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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Створення мобільного додатку для обміну книгами «BookSharing» з використанням фреймворку Flutter

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Бережник Ю.Ю.
(підпис) (прізвище та ініціали)

Керівник _____ Фриз М.Є.
(підпис) (прізвище та ініціали)

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

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

Рецензент _____ .
(підпис) (прізвище та ініціали)

Тернопіль
2023

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

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

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

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

Студенту Бережнику Юрію Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Створення мобільного додатку для обміну книгами «BookSharing» з використанням фреймворку Flutter

Керівник роботи Фриз Михайло Євгенович, кандидат технічних наук, доцент кафедри комп'ютерних наук
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 7 » лютого 2023 року № 4/7 -133

2. Термін подання студентом завершеної роботи 2023р.

3. Вихідні дані до роботи Літературні та інтернет джерела про фреймворк Flutter, про мову програмування Dart та про платформу Firebase.

4. Зміст роботи РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ BOOKSHARING. 1.1 Опис предметної області. 1.2 Огляд існуючих альтернатив. 1.2.1 – BooksAround. 1.2.2 – BookSwap. 1.2.3 – BookCrossing. 1.3 Формування вимог до мобільного додатку. 1.4 Побудова діаграм варіантів використання додатку. 1.5 Прототип додатку. РОЗДІЛ 2 РОЗРОБКА ТА ВИКОРИСТАННЯ МОБІЛЬНОГО ДОДАТКУ BOOKSHARING. 2.1 Обґрунтування вибору засобів розробки мобільного додатку. 2.1.1 Аналіз переваг фреймворку Flutter. 2.1.2 Структура та можливості платформи Firebase. 2.2 Створення додатку. 2.2.1 Структура проекту. 2.2.2 Підключення проекту до Firebase. 2.2.3 Реалізація надсилання сповіщень. 2.3 Реліз мобільного додатку BookSharing. 2.4 Демонстрація роботи додатку BookShairing. РОЗДІЛ 3 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ,. ОСНОВИ ОХОРОНИ ПРАЦІ. 3.1 Діяльність її види та розуміння в безпеці праці. 3.1.1 Виробнича діяльність. 3.1.2 Будівельна діяльність та її розуміння в безпеці праці.. 3.1.3 Промислова діяльність. 3.2 Загальні вимоги безпеки з охорони праці для користувачів ПК. ВИСНОВКИ. ПЕРЕЛІК ДЖЕРЕЛ. ДОДАТКИ. Додаток А. Додаток Б.

5. Перелік графічного матеріалу 1. Титульний слайд. 2.Актуальність та мета. 3.Огляд Існуючи альтернатив. 4.Формування вимог. 5.Варіанти використання додатку. 6. Прототипи додатку. 7.Переваги Flutter. 8. Порівняння фреймворків. 9. Ознайомлення з Firebase. 10. Зв'язки між таблицями. 11. Трьохрівнева архітектура. 12. Демонстрація додатку. 13. Посилання на додаток. 14. Подяка.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці		05.06.2023	08.06.2023

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	23.01.2023	*Виконано*
2.	Підбір джерел про фреймворки мобільної розробки	24.01.2023-26.01.2023	*Виконано*
3.	Опрацювання джерел по темі кваліфікаційної роботи	27.01.2023-31.01.2023	*Виконано*
4.	Виконання дослідження щодо взаємодії Flutter та Firebase	01.02.2023-07.02.2023	*Виконано*
5.	Оформлення розділу «Аналіз предметної області та Проектування мобільного додатку BookShairing»	08.02.2023-09.02.2023	*Виконано*
6.	Оформлення розділу « Розробка та використання мобільного додатку BookShairing	10.02.2023-12.02.2023	*Виконано*
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	05.06.2023-06.06.2023	*Виконано*
8.	Виконання завдання до підрозділу «Основи хорони	07.06.2023-08.06.2023	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	09.06.2023-11.06.2023	*Виконано*
10.	Нормоконтроль	12.06.2023-13.06.2023	*Виконано*
11.	Перевірка на плагіат	14.06.2023	*Виконано*
12.	Попередній захист кваліфікаційної роботи	15.06.2023	*Виконано*
13.	Захист кваліфікаційної роботи	19.06.2023	

Студент _____ Бережник Ю.Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Фриз М. Є.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Створення мобільного додатку для обміну книгами «BookSharing» з використанням фреймворку Flutter // Кваліфікаційна робота освітнього рівня «Бакалавр» // Бережник Юрій Юрійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2023 // С. -58, рис. – 25, табл. – 6, додат. – 2 , бібліогр. – 26.

