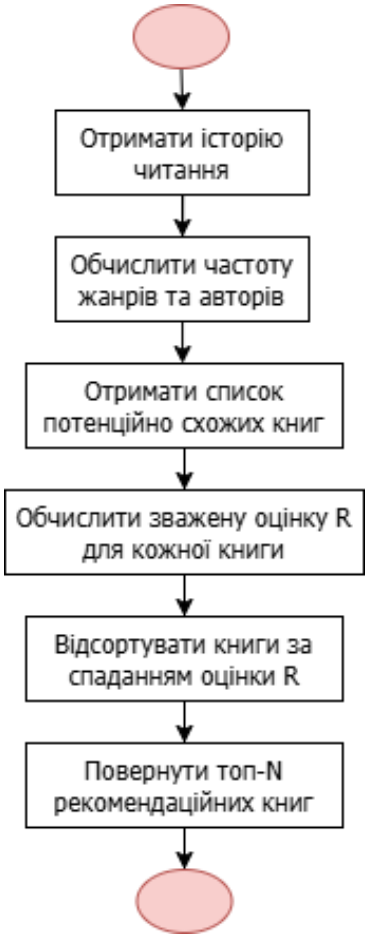

Рисунок 2.21 – Блок-схема алгоритму персоналізованих рекомендацій

Наступним розглянемо алгоритм розрахунку прогресу читання, який використовується для відображення відсотка прочитаної частини книги кожним користувачем. Ця функціональність є базою для інтерфейсу користувача, дозволяє стежити за динамікою читання та використовується в читацькій статистиці. Математична модель та формула будуть виглядати наступним чином:

$$\text{Прогрес} = \frac{\text{Остання прочитана сторінка}}{\text{Загальна кількість сторінок}} \times 100\% \qquad (2.2)$$

де: Остання прочитана сторінка – значення, яке оновлюється після кожної сесії читання;

Загальна кількість сторінок – атрибут сутності Book.

Блок схема для цього алгоритму буде представлена на рисунку 2.22.

Рисунок 2.22 – Блок-схема алгоритму розрахунку прогресу читання

Далі розглянемо алгоритм прогнозу завершення книги та блок-схему до нього (рис. 2.23). Метою даного алгоритму є оцінювання залишкового часу, необхідного користувачу для завершення читання поточної книги, на основі середнього темпу читання та кількості сторінок, що залишились. Математична модель:

$$T = \frac{P_{залишилося}}{v_{ср}} \quad (2.3)$$

де: $P_{залишилося} = P_{загальна} - P_{поточна}$ – кількість сторінок, що залишилися до завершення;

$v_{ср}$ – середня швидкість читання користувача (визначається за даними з сутності Stats);

T – прогнозований час у годинах (або іншій обраній одиниці).

Рисунок 2.23 – Прогноз завершення книги

Також важливою функцією є ведення читацької аналітики. Відстеження активності дозволяє не лише формувати статистику, але й будувати рекомендації, мотивувати користувачів через челенджі, а також аналізувати динаміку читання. Основні метрики: загальна кількість прочитаних сторінок за день; середня тривалість сесії читання; найкращі дні тижня; найчастіше обирані жанри. У математичній моделі будуть використовуватися для вирази:

Середня тривалість сесії:

$$\bar{t} = \frac{\sum_{i=1}^{n}(t_{\text{зак}_i} - t_{\text{початку}_i})}{n} \qquad (2.4)$$

Кількість прочитаних сторінок за день:

$$\mathrm{P}_{\text{д}} = \sum_{j=1}^{m} p_j \qquad (2.5)$$

де n – кількість сторінок;

$t_{\text{зак}_i}, t_{\text{початку}_i}$ – час початку та закінчення процесу читання;

p_j – сторінки, прочитані в j-тій сесії за день.

Блок-схема описує поетапний процес збору сесій читання, обчислення метрик та збереження результатів у базі даних (рис. 2.24).

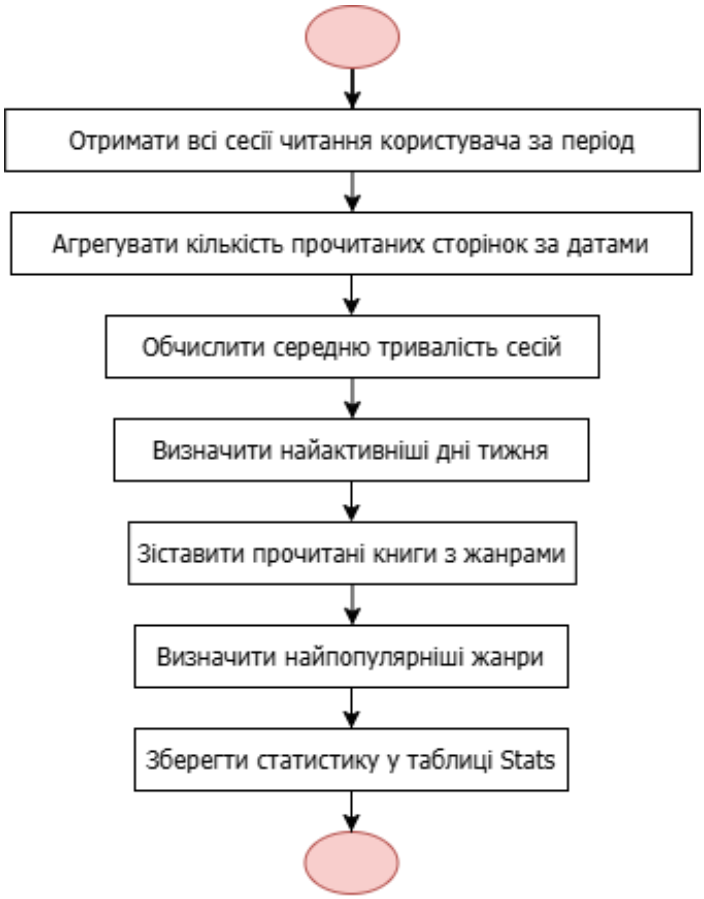

Рисунок 2.24 – Відстеження читацької активності

Можна сказати, що у межах математичного та алгоритмічного забезпечення мобільного застосунку BookHive було розроблено набір базових алгоритмів, що забезпечують ключовий функціонал системи.

Застосування алгоритмів дозволяє забезпечити персоналізацію, аналітику та зручність взаємодії з системою. Використання математичних формул, блок-схем і структурованих обчислень сприяє реалізації інтелектуального функціоналу без суттєвого навантаження на ресурси пристрою.

Висновки до розділу

У результаті проведеного аналізу та проєктування було проведене інформаційне та математичне забезпечення проєкту, що дозволило сформувати цілісне бачення майбутнього застосунку BookHive. Було визначено ключові об’єкти, які лежать в основі функціонування системи, та сформовано логічну структуру обробки даних. Створена база даних дозволяє ефективно зберігати і взаємопов’язувати дані про користувачів, книги, сесії читання та інші сутності.

Розроблена об’єктно-орієнтована модель системи забезпечує гнучкість та масштабованість, а спроєктований інтерфейс користувача враховує зручність і доступність взаємодії. Впроваджені математичні алгоритми підтримують персоналізацію досвіду, дозволяючи адаптувати рекомендації, відстежувати прогрес та прогнозувати читацьку активність.

Таким чином, створена основа дає змогу перейти до етапу реалізації функціональних модулів і забезпечує технічну готовність до подальшого розвитку застосунку.

РОЗДІЛ 3
ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Засоби розробки

Для розробки мобільного застосунку BookHive було обрано сучасний технологічний стек, що забезпечує кросплатформність, зручність розгортання та гнучкість масштабування. Вибір програмного й апаратного забезпечення здійснювався з урахуванням специфіки розробки інформаційних систем, зокрема відповідно до моделі архітектури відкритих систем OSI.

У процесі реалізації застосунку були використані наступні компоненти технологічного стеку:

– React Native [20] + Expo [21] – фреймворк для розробки кросплатформених мобільних застосунків, що дозволяє створювати рішення одночасно для Android та iOS з єдиною кодовою базою. Він базується на мові програмування: JavaScript з TypeScript. Expo значно спрощує процес налаштування середовища, розгортання та тестування.

– Firebase [22] – хмарна платформа від Google, що забезпечує функціональність бекенду: автентифікацію користувачів, зберігання даних у реальному часі (Realtime Database / Firestore), хмарне сховище для файлів та аналітику.

– додаткові бібліотеки: React Navigation – бібліотека для реалізації навігації між екранами; AsyncStorage – локальне сховище даних на пристрої; Expo SDK – набір готових модулів і компонентів для мобільної розробки (камера, доступ до файлів, push-сповіщення тощо).

Цей вибір середовища розробки дозволяє створити гнучку та масштабовану систему, легко адаптовану до змін і оновлень.

Згідно з архітектурою відкритих систем OSI, інформаційна система реалізує такі рівні:

– фізичний та канальний рівні: забезпечуються комп’ютерною мережею підприємства;

– мережевий рівень: налаштовується через підключення до інтернету (локальна мережа або Wi-Fi);

– транспортний рівень: реалізований через HTTPS-з'єднання Firebase;

– прикладний рівень: мобільний застосунок, який реалізує логіку взаємодії з користувачем.

3.2 Вимоги до технічного та програмного забезпечення

Для забезпечення ефективної роботи під час розробки мобільного застосунку BookHive було визначено перелік необхідних технічних і програмних засобів, які забезпечують комфортне програмування, тестування та запуск застосунку як у середовищі розробки, так і на реальному пристрої.

Технічне забезпечення. Розробка здійснювалася на персональному комп’ютері із наступними характеристиками:

– процесор: Intel Core i5 (або аналогічний за продуктивністю);

– оперативна пам’ять: не менше 8 ГБ для забезпечення стабільної роботи емуляторів та IDE;

– накопичувач: SSD-накопичувач обсягом від 256 ГБ для зберігання проєкту та залежностей;

– операційна система: Windows 11. Також підтримуються Windows 10, macOS та дистрибутиви Linux, оскільки всі обрані інструменти кросплатформенні;

– додаткові умови: стабільне інтернет-з’єднання, необхідне для синхронізації з Firebase та завантаження залежностей; застосунок на мобільному пристрої для тестування Expo Go, що дозволяє швидко переглядати зміни у проєкті без перекомпіляції.

Програмне забезпечення. У процесі розробки було використано сучасний набір інструментів, що забезпечує високу продуктивність і зручність:

– Visual Studio Code (v1.85.1) – головне середовище розробки з підтримкою плагінів, відлагодження, автодоповнення та інтеграції з Git;

– Node.js (v20+) – середовище виконання JavaScript, потрібне для роботи з Expo CLI та залежностями;

– Expo CLI (v6.3.6) – інструмент для керування проєктом, запуску, емуляції та публікації;

– React Native (v0.74+) – фреймворк для кросплатформної розробки мобільних застосунків;

– Expo Go – мобільний застосунок для Android/iOS, що дозволяє миттєво відкривати проєкт зі скануванням QR-коду;

– Git – система керування версіями для відстеження змін у коді;

– GitHub – хостинг-платформа для збереження репозиторію та організації командної роботи;

– Firebase – хмарна платформа для реалізації бекенд-функціоналу (автентифікація, база даних, зберігання, аналітика).

Отже, застосунок BookHive є повноцінною кросплатформною системою, яка поєднує сучасні технології, модульну архітектуру та зручний стек інструментів, що відповідає вимогам до масштабованих та надійних мобільних рішень.

3.3 Опис програмної реалізації

Мобільний застосунок BookHive реалізований з використанням сучасної архітектури на базі фреймворку React Native з підтримкою Expo, що дозволяє забезпечити кросплатформенність і швидке розгортання. Основною перевагою реалізації є модульна структура, яка дозволяє ізольовано

розробляти, тестувати та повторно використовувати компоненти в межах усього застосунку.

Застосунок поділено на кілька логічних рівнів:

– екрани (/app/(tabs)/) – містять основну логіку рендерингу для кожного розділу:index.tsx, library.tsx, book.tsx, reader.tsx, statistics.tsx, challenges.tsx;

– компоненти (/components/) – повторно використовувані елементи інтерфейсу, наприклад: ContinueCard.tsx, RecommendationCard.tsx, ChallengeCard.tsx, IconSymbol.tsx, ProgressBar.tsx;

– константи (/constants/) – зберігають кольорову палітру (colors.ts), стилі, глобальні налаштування;

– сервіси (/services/) (опційно додається) – міститимуть логіку роботи з Firebase, AsyncStorage, API-запитами (наприклад, firebaseService.ts, statsService.ts).

На рисунку 3.1 зображено спрощену файлову структуру проєкту. Поділ структури, який використовується, дозволяє забезпечити чисту архітектуру, що відповідає принципам Separation of Concerns.

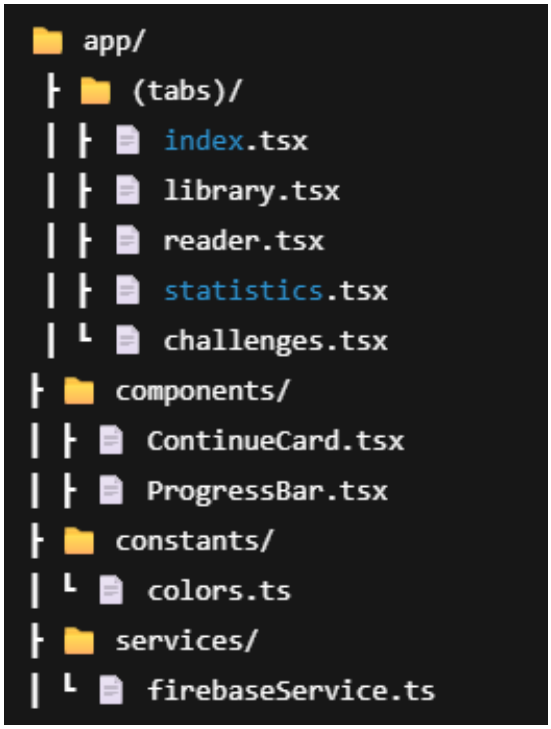

Рисунок 3.1 – Спрощена файлова структура

Навігація між екранами – для реалізації навігації застосовано бібліотеку expo-router, яка базується на файловій структурі. Кожен .tsx-файл у директорії app/(tabs)/ автоматично вважається окремим маршрутом. Навігація між екранами здійснюється через хук (рис. 3.2).

```
import { useRouter } from "expo-router";
const router = useRouter();
router.push("/book");
```

Рисунок 3.2 – Навігація між екранами

Для передачі параметрів між екранами використовується об'єкт params (рис. 3.3).

```
router.push({
  pathname: "/reader",
  params: { fileUri: book.file },
});
```

Рисунок 3.3 – Об’єкт params

Цей підхід спрощує логіку переходів і дозволяє централізовано обробляти навігацію.

3.3.1 Опис основних класів та функціональних модулів

Реалізація BookHive ґрунтується на компонентно-модульному підході. Основна логіка взаємодії користувача з інтерфейсом реалізована через функціональні компоненти, тоді як допоміжні функції та сервіси згруповані в окремі модулі. Така структура забезпечує легкість повторного використання коду та масштабованість проєкту. Основні компоненти інтерфейсу:

- ContinueCard – відображення останньої відкритої книги з прогрес-баром;
- RecommendationCard – візуалізація рекомендованої книги за жанрами;
- ChallengeCard – відображення активного читацького виклику;
- ProgressBar – універсальний індикатор прогресу;

– GenreItem – відображення жанрового тегу;

– IconSymbol – стандаризована обґортка для іконок PhosphorIcons.