Ключові слова: flutter, dart, firebase, android, google, api, widget.

Кваліфікаційна робота присвячена створенню мобільного додатку для обміну книгами «BookShairing».

Мета роботи полягає в створенні мобільного додатку з використанням фреймворку Flutter та платформи Firebase для реалізації додаткових можливостей.

В першому розділі кваліфікаційної роботи проведено аналіз вже існуючих альтернатив, сформовано вимоги до мобільного додатку, побудовано діаграми використання додатку, створено прототип мобільного додатку.

В другому розділі кваліфікаційної роботи обґрунтовано вибір засобів розробки мобільного додатку, проведено аналіз переваг фреймворку Flutter, показано можливості платформи Firebase, продемонстровано фрагменти з розробки мобільного додатку, наведено інструкції для коректного релізу додатку, продемонстровано роботу мобільного додатку.

В третьому розділі кваліфікаційної роботи розглянуто питання з безпеки життєдіяльності, основи охорони праці: діяльність, її види та розуміння в безпеці праці, загальні вимоги х охорони праці для користувачів ПК.

ANNOTAITION

Development of Mobile Application "BookSharing" Using Flutter Framework // Qualification work of the educational level «Bachelor» // Berezhnyk Yurii Yuriiovych // Ternopil Ivan Pului National Technical University, faculty of computer information systems and software engineering, cathedra of computer sciences, group SN-41 // Ternopil, 2023 // p. – 58, img. – 25, spreadsheets – 6, appendix – 2, references – 26.

Key words: flutter, dart, firebase, android, google, api, widget.

The qualification work is dedicated to the development of a mobile application called "BookShairing."

The objective of the work is to create a mobile application using the Flutter framework and the Firebase platform to implement additional features.

The first chapter of the qualification work includes an analysis of existing alternatives, the formulation of requirements for the mobile application, the construction of application use case diagrams, and the creation of a mobile application prototype.

In the second chapter of the qualification work, the choice of mobile application development tools is justified, the advantages of the Flutter framework are analyzed, the capabilities of the Firebase platform are demonstrated, development fragments of the mobile application are shown, instructions for a proper application release are provided, and the functionality of the mobile application is demonstrated.

The third chapter of the qualification work discusses issues related to life safety and the basics of occupational safety: the nature of activities, their types, and understanding of safety at work, as well as general requirements for occupational safety for PC users.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Android – Операційна система для мобільних телефонів створена компанію Google.

CRUD – Create, Read, Update, Delete.

Dart – Мова програмування.

Firebase – Платформа розробки мобільних та веб додатків.

Hi-Fi – High fidelity prototype.

Ios - Операційна система для мобільних телефонів створена компанію Apple.

Lo -Fi – Low fidelity prototype.

ПК- Персональний комп’ютер.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі, технології використовуються майже у всіх аспектах людського життя, мобільні додатки стали одним з головних джерел для обміну та доступу до інформації. Ідея обміну книгами через мобільний додаток полягає в заохоченні обміну знаннями та сприянні культури читання.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є створення мобільного додатку за допомогою якого користувачі зможуть отримувати доступ до нових знань, без необхідності постійно купувати нові книги. За допомогою BookSharing користувачі зможуть завантажувати свої колекції книг, переглядати доступні книги інших користувачів, та пропонувати їм обмін.

Практичне значення одержаних результатів. Практичне значення кваліфікаційної роботи полягає в отриманні готового мобільного додатку для обміну книгами.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ BOOKSHARING

1.1 Опис предметної області

Метою кваліфікаційної роботи є створення мобільного додатку з використанням фреймворку Flutter [1], який буде призначений задля об’єднання людей які хочуть ділитися книгами один з одним.

Обмін книгами вже довгий час є звичайною практикою серед книголюбів, це надає їм доступ до широкого спектру літератури, а також розширює коло знайомств.

Додаток надасть користувачам простий і зрозумілий інтерфейс задля полегшення обміну книгами. Він дозволить користувачам створювати особисті профілі, створювати власні бібліотеки, шукати книги, надсилати запити на отримання книг, вести бесіду з іншим користувачем, та відстежувати хід обміну книгами.