Розглянемо деякі функціональні методи та сервіси. Метод getUserStats() (рис. 3.4) – отримує статистику користувача з локального сховища. Вхідні дані: відсутні. Вихідні дані: об’єкт статистики (дні, тривалість, книги).

```
const getUserStats = async () => {
  const data = await AsyncStorage.getItem("readingStats");
  return data ? JSON.parse(data) : { days: [], totalMinutes: 0 };
};
```

Рисунок 3.4 – Метод getUserStats()

Метод addQuote() (рис. 3.5) – додає цитату або нотатку до Firebase/локального кешу. Вхідні дані: рядок із цитатою. Вихідні дані: оновлена база цитат користувача.

```
const addQuote = async (quote: string) => {
  const existing = await AsyncStorage.getItem("quotes");
  const updated = [...(JSON.parse(existing) || []), quote];
  await AsyncStorage.setItem("quotes", JSON.stringify(updated));
};
```

Рисунок 3.5 – Метод addQuote()

Метод fetchRecommendations() (рис. 3.6) – формує список рекомендацій на основі жанрових вподобань. Вхідні дані: масив обраних жанрів. Вихідні дані: масив рекомендованих книг.

```
const fetchRecommendations = (userGenres: string[]) => {
  return booksData.filter((book) =>
    userGenres.some((genre) => book.genres.includes(genre))
  );
};
```

Рисунок 3.6 – Метод fetchRecommendations()

Функція для обчислення прогресу книги (рис. 3.7) – використовується на екрані BookScreen. Вхідні дані: поточна сторінка, загальна кількість сторінок. Вихідні дані: відсоток прогресу.

```
const calculateProgress = (currentPage: number, totalPages: number) => {
  return Math.round((currentPage / totalPages) * 100);
};
```

Рисунок 3.7 – Функція для обчислення прогресу книги

Взаємодія з навігацією (Router Push) (рис. 3.8). Потрібна для переходів між екранами реалізовані централізовано.

```
router.push({
  pathname: "/reader",
  params: { fileUri: book.fileUri },
});
```

Рисунок 3.8 – Router Push

Ці методи згруповано в сервіси (/services/) або використовуються безпосередньо в екранах.

3.3.2 Обробка даних

У застосунку BookHive обробка даних охоплює роботу з локальними файлами, структуроване збереження прогресу та взаємодію з внутрішніми масивами, що моделюють майбутню інтеграцію з базою даних. Основна логіка зосереджена на:

– зчитуванні книг із файлової системи;
– побудові читацької статистики;
– формуванні блоків бібліотеки;
– підрахунку активності користувача.

Зчитування книг з файлової системи (Expo FileSystem) (рис. 3.9) – при відкритті library.tsx програма читає список усіх доступних файлів у папці documentDirectory і фільтрує їх за розширенням.

```
const files = await FileSystem.readDirectoryAsync(FileSystem.documentDirectory);
const books = files.filter((file) =>
  SUPPORTED_EXTENSIONS.some((ext) => file.endsWith(ext))
);
```

Рисунок 3.9 – Сканування директорії

У результаті створюється масив об’єктів типу BookFile (рис. 3.10):

```
{
  title: "Мовчання ягнят",
  author: "Томас Гаррис",
  progress: 0,
  filename: "Мовчання_ягнят.epub"
}
```

Рисунок 3.10 – масив об’єктів типу BookFile

Зчитування вмісту книги (рис. 3.11) – екран reader.tsx використовує метод readAsStringAsync() для відображення тексту книги.

```
const data = await FileSystem.readAsStringAsync(fileUri, {
  encoding: FileSystem.EncodingType.UTF8,
});
setContent(data);
```

Рисунок 3.11 – Зчитування вмісту книги

Формування днів для статистики (рис. 3.12). Для побудови графіку активності за 7 днів застосовується локальна генерація шаблонної структури.

```
const generateDefaultWeek = () => {
  const days = ["Пн", "Вт", "Ср", "Чт", "Пт", "Сб", "Нд"];
  return days.map((day) => ({ day, minutes: 0 }));
};
```

Рисунок 3.12 – Формування днів для статистики

Збереження прочитаного файлу (рис. 3.13). Коли книга відкривається , її назва додається до останніх активностей.

```
const saveLastOpenedBook = async (book: BookFile) => {
  await AsyncStorage.setItem("lastBook", JSON.stringify(book));
};
```

Рисунок 3.13 – Збереження прочитаного файлу

Визначення останньої активності (рис. 3.14) – використовується для головного екрана потрібно відображати картку «Продовжити читання».

```
const getLastOpenedBook = async () => {
  const stored = await AsyncStorage.getItem("lastBook");
  return stored ? JSON.parse(stored) : null;
};
```

Рисунок 3.14 – Визначення останньої активності

Обробка жанрів (рис. 3.15) – жанри зберігаються у вигляді коми-розділеного рядка, який перетворюється на масив.

```
const genreList = genres.split(",").map((g) => g.trim()) || [];
```

Рисунок 3.15 – Обробка жанрів

Таким чином, застосунок обробляє як структуровані JSON-дані (прогрес, активність), так і текстові файли (книги), поєднуючи роботу з файловою системою та локальним сховищем.

3.3.3 Приклад обробки помилок

Під час реалізації застосунку BookHive було враховано типові виняткові ситуації, які можуть виникати під час роботи з файлами, локальним сховищем або мережею. Для цього застосовуються механізми обробки помилок через блоки try...catch, а також запобігання порожнім станам через валідацію й дефолтні значення.

Помилки при зчитуванні книги (рис. 3.16). Коли користувач відкриває файл для читання, він зчитується з файлової системи. Якщо файл пошкоджений або недоступний – показується повідомлення.

```
useEffect(() => {
  const loadFile = async () => {
    try {
      const data = await FileSystem.readAsStringAsync(fileUri, {
        encoding: FileSystem.EncodingType.UTF8,
      });
      setContent(data);
    } catch (error) {
      setContent("Не вдалося відкрити файл.");
      console.error("Помилка зчитування файлу:", error);
    } finally {
      setLoading(false);
    }
  };

  loadFile();
}, [fileUri]);
```

Рисунок 3.16 – Помилки при зчитуванні книги

Відсутність прав доступу на Android (рис. 3.17) – на Android перед читанням з файлової системи потрібен дозвіл. Якщо користувач його не надає – робота блокується з попередженням.

```
if (Platform.OS === "android") {
  const granted = await PermissionsAndroid.request(
    PermissionsAndroid.PERMISSIONS.READ_EXTERNAL_STORAGE
  );
  if (granted !== PermissionsAndroid.RESULTS.GRANTED) {
    console.warn("Доступ до файлів не наданий");
    return;
  }
}
```

Рисунок 3.17 – Відсутність прав доступу на Android

Захист від порожніх відповідей (рис. 3.18). Наприклад, при спробі зчитати останню прочитану книгу або статистику.

```
const getLastOpenedBook = async () => {
  const stored = await AsyncStorage.getItem("lastBook");
  return stored ? JSON.parse(stored) : null;
};
```

Рисунок 3.18 – Захист від порожніх відповідей

Якщо збережених даних немає – повертається null, а інтерфейс відповідно адаптується (наприклад, не показується блок "Продовжити читання").

Застосування дефолтних значень (рис. 3.19) – у багатьох функціях використано логіку за замовчуванням.

```
const defaultStats = {
  days: ["Пн", "Вт", "Ср", "Чт", "Пт", "Сб", "Нд"].map((day) => ({
    day,
    minutes: 0,
  })),
  totalMinutes: 0,
};
```

Рисунок 3.19 – Застосування дефолтних значень

Це гарантує, що навіть у разі відсутності даних користувач побачить коректну, хоч і нульову статистику, а застосунок не впаде.

Вивід у консоль – усі критичні помилки логуються через console.error() або console.warn() для подальшого аналізу під час тестування.

Завдяки використанню асинхронних функцій з обробкою помилок та умовного рендерингу, застосунок BookHive залишається стабільним і зрозумілим у будь-яких ситуаціях.

3.4 Керівництво користувача

У процесі розробки мобільного застосунку BookHive велика увага приділялася етапу тестування. Основна функція його – перевірити

стабільність, функціональність та зручність взаємодії користувача із застосунком, а також визначити можливі проблеми при використанні.

Для перегляду застосунку у тестовому режимі достатньо:

1. Завантажити застосунок Expo Go з Google Play (або App Store).

2. Сканувати QR-код або запустити локальний проєкт через expo start.

3. Застосунок відкриється на мобільному пристрої, дозволяючи взаємодіяти з усіма основними функціями.

Застосунок на платформі Android, з подальшим плануванням розширення на iOS. Було реалізовано ручне тестування ключових функціональних модулів, у тому числі:

– навігації між основними екранами;

– відкриття та зчитування книг у різних форматах;

– фіксації читацького прогресу;

– побудови статистики;

– взаємодії з читацькими викликами;

– обробки помилок і нестандартних ситуацій (відсутність інтернету, пусті дані тощо).

Розглянемо деякі функціональні сценарії тестування:

1. Перегляд бібліотеки та вибір книги (рис. 3.20). Користувач заходить на екран "Бібліотека", де відображаються книги, збережені на пристрої. Перевірено відображення інформації про назву, автора, прогрес читання. При натисканні на книгу відкривається детальний опис.

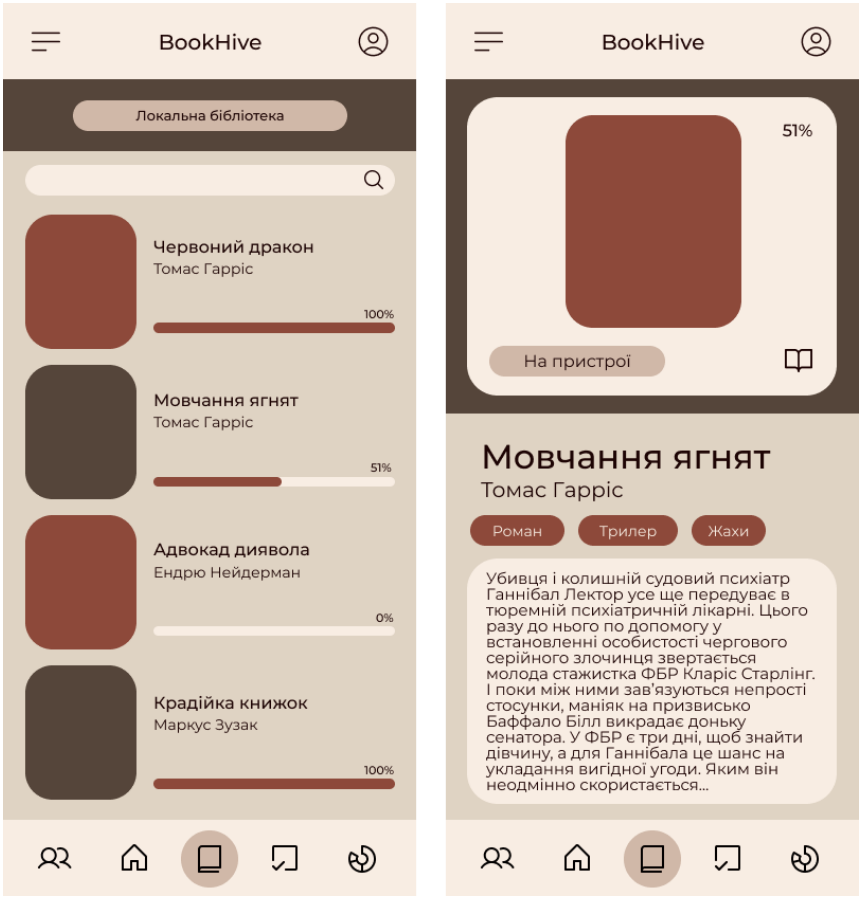

Рисунок 3.20 – Перегляд бібліотеки та вибір книги

2. Відкриття файлу та режим читання (рис. 3.21). Книга відкривається на новому екрані з прокручуваним текстом. Перевірено сумісність форматів .epub, .txt, .fb2, .pdf, (для .pdf та підтримка тестувалась частково, бо вимагає додаткової обробки – планується підтримка в майбутніх оновленнях).

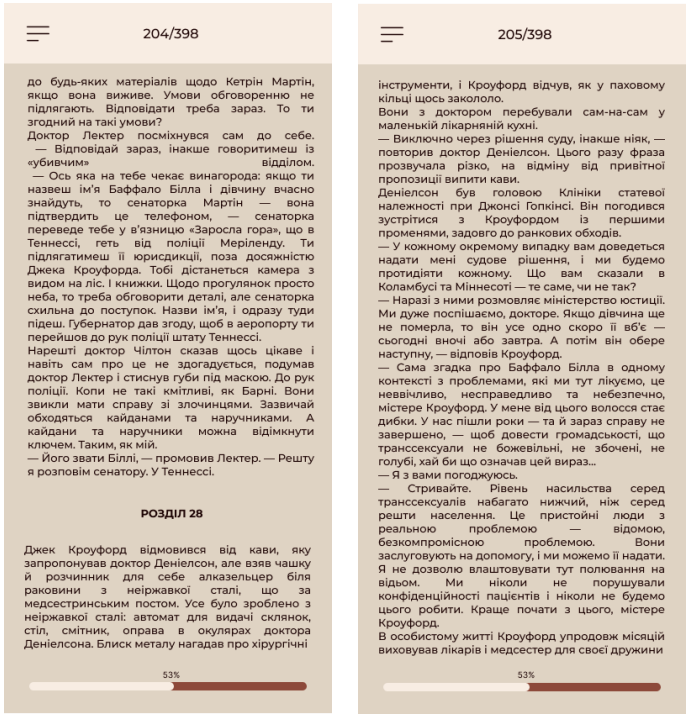

Рисунок 3.21 – Відкриття файлу та режим читання

Також є змога адаптувати інтерфейс читання (рис. 3.22): змінити розмір шрифту, стиль, інтервал між рядками. Для налаштування можна відкрити, якщо натиснути на область біля строки з процентом прочитаного або на неї.

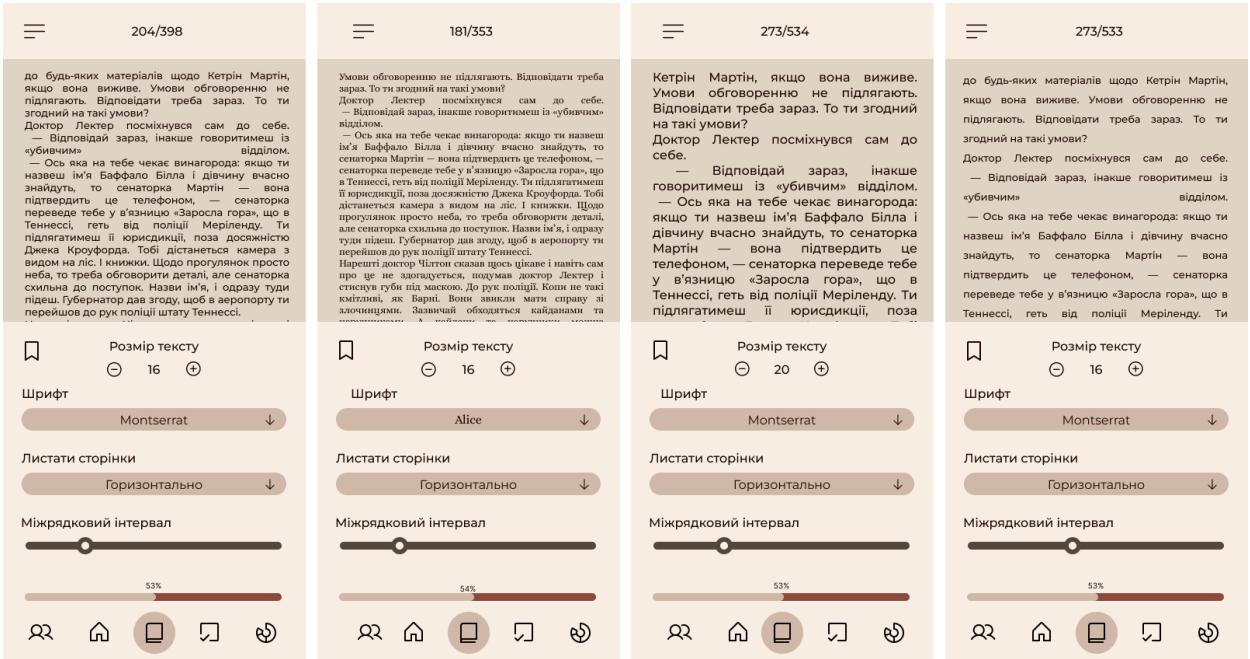

Рисунок 3.22 – Адаптування інтерфейсу читання

3. Збереження прогресу та повернення до читання (рис. 3.23). Після виходу з режиму читання дані про останню відкриту книгу зберігаються локально. При повторному запуску на головному екрані з’являється блок «Продовжити».