Створення додатку «BookSharing» передбачатиме використання фреймворку Flutter, популярної платформи для розробки мобільних додатків з відкритим кодом. Flutter використовує єдину кодову базу для створення кросплатформерних додатків. Використання цього фреймворку дозволить створити швидкий, високоякісний додаток з привабливим та адаптивним інтерфейсом, який зможе залучити широке коло користувачів.

1.2 Огляд існуючих альтернатив

Було проведено аналіз вже існуючих сервісів зі схожим або частково схожим функціоналом. Існуючі додатки було знайдено в офіційних магазинах Google Play Market та AppStore, а також на просторах інтернету. Для огляду було обрано додатки з такими функціями як: ведення власної бібліотеки та обмін книгами.

Для порівняння було обрано найбільш відомі додатки, які підходять під вище наведені критерії. В наступних підрозділах буде наведено короткий опис кожного додатку, їх переваги та недоліки..

1.2.1 – BooksAround

BooksAround [2] – це сучасний мобільний додаток, створений українським розробником Олександром Резвінським. Як зазначає сам автор [3], робота над цим проектом почалась ще в далекому 2017 році. Після невдалих спроб, він поновив роботу в 2019 році, переписавши код з нуля. І в тому ж році була випущена перша версія під iOS, а в наступному році під Android.

BooksAround містить такий функціонал:

- Створення власної бібліотеки;
- Пошук книг за назвою чи автором;
- Функція геолокації, задля пошуку найближчих користувачів;
- Месенджер для організації обміну книгами;
- Створення списку бажаних книг;
- Інтерфейс різними трьома мовами;

Переваги:

- Простий та сучасний інтерфейс(Див. рисунок 1.1);
- Мобільний додаток є як на Android так і на iOS;
- Підтримка додатку. Регулярно виходять оновлення які його покращують;
- Підтримка цікавості. На офіційних сторінках додатку в соц. мережах ведуться блоги, в яких розповідається про книги;

Недоліки:

- Погана оптимізація, додаток часто вилітає;
- Забагато реклами, що погіршує продуктивність додатку;
- Не працює функція «Рецензії», немає відгуків про книгу, неможливо дізнатись її стан;

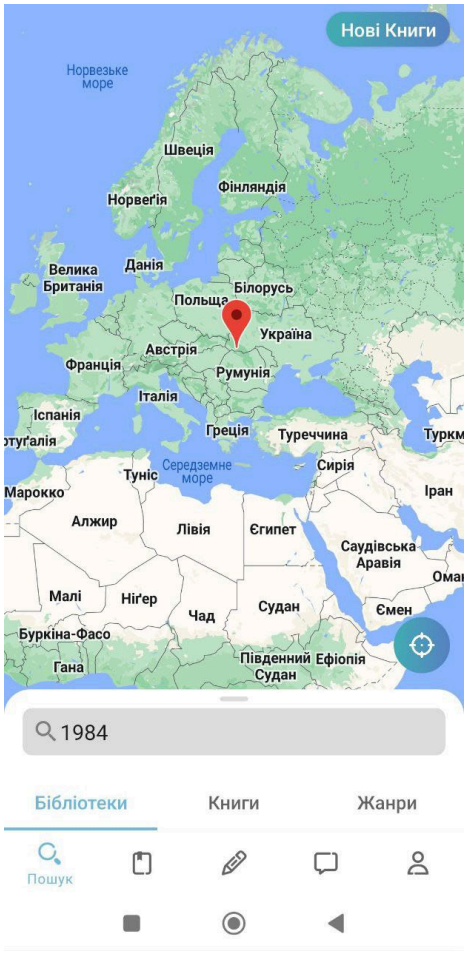


Рисунок 1.1 – Інтерфейс додатку BooksAround

Підсумовуючи, можна сказати, що додаток в майбутньому може отримати популярність, за умови якщо розробник виправить всі вище сказані недоліки, бо наразі там всього-на-всього кілька сотень завантажень.

1.2.2 – BookSwap

BookSwap [4] – це веб-додаток, створений в 2020 році під час пандемії коронавірусу. Він є доволі популярним у вузькому колі осіб, що підтверджується кількістю доступних книг на сайті. Зараз їх налічується близько 15-ти тисяч. Сайт має простий та зрозумілий інтерфейс(Див. рисунок 1.2), і у нового користувача не виникне жодних проблем з навігацією, тощо.