4. Відображення читацької статистики (рис. 3.24). На окремому екрані візуалізується кількість прочитаних хвилин за тиждень у вигляді гістограми. Статистика оновлюється при зміні активності користувача.

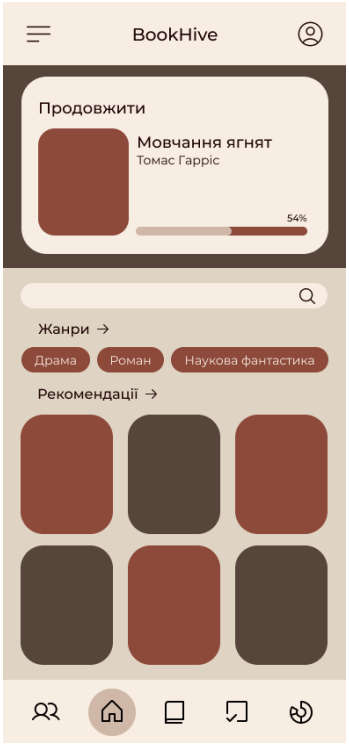

Рисунок 3.23 – Збереження прогресу та повернення до читання

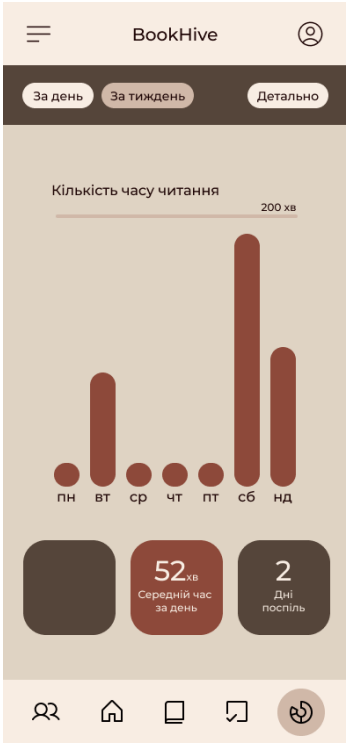

Рисунок 3.24 – Відображення читацької статистики

Під час експлуатації мобільного застосунку користувач може зіткнутися з ситуаціями, які потребують особливої обробки або повідомлення про помилки. У BookHive реалізовано низку захисних механізмів для забезпечення стабільності та зручності взаємодії з інтерфейсом навіть у несприятливих умовах.

Обробка нестандартних ситуацій (рис. 3.25):

– відсутність дозволів – при спробі читання з пам’яті застосунок вимагає дозвіл, якщо його не надано – виводиться попередження;

– відсутність інтернету: застосунок працює офлайн, усі локальні дані доступні;

– помилкові або пусті файли: виводиться повідомлення "Не вдалося відкрити файл";

– порожні списки або відсутність даних – якщо на екрані бібліотеки, статистики чи викликів відсутні дані, реалізовано відображення заглушок з інформативними повідомленнями.

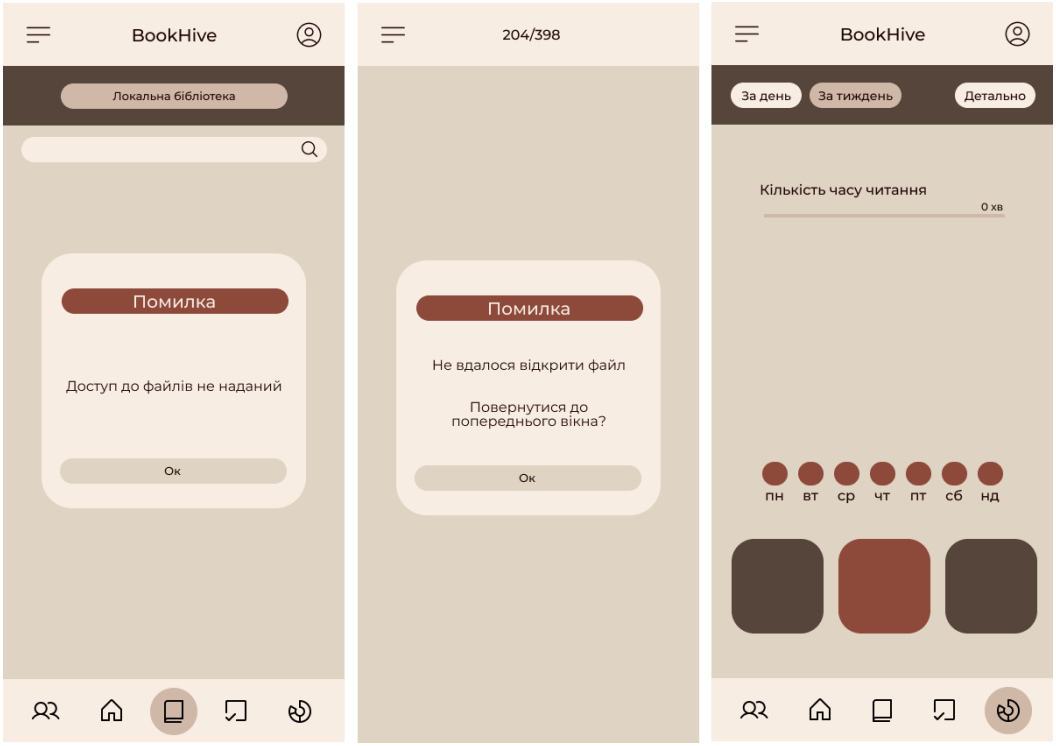

Рисунок 3.25 – Обробка нестандартних ситуацій

Висновки до розділу

У цьому розділі було детально розглянуто процес розробки, реалізації та тестування мобільного застосунку BookHive. Проведено аналіз структури проєкту, описано ключові екрани та компоненти, механізми обробки даних,

реалізацію навігації, а також основні функціональні модулі, що забезпечують персоналізоване читання.

Особливу увагу було приділено створенню інтерфейсу користувача, який поєднує інтуїтивність, привабливий дизайн та функціональність. Запропоновано і реалізовано зручні механізми збереження прогресу, статистичного відстеження, участі у читацьких викликах, а також інтерактивну взаємодію з електронними книгами різних форматів.

Розглянуто питання стосовно тестування, взаємодії користувача із застосунком у різних сценаріях, а також поведінку системи у виключних ситуаціях. Застосунок показав стабільну роботу в офлайн-режимі, підтримку адаптивного інтерфейсу та гнучкість при роботі з різними даними.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ

4.1 Регулювання питань охорони праці на законодавчому рівні

Охорона праці є однією найважливіших складових організації безпечної діяльності працівників у будь-якій галузі, зокрема у сфері інформаційних технологій та розробки програмного забезпечення. Незважаючи на відносну безпеку ІТ-офісів, є ризики професійних захворювань, які пов’язані із тривалим перебуванням за комп’ютером, стресовими навантаженнями, неправильною організацією робочого місця та порушення режиму праці з відпочинком. Тому питання охорони праці регулюються на рівні законодавства та нормативно-правових актів.

Основним документом, який регламентує питання охорони праці в Україні, є Закон України «Про охорону праці» від 14 жовтня 1992 року № 2694-XII. У ньому визначені права та обов’язки роботодавців і працівників, вимоги до умов праці, механізми державного нагляду та відповідальності за порушення. Закон встановлює, що роботодавець зобов’язаний створити умови на кожному робочому місці, що будуть відповідати нормативно-правовим актам, а також нести відповідальність за життя і здоров’я працівників у процесі трудової діяльності [12].

Також важливими документами є:

– Кодекс законів про працю України (КЗпП), зокрема ст. 153, 154, які регламентують безпечні й нешкідливі умови праці;

– Закон України «Про загальнообов’язкове державне соціальне страхування від нещасного випадку» [13];

– Типове положення про службу охорони праці [14];

– Державні санітарні норми та правила роботи за комп’ютером ДСанПіН 3.3.2.007-98, які є актуальними для організації безпечної діяльності працівників сфери розробки ПЗ [15].

Роль роботодавця у системі охорони праці полягає у створенні безпечних умов праці, забезпечення працівників інструктажами, засобами індивідуального захисту (якщо це потрібно), проведенні атестацій робочих місць, здійсненні контролю за дотриманням норм. Окрім того, роботодавець зобов’язаний організовувати навчання з питань охорони праці відповідно до НПАОП 0.00-4.12-05 [16].