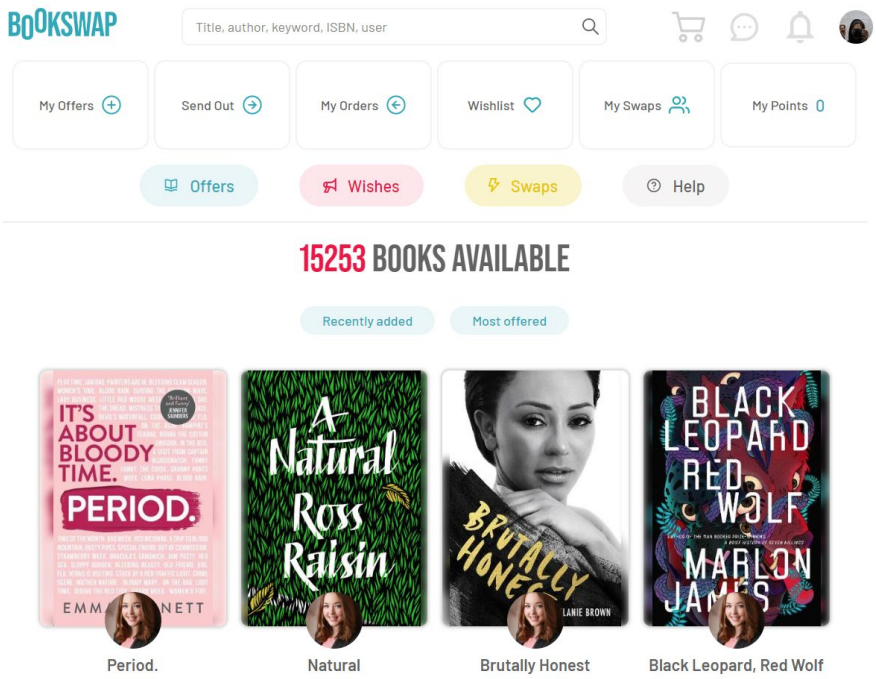


Рисунок 1.2 – Головна сторінка веб-сайту BookSwap

В цілому, додаток має багато плюсів, він простий у використанні, немає нічого зайвого, що може відволікати чи заплутати користувача. Також є відгуки, де користувачі можуть інформувати про стан книги. Але є один великий мінус, немає мобільного додатку, що в значну кількість разів скорочує потенційну аудиторію. Бо користуватись додатком, зручніше ніж веб-сайтом.

1.2.3 – BookCrossing

BookCrossing [5] – це сервіс, який був заснований в 2001 році письменницею Ронною Роуз у місті Канзас-Сіті, штат Міссурі, США. Ідея полягає в тому, що люди зможуть зареєструвати свої книги на сайті, позначити їх унікальним ідентифікатором та випустити на волю, щоб інші могли їх знайти та передати далі. Натхненням для цієї ідеї стала концепція «Геокешінгу» - гри, суть якої полягає в тому, щоб знаходити та залишати предмети у різних місцях вказаних на карті.

Кожна зареєстрована книга має унікальний ідентифікатор, який можна використовувати для відстеження її переміщення. Учасники можуть

відстежувати переміщення своїх книг на сайті (Див. рисунок 1.3) та отримувати повідомлення про їх нове місцезнаходження. Крім того, на сайті BookCrossing можна шукати книги за автором, назвою, жанром, мовою та місцезнаходженням.

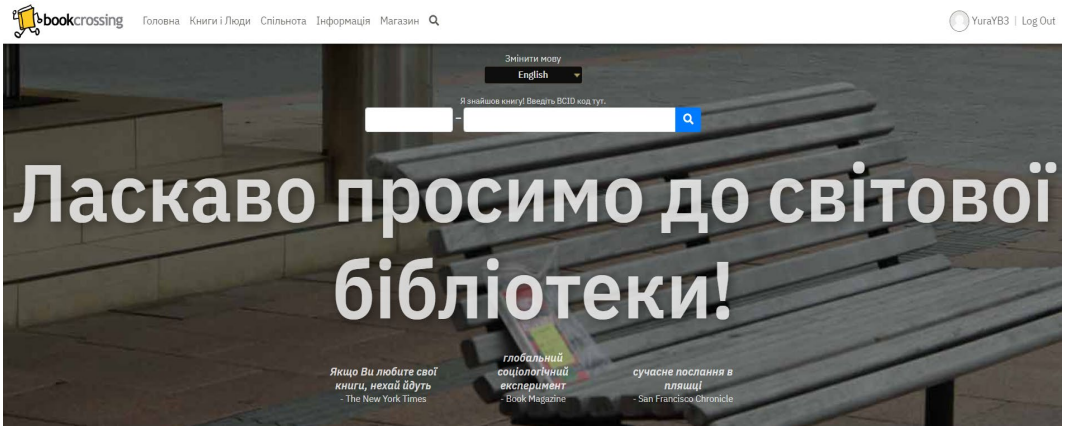


Рисунок 1.3 – Головна сторінка сайту BookCrossing

BookCrossing не має фізичних магазинів або складів. Це соціальна мережа, яка дозволяє людям ділитися своїми книгами з іншими. BookCrossing є безкоштовним сервісом, і кожен може приєднатися до цієї спільноти та випустити свої книги на волю.

Учасники можуть додавати відгуки та рецензії на книги, які вони знайшли або прочитали. Також можна приєднатися до груп та форумів, щоб обговорювати книги з іншими учасниками.

Проте, в цього сервісу є той самий головний недолік, що і в попередньому – це, відсутність мобільного додатку. Також, тут дуже складний інтерфейс і важко знайти саме те, що тобі потрібно. Багато вкладок, багато зайвої інформації, серед якої дуже просто загубитись і потенційний користувач, може покинути сайт, так і не зрозумівши, як саме ним користуватись.

1.3 Формування вимог до мобільного додатку

Успіх будь-якого мобільного додатку залежить від його можливості задовільнити потреби та очікування цільової аудиторії. Для того, щоб досягнути

бажаного результату необхідно сформулювати набір вимог, які чітко визначають функціональність, зручність використання, продуктивність та технічні аспекти додатку.

Функціональні вимоги мобільного додатку BookSharing включають реєстрацію та вхід користувача, створення власної бібліотеки, пошук книг, обмін книгами, рецензії та оцінки книг, месенджер та сповіщення. Реєстрація та вхід дозволяє користувачам створювати облікові записи та отримувати доступ до функцій додатку. Створення власної бібліотеки надасть змогу користувачам завантажити в додаток книги, які користувач готовий обміняти. Пошук книг дозволяє користувачам знаходити книги за їх вподобанням та інтересами. За допомогою рецензій та оцінок користувачі зможуть переглядати стан книг, та дізнаватись враження інших користувачів. За допомогою месенджера користувачі зможуть обговорити де та як їм обмінятись книгами. Сповіщення будуть повідомляти про запит на обмін, підтверджений запит та нові повідомлення.

Зручність використання є однією з ключових вимог до мобільного додатку. Це означає, що додаток повинен бути легким у використанні та мати інтуїтивно зрозумілий інтерфейс, щоб користувачі могли швидко зрозуміти, як взаємодіяти з додатком та виконувати потрібні дії. Необхідно реалізувати навігацію між вікнами, структурувати інформацію по категоріях(наприклад: книги, відгуки і т.д.), щоб користувач за допомогою одного-двох кліків, отримав потрібний йому результат. Також потрібно звернути увагу на кольорову схему додатку, обрати кольори які доповнюють один одного і не викликатимуть у користувача відразу, за необхідності можна створити 2-3 теми додатку, щоб користувач міг обрати яка йому більше до вподоби.

Продуктивність також є важливою вимогою до мобільного додатку. Додаток повинен бути швидким та ефективним у використанні, незалежно від обсягу даних, що обробляються. Додаток має працювати стабільно та безперебійно, щоб користувачі отримували задоволення при роботі з ним.

Додаток повинен бути сумісним з операційною системою Android. Також додаток повинен коректно відображатись на різних розмірах мобільних екранів. Для цього необхідно створити адаптивний дизайн мобільного додатку, який буде визначати розмір екрану користувача, і відповідно до нього відображати елементи. Крім того, додаток повинен підтримувати безпеку та захист персональних даних користувачів. Для того щоб це забезпечити, потрібно реалізувати аутентифікацію користувача.

Всі ці вимоги відіграють ключову роль в успіху майбутнього мобільного додатку, дотримування їх є вкрай важливим. Від цих вимог залежить кількість користувачів та потенційних доходів. Також потрібно пам’ятати, що цих вимог потрібно дотримуватись і після випуску додатку в «відкрите море», щоб тримати зацікавленість користувачів.