Також в організації важлива діяльність фахівця з охорони праці, який здійснює контроль за виконанням законодавчих вимог, веде документацію, проводить інструктажі, слідкує за станом умов та проводить профілактичні заходи. У малих ІТ-компаніях, де відсутня окрема служба охорони праці, ці функції можуть виконувати особи, які відповідальні за адміністративні напрями.

Кожен працівник, зі свого боку, зобов’язаний дотримуватись правил техніки безпеки, інструкцій з охорони праці, проходити навчання та своєчасно повідомляти про будь-які загрози або нещасні випадки на робочому місці. Таким чинок, охорона праці є спільною відповідальністю всіх сторін трудового процесу.

До того ж, на державному рівні функціонують органи, що здійснюють нагляд та контроль у сфері охорони праці, зокрема Державна служба України з питань праці (Держпраці), яка проводить перевірки, розробляє нормативи та контролює виконання законодавства.

На останок, слід зазначити, що впровадження ефективної системи охорони праці має не лише юридичне, але й соціально-економічне значення. Вона сприяє підвищенню продуктивності праці, зниженню захворювань, покращенню мікроклімату в колективі та зменшення витрат, пов’язаних із наслідком порушень.

4.2 Виявлення потенційних небезпек стосовно об’єкту проєктування

Об’єктом проєктування у даній роботі є програмне забезпечення – мобільний застосунок BookHive, а саме – процес його розробки в умовах ІТ-компанії. Як середовище реалізації проєкту розглядається стандартне офісне приміщення, де ведеться інтелектуальна праця за допомогою комп’ютерної техніки. Незважаючи на те, що діяльність цій сфері не пов’язана з високим рівнем виробничих травм, вона супроводжується низкою потенційно небезпечних та шкідливих факторів, що можуть впливати на фізичне та психоемоційне здоров’я працівників.

Нижче наведено таблицю 4.1 з переліком небезпечних та шкідливих виробничих факторів, поділених за їхньою природою дії:

Таблиця 4.1 – Потенційні небезпечні та шкідливі фактори для працівників ІТ-сфери

Тип фактору	Потенційна небезпека/шкідливий фактор
Фізичні	– напруження зору при тривалій роботі за монітором; – недостатня або надмірна освітленість; – шум від техніки (вентилятори, системні блоки) – може призводити до дратівливості; – незручна ергономіка робочого місця (незручне крісло, стіл, відсутність підставки під руки), що може викликати захворювання опорно-рухового апарату, зокрема шийно-грудного відділу хребта; – погана вентиляція, нестача кисню; – перегрів або пересушене повітря в приміщені.
Хімічні	– пил від оргтехніки (захворювання дихальних шляхів); – випаровування з матеріалів меблів; – хімічні мийні засоби (дратівливість, алергії).
Психофізіологічні	– статичні фізичні навантаження (довге сидіння); – розумове перенапруження; – монотонність праці; – емоційні перевантаження; – перенапруження аналізаторів (зір, моторика).
Біологічні	– вірусні інфекції у колективі, особливо в холодну пору року.
Загальні небезпеки	– пожежна небезпека (електропристрої, коротке замикання); – воєнна загроза (обстріли, евакуація, відключення електрики); – електротравматизм при порушенні правил безпеки; – ризики кібератак та витоку інформації (як ризик інформаційної безпеки).

Як бачимо, діяльність у сфері розробки програмного забезпечення супроводжується не лише фізичними навантаженнями, а й психоемоційними навантаженнями, пов’язаними з дедлайнами, високим інформаційним тиском і потребою в постійні концентрації. Крім того, зовнішні загрози – такі як воєнна небезпека чи пожежна – також мають бути враховані при організації офісного середовища. Усі виявлені фактори можуть негативно впливати на стан здоров’я, продуктивність праці та безпеку персоналу.

Зменшення впливу вказаних шкідливих і небезпечних факторів можливе за рахунок:

– дотримання санітарно-гігієнічних норм;

– використання ергономічних меблів;

– впровадження регулярних перерв у роботі (наприклад, за принципом «20-20-20» для очей [17]);

– забезпечення належної вентиляції, освітлення;

– проходження медоглядів і навчання з охорони праці;

– встановлення та регулярне обслуговування засобів пожежогасіння;

– облаштування укриттів та систем сповіщення у разі загрози.

Таким чином, попри відсутність фізичної небезпеки у традиційному розумінні, робота в ІТ-сфері передбачає наявність різнопланових ризиків, вплив яких потрібно мінімізувати на етапі проєктування організації праці.

4.3 Дослідження ризику реалізації небезпек на об’єкті проєктування та розробка заходів щодо їх попередження

Процедура оцінювання ризиків є ключовим інструментом у сфері охорони праці, оскільки дозволяє виявити та оцінити ступінь небезпеки потенційних загроз, пов’язаних з умовами праці або робочим середовищем. Вона включає послідовне визначення можливих небажаних подій, оцінювання

серйозності, ймовірність виникнення та розробку для зменшення ризиків або їх усунення.

Основна мета оцінювання ризику полягає в зниженні загроз для життя, здоров'я та працездатності працівників, а також у запобіганні аваріям і надзвичайним ситуаціям. Згідно із Законом Україна «Про охорону праці» роботодавець зобов'язаний забезпечити системний підхід до виявлення і мінімізації ризиків на робочих місцях, у тому числі шляхом проведення оцінки небезпечних і шкідливих факторів виробничого середовища [12].

Серед основних завдань оцінки ризиків є:

- виявлення всіх потенційних небезпек на робочому місці;
- визначення причин та умов, що сприяють їх реалізації;
- оцінка можливих наслідків;
- класифікація ризиків за рівнем серйозності та ймовірності;
- розробка заходів безпеки.

У межах розробки мобільного застосунку для ІТ-сфери оцінку ризику доцільно проводити для середовища типового офісу. Найбільш поширені загрози тут пов'язані з електробезпекою, зоровим та психофізичним навантаженням, низькою ергономікою робочого місця.

Для аналізу ризику реалізації потенційних загроз на об'єкті розробки було обрано метод «дерева відмов» - один із найбільш поширених у практиці системного аналізу безпеки. Цей метод передбачає побудову логічної структури, в якій зазначаються усі можливі причини виникнення небажаної події та встановлюються між ними логічні зв'язки за допомогою операцій «Та» і «Або» [19].

Розглянемо на прикладі наступний ризик – професійне вигорання розробника (рис. 4.1), як наслідок надмірного стресового навантаження, відсутності відпочинку, низької мотивації та поганої організації процесів. Цей тип ризику є особливо актуальним для ІТ-сфери, де поширеним є високий темп виконання завдань, постійні дедлайни, багатогодинне сидіння за монітором, а також віддалений або гнучкий графік, що призводить до розмиття межі між

роботою і особистим життям. Наслідки вигорання не лише знижують ефективність працівників, але й можуть спричинити тривалі психоемоційні розлади, втрату професійної мотивації та потребу в тривалому відновлені. Саме тому виявлення та профілактика цього ризику є критично важливою частиною загальної системи охорони праці ц сфері розробки програмного забезпечення.