1.4 Побудова діаграм варіантів використання додатку

З наведеної вище інформації сформуємо варіанти використання, визначивши зовнішні сутності та зв’язки між ними. На рисунку нижче представлено варіанти використання додатку для неавторизованого користувача.

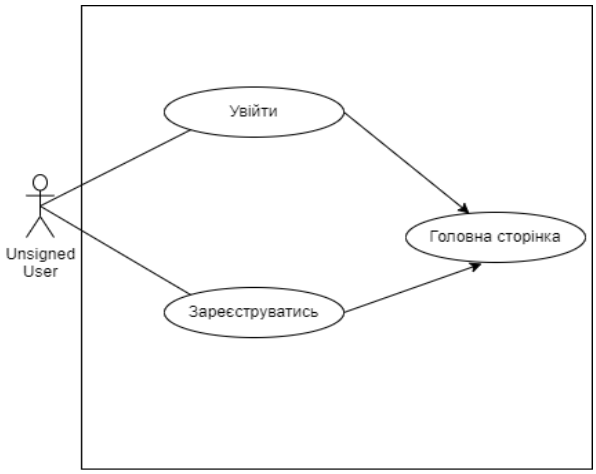


Рисунок 1.4 – Сценарій для неавторизованого користувача

Неавторизований користувач має лише дві примітивні дії: увійти в додаток або зареєструватись.

Увійшовши в додаток користувач має декілька сценаріїв подальшого використання. Сценарій для авторизованого користувача наведено в рисунку нижче.

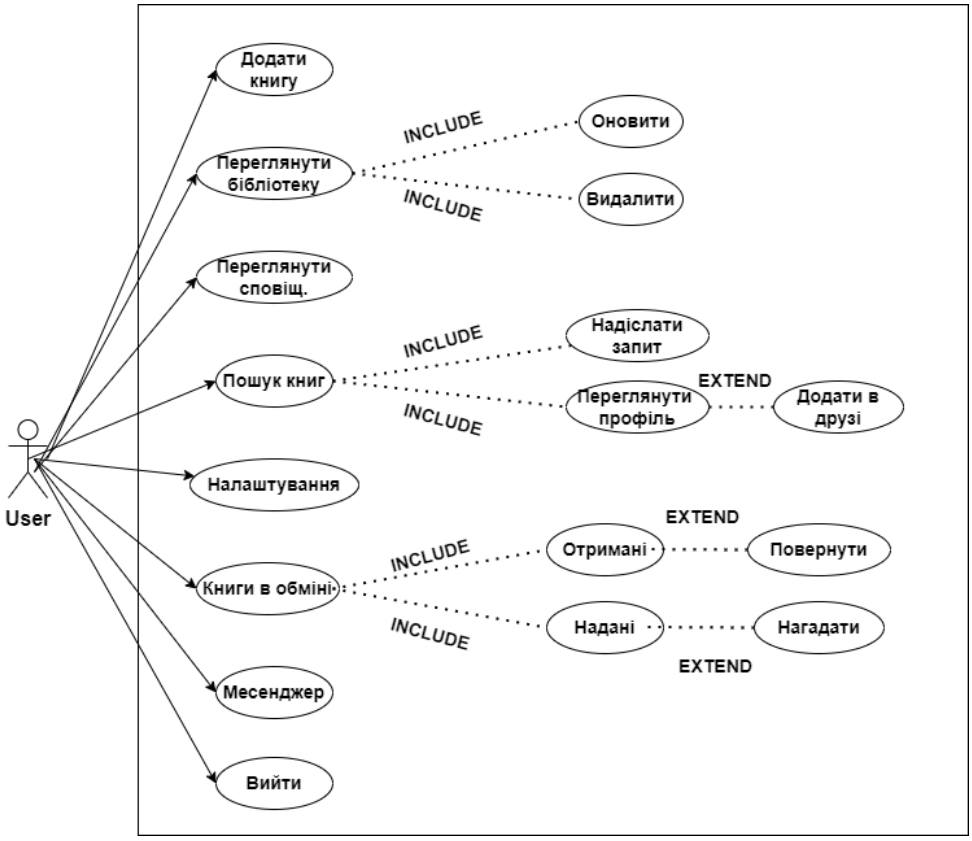


Рисунок 1.5 – Сценарій для авторизованого користувача

Як показано на схемі, користувач має такі варіанти використання:

- Додати книгу;
- Переглянути власну бібліотеку;
- Переглянути сповіщення;
- Пошук книг;
- Налаштування;
- Переглянути книг в обміні;
- Месенджер;
- Вийти з аккаунту;

Розгляньмо детальніше кожний з варіантів.