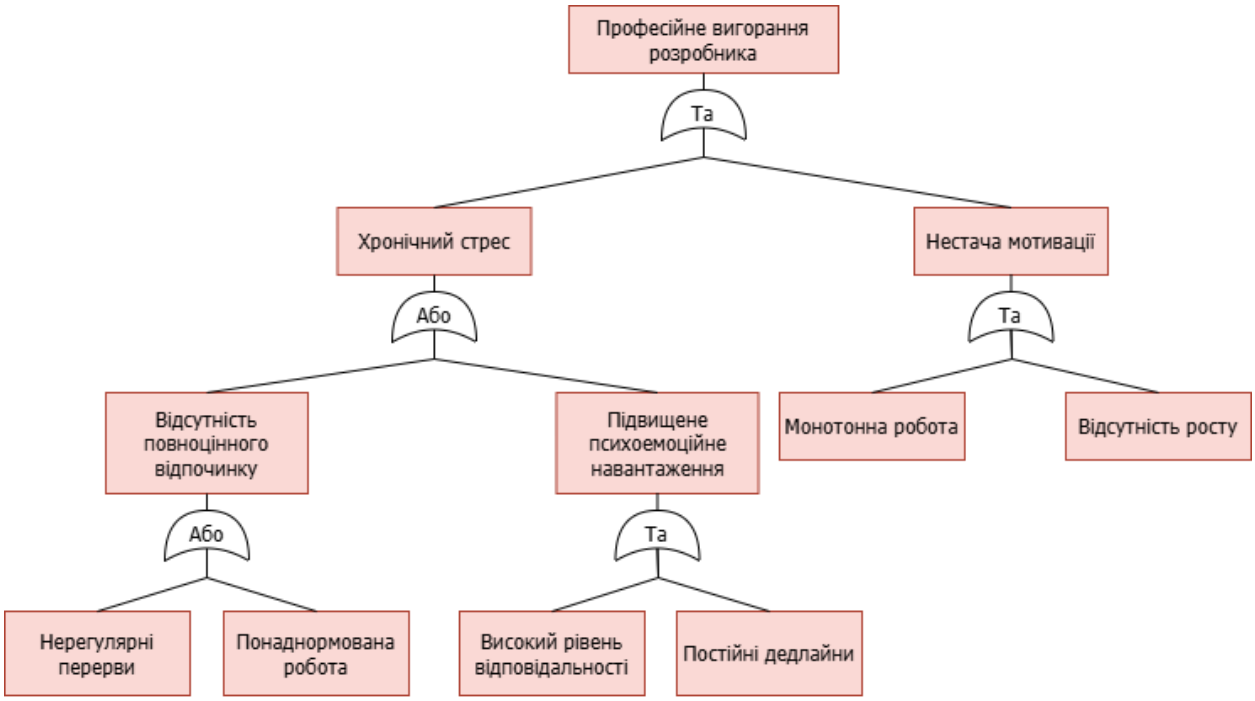

Рисунок 4.1 – «Дерево відмов» для ситуації «Професійне вигорання розробника»

Додатково було застосовано метод матриці оцінювання ризиків (табл. 4.2), що дозволяє класифікувати й оцінити рівень ризику. Це відбувається за відповідною шкалою, згідно з якою можна визначити індекс ризику: від 1A (найвищий) до 4E (найнижчий) – відповідно до категорії небезпеки та ймовірності її настання [18].

Таблиця 4.2 – Оцінювання ризиків методом матриці

Назва небезпеки	Категорія серйозності	Ймовірність виникнення	Індекс ризику	Рівень ризику
1	2	3	4	5
Електротравматизм через несправне обладнання	II – Критична (можливі травми, опіки, госпіталізація)	C – Випадкова (можлива при відсутності заходів безпеки	2C	Неприпустимий (високий)
Психофізіологічне перевантаження	IV – Незначна (втома, зниження ефективності)	B – Можлива (у режимі реальної роботи)	4B	Прийнятний (низький)
Перевантаження зорового апарату	III – Помірна (погіршення зору, сухість очей)	B – Можлива	3B	Гранично допустимий
Пожежа в офісі	I – Катастрофічна (потенційна загибель, знищення майна)	D – Віддалена (малоймовірна, але можлива)	1D	Небажаний (неприпустимий)
Втрата даних унаслідок збою техніки	II – Критична (збитки, зупинка роботи)	C – Випадкова	2C	Неприпустимий

На основі аналізу було сформульовано практичні заходи з профілактики основних виявлених ризиків:

1. Електротравматизм. Необхідно регулярно перевіряти техніку, забезпечити заземлення, маркувати небезпечні зони та проводити інструктажі з охорони праці.

2. Психофізіологічне навантаження. Слід дотримуватися чіткого графіку роботи, впровадити регулярні перерви, оптимізувати розподіл завдань, підтримувати здорову атмосферу у команді.

3. Зорові перенавантаження. Рекомендується налаштування ергономічного робочого місця, використання фільтрів синього світла, забезпечення достатнього освітлення та регулярні вправи для очей.

4. Пожежна безпека. Важливо мати діючі вогнегасники, пожежну сигналізацію, схеми евакуації та чіткі інструкції на випадок надзвичайної ситуації.

5. Збереження даних. Потрібно забезпечити регулярне резервне копіювання та використання джерел безперебійного живлення для критичного обладнання

Здійснення цих заходів сприятиме зниженню виробничих ризиків і покращенню умов праці у сфері ІТ.

Висновки до розділу

Охорона праці в сфері розробки програмного забезпечення є важливою складовою забезпечення безпечного та ефективного робочого процесу. Незважаючи на відсутність фізичної небезпеки, характерної для виробництва, працівники ІТ-сфери щоденно стикаються з ризиками, пов’язаними з тривалою розумовою напругою, статичним навантаженням, напруженням зору, низькою фізичною активністю та психоемоційними перенавантаженнями.

Було виявлено, що робоче середовище офісного типу має низку потенційно шкідливих і небезпечних факторів: фізичних, хімічних, психофізіологічних та загальних. Серед найпоширеніших – недостатня ергономіка, надмірне навантаження на зір, погана вентиляція, стресові ситуації та ризик професійного вигорання. Окрему увагу привертають на себе зовнішні загрози: пожежна безпека, кібератаки та ризики, які пов’язані з воєнною ситуацією.

Проведено оцінювання ризиків методом матриці, що дозволило класифікувати ймовірні загрози за рівнем небезпеки та визначити пріоритетність заходів. Також було проведено аналіз ризику професійного вигорання за допомогою методу «дерева відмов», який продемонстрував багатофакторну природу та важливість своєчасної профілактики. На основі цих аналізів сформульовано практичні рекомендації для підвищення рівня безпеки: дотримання режиму праці та відпочинку, організація ергономічного робочого місця, впровадження психоемоційної підтримки, навчання

персоналу правилам безпечної експлуатації техніки, пожежна та інформаційна безпека, регулярне резервне копіювання даних та використання джерел безперебійного живлення.

Застосування цих заходів сприятиме зниженню ризиків, підвищенню продуктивності та створенню здорового робочого середовища для фахівців у галузі розробки програмного забезпечення.

ВИСНОВКИ

У межах роботи було досягнуто поставлену мету – створено мобільний застосунок BookHive для персоналізованого читання електронних книг. Застосунок надає користувачам можливість не лише читати файли різних форматів, а й зберігати читацький прогрес, брати участь у викликах, переглядати статистику та взаємодіяти з іншими користувачами. Робота охопила повний цикл проєктування інформаційної системи – від аналізу предметної області до реалізації програмного продукту та тестування його функціоналу.

У розділі 1 «Загальні положення» було проведено аналіз сучасного стану ринку мобільних застосунків для читання книг. Визначено основні недоліки існуючих рішень. Сформульовано мету, завдання, об’єкт і предмет дослідження. Було обґрунтовано доцільність створення застосунку, який би поєднував зазначені функції в єдиному середовищі.

У розділі 2 «Інформаційне та математичне забезпечення» було описано моделі та підходи, використані при розробці BookHive. Зокрема, розглянуто структуру зберігання даних, підхід до реалізації рекомендацій на основі активності користувачів, а також принципи відображення статистики. Описано алгоритмічні елементи, пов’язані з розрахунком прогресу, генерацією рекомендацій та створенням читацьких викликів.

У розділі 3 «Програмне та технічне забезпечення» наведено опис архітектури застосунку та ключових модулів. Створено багаторівневу структуру з розмежуванням інтерфейсу, логіки та взаємодії з Firebase. Реалізовано екрани домашньої сторінки, бібліотеки, читання, статистики, рекомендацій та викликів. Окрему увагу приділено створенню компонентів, які можна багаторазово використовувати. Також подано приклади коду та фрагменти логіки обробки помилок, збереження даних та асинхронної

взаємодії з сервером. Проведено тестування на реальному пристрої, а також розроблено базове керівництво для кінцевого користувача.

У розділі 4 «Охорона праці та безпека в інформаційній діяльності» розглянуто умови безпечної праці під час розробки ПЗ, рекомендовані параметри організації робочого місця, вимоги до психофізіологічної безпеки та заходи щодо мінімізації ризиків під час тривалої взаємодії з мобільними пристроями. Також розглянуто аспекти інформаційної безпеки при роботі із зовнішніми хмарними сервісами.