Після того як користувач авторизувався у додаток, перед ним з’являється екран вже з наявними в його бібліотеці книгами, також тут він може додати у власну бібліотеку нову книгу.

Обравши одну з наявних книг, користувач може переглянути інформацію про книгу, оновити її або видалити.

Обравши варіант переглянути сповіщення, користувач може переглянути наявні сповіщення якщо такі існують. Сповіщення можуть бути таких типів: запит на обмін книгами, результат надісланого запиту, нагадування та повідомлення від іншого користувача.

Якщо користувач вибере варіант пошук книги, то з’являється поле в якому потрібно вказати назву книги, після цього користувач може переглянути інформацію про неї, надіслати запит на обмін, переглянути профіль користувача та надіслати йому запит на спілкування.

Обравши варіант налаштування користувач може змінити інформацію про себе.

Також користувач має можливість переглянути книги які він отримав від інших користувачів та переглянути книги які він надіслав. В цьому ж вікні користувач може повернути книгу та нагадати про книгу.

Перейшовши в вікно «Месенджер» користувач може надіслати повідомлення раніше доданому користувачеві в друзі.

Обравши варіант «Вийти з додатку», користувач вийде зі свого профілю і перед ним знову появиться вікно авторизації.

1.5 Прототип додатку

Прототипування [6] – це створення малюнку, схеми або готового макета інтерфейсу користувача. Прототипи дозволяють зберегти час і гроші, які б могли бути витраченими на розробку невірних рішень.

Створення прототипу додатку є одним з найголовніших етапів у процесі проектування програмного коду. З його допомогою можна відобразити різні ідеї

дизайну мобільного додатку, протестувати їх, надати їм оцінку та змінити за необхідності. Існує багато способів прототипування, але в кожного з них одна мета – надати візуальну концепцію кінцевого продукту. Прототипи допомагають не тільки візуально, вони також є частиною дослідження умовного досвіду користувача і можуть допомогти з’ясувати які частини інтерфейсу потрібно змінити перед їх безпосередньою реалізацією.

Немає однозначного визначення, яким повинен бути прототип та як його потрібно створювати. Прототипом може бути як набір малюнків на папері, так і складною картинкою з ідеальними деталями. Вони можуть бути створенні декілька разів, в різний момент часу під час процесу розробки.

Зазвичай на початковому етапі створюється прототип з низькою деталізацією (Lo-Fi), що дозволяє перевірити використання та зрозумілість концепції дизайну для користувачів. Коли робота з Lo-Fi завершена, то створюється прототип з високою деталізацією (Hi-Fi), який повинен повністю відображати функціональність продукту.

Прототип Lo-Fi є засобом перетворення високорівневих концепцій на матеріальні та перевірені артефакти. Основне завдання цього прототипу – перевірка та тестування функціональності продукту перед візуальним оформленням. Головною перевагою Lo-Fi є те, зо це швидкий за часом та доступний за вартістю варіант Варто також зазначити, що Lo-Fi прототипи мають певні обмеження в складності взаємодії з користувачем.

Після створення Lo-Fi прототипу, створюється Hi-Fi прототип. Останній має вигляд найбільш схожий на фактично побудований додаток. Основним завданням Hi-Fi прототипу є створення більш складної та деталізованої взаємодії користувача з інтерфейсом та отримання кращих відгуків під час тестування. Це можливо завдяки тому, що Hi-Fi прототип виглядає як реальний мобільний додаток, що дозволяє користувачам поводитись більш природньо та давати більш точні відгуки, ніж у випадку з Lo-Fi прототипом.

Узагальнюючи, створення Lo-Fi та HiFi прототипів є важливим етапом у процесі розробки програмного продукту. Ці два типи прототипів

використовуються для тестування та визначення функціональних вимог перед фактичною розробкою програмного коду. Хоча Lo-Fi прототипи дозволяють швидко та доступно створити прототип, Hi-Fi прототипи забезпечують кращу зворотну інформацію та можливість більш детального тестування з більш складною взаємодією з користувачами. Отже, важливо правильно використовувати Lo-Fi та Hi-Fi, враховуючи їх переваги та обмеження, з метою максимально ефективної розробки продукту.

Скористаємось раніше побудованою діаграмою варіантів використання додатку для побудови прототипів. Прототип повинен містити всі важливі функції, а також неочевидні, щоб зрозуміти їхню необхідність в проекті.

Для початку створимо Lo-Fi прототип, простий малюнок на аркуші, в якому продемонструємо головну сторінку додатку з її головними елементами. На рисунку нижче представлено Lo-Fi прототип.

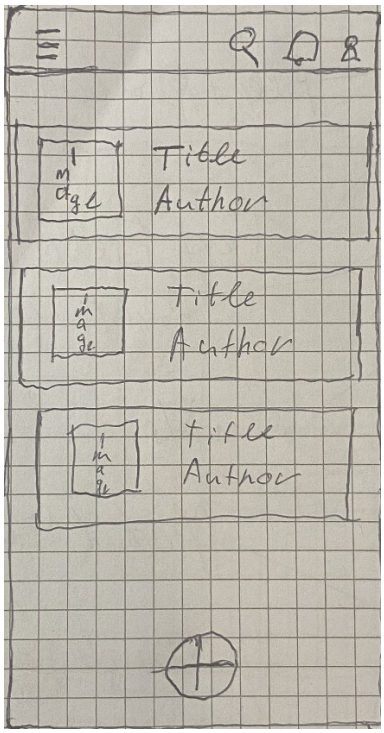


Рисунок 1.6. – Lo-Fi прототип BookSharing

Зверху на рисунку ми бачимо верхню панель на якій розміщені такі елементи: зліва знаходиться іконка навігаційного меню, а з правої сторони іконка пошуку, іконка сповіщень та іконка профілю. Посередині екрану знаходяться

список з книг раніше доданих користувачем у власну бібліотеку. Внизу екрану знаходиться кнопка для додавання нової книги.

Отримавши умовне розуміння як має виглядати додаток, приступимо до створення Hi-Fi прототипу. Для його реалізації скористаємось сервісом Figma. Figma – це онлайн інструмент для дизайну та проектування, який дозволяє створювати високоякісні макети веб-сайтів, мобільних додатків та інших проектів. Перевагою Figma, є те, що він є інтерактивним інструментом, що дозволяє створювати прототипи з функціональними кнопками та елементами управління.

Продемонструємо декілька інтерактивних функцій у створеному Hi-Fi прототипі. На рисунку нижче зображена головна сторінка прототипу додатка.

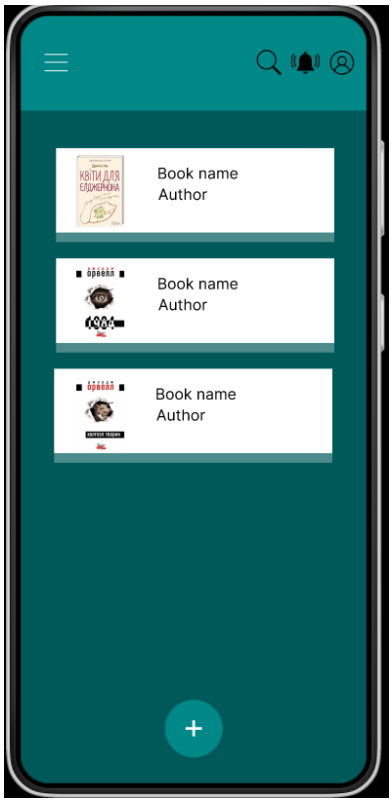


Рисунок 1.7 – Hi–Fi прототип додатку

Натиснувши на іконку у верхньому лівому куті, перед користувачем з’являється бокова панель навігації. Вона продемонстрована на рисунку нижче.

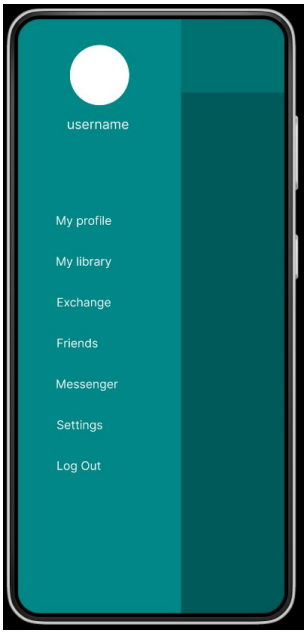


Рисунок 1.7 – Панель навігації

Якщо ж на головному екрані