У результаті проведеної роботи:

– досягнуто сформульованої мети – створено сучасний функціональний застосунок для читання;

– виконано поставлені завдання: від аналізу предметної області до реалізації та тестування;

– розроблено технічно обґрунтовану архітектуру та реалізовано функціональне програмне забезпечення;

– забезпечено зручність для користувачів завдяки продуманому UX/UI-дизайну;

– отримано результат, який має практичне застосування та потенціал для подальшого розвитку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Полторак В.В., Бочаров Б.П. Актуальність та перспективи розробки розумних додатків для читання та обліку книг // Інформаційні технології: теорія і практика: Тези VIII (II) Міжнародної Інтернет-конференції здобувачів вищої освіти і молодих учених (Запоріжжя-Харків-Дніпро, 2-4 квітня 2025 р.), [Електронний ресурс] Електрон. дані. – Запоріжжя : НУ «Запорізька політехніка», 2025.– 320 с. – [Електронний ресурс]. – Режим доступу: https://eir.zp.edu.ua/server/api/core/bitstreams/161e939a-eb51-4619-82f2-f2d2708c02cf/content . – Назва з екрана.

2. Statista. (2023). *eBooks – Worldwide*. [Електронний ресурс]. – Режим доступу: https://www.statista.com/outlook/amo/media/books/ebooks/worldwide#analyst-opinion – Назва з екрану.

3. Doiron, R. (2011). Using E-Books and E-Readers to Promote Reading in School Libraries: Lessons from the Field. *Teacher Librarian*.

4. Rork – персоналізований застосунок для рекомендацій книг [Електронний ресурс]. – Режим доступу: https://rork.io. – Назва з екрану.

5. Moon+ Reader – читання електронних книг [Електронний ресурс]. – Режим доступу: https://play.google.com/store/apps/details?id=com.flyersoft.moonreader. – Назва з екрану.

6. Goodreads — офіційний сайт [Електронний ресурс]. – Режим доступу: https://www.goodreads.com – Назва з екрану.

7. Recker J., Lukyanenko R., Castellanos A. From Representation to Mediation: A New Agenda for Conceptual Modeling in a Digital World. MIS Quarterly, March 2021.

8. What is an Information System? [Електронний ресурс] / TechTarget. – Режим доступу: https://www.techtarget.com/whatis/definition/IS-information-system-or-information-services?utm_source=chatgpt.com – Назва з екрана.

9. Object Oriented Programming – Introduction [Електронний ресурс] / GeeksforGeeks. – Режим доступу: https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/ – Назва з екрана.

10. TechTarget. Object-Oriented Programming (OOP): Definition and Benefits [Електронний ресурс]. – 2024. – Режим доступу: https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP . – Назва з екрана.

11. Ali H. M., Hamza M. Y., Rashid T. A. Exploring Polymorphism: Flexibility and Code Reusability in OOP // Passer Journal of Basic & Applied Sciences. – 2023. – Vol. 6, No. 2. – Available: https://www.researchgate.net/publication/380927123.

12. Закон України «Про охорону праці». – № 2694-XII від 14.10.1992. [Електронний ресурс] / Офіційний сайт Верховної Ради України. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2694-12. – Назва з екрану.

13. Кодекс законів про працю України [Електронний ресурс] / Офіційний сайт Верховної Ради України. – Режим доступу: https://zakon.rada.gov.ua/laws/show/322-08. – Назва з екрану.

14. Типове положення про службу охорони праці : НПАОП 0.00-4.21-04 [Електронний ресурс] / Офіційний сайт Верховної Ради України. – Режим доступу: https://zakon.rada.gov.ua/laws/show/z1526-04#Text. – Назва з екрану.

15. Державні санітарні правила і норми ДСанПіН 3.3.2.007-98 «Гігієнічні вимоги до відеодисплейних терміналів, ПК та організації роботи» [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/z0203-98. – Назва з екрану.

16. НПАОП 0.00-4.12-05. Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/z0231-05. – Назва з екрану.

17. Chou B. Deconstructing the 20-20-20 Rule for digital eye strain [Електронний ресурс] / Optometry Times. – Режим доступу: https://www.optometrytimes.com/view/deconstructing-20-20-20-rule-digital-eye-strain. – Назва з екрану.

18. Левченко О.Г., Землянська О.В., Праховнік Н.А., Зацарний В.В. «Безпека життєдіяльності та цивільний захист: підручник». – Київ: Каравела, 2019. – 267 с.

19. Chovancová, J., Fidlerová, H., & Kráľová, P. – The Importance of Risk Assessment in Occupational Health and Safety Management. // Sustainability. – 2022. – Vol. 14, No. 18. – Article №11430. – [Електронний ресурс]. – Режим доступу: https://www.mdpi.com/2071-1050/14/18/11430 – Назва з екрана.

20. React Native [Електронний ресурс]. – Режим доступу: https://reactnative.dev/ – Назва з екрана.

21. Expo – Documentation [Електронний ресурс]. – Режим доступу: https://expo.dev/ – Назва з екрана.

22. Firebase – Build and run successful apps [Електронний ресурс]. – Режим доступу: https://firebase.google.com/ – Назва з екрана.

Додаток А
Апробація результатів роботи

Міністерство освіти і науки України
Національний університет
«Запорізька політехніка»
Національний технічний університет
«Дніпровська політехніка»
Харківський національний університет
міського господарства імені О.М. Бекетова
ГО «Системні дослідження»
ГО МДЦВЕ
Esslingen University of Applied Sciences
University of Koblenz
Cardiff University
Kırklareli University
Universidad Politécnica de Madrid

**ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ:
ТЕОРІЯ І ПРАКТИКА**

**ІІ (VIII) Міжнародна Інтернет-конференція
здобувачів вищої освіти і молодих учених**

2-4 квітня 2025 р

Тези доповідей

Запоріжжя-Харків-Дніпро

Україна

УДК 004.51/.92

Полторак В.В.[1], Бочаров Б.П. [2]

АКТУАЛЬНІСТЬ ТА ПЕРСПЕКТИВИ РОЗРОБКИ РОЗУМНИХ ДОДАТКІВ ДЛЯ ЧИТАННЯ ТА ОБЛІКУ КНИГ

Вступ. Цифрове читання стає все більш важливим для різних жанрів і типів тексту. Читання новин на цифрових носіях стало звичним явищем за останні два десятиліття, а споживання електронних книг може залишатися постійним явищем [1].

Зростання цифрових технологій сприяє популяризації електронного читання та персоналізованих книжкових сервісів. Сучасні користувачі стикаються з проблемою організації власної бібліотеки, запам'ятовування списку бажаних книг та отримання релевантних рекомендацій. Інтелектуальні книжкові платформи дозволяють автоматизувати процеси відстеження читання, створення рекомендацій та взаємодії між користувачами, що значно підвищує ефективність роботи з літературними джерелами.

Актуальність проблеми. Електронні книги – це електронні версії книг, які можна читати на мобільних пристроях із вбудованими функціями, такими як вбудовані словники, інструменти пошуку слів і оцифрована аудіорозповідь.[2] На ринку електронних книг важливою тенденцією є все більше використання інтерактивного та мультимедійного вмісту, оскільки видавці експериментують із розширеними форматами, які включають аудіо, відео та інтерактивні елементи, щоб глибше зацікавити читачів. Крім того, розвиток штучного інтелекту впливає на персоналізовані рекомендації, підлаштовуючи контент відповідно до індивідуальних уподобань і читацьких звичок [3].

Але на сьогодні більшість книжкових сервісів зосереджені на продажу літератури, а не забезпечують гнучких інструментів для організації читання. Таким чином, розвиток технологій, що дозволяють каталогізувати книги, відстежувати прогрес читання та отримувати індивідуальні рекомендації, є актуальним та перспективним напрямом.

Мета дослідження. Метою дослідження є аналіз можливостей та перспектив розвитку інтелектуальних систем для читання та обліку книг, які можуть забезпечити зручне управління книжковими списками, автоматизоване відстеження прогресу читання, персоналізовані рекомендації та інтеграцію з іншими користувачами.

[1] студентка групи КН 2021-1, ХНУМГ ім. О. М. Бекетова
[2] доцент кафедри Комп'ютерних наук та інформаційних технологій, ХНУМГ ім. О. М. Бекетова, к.т.н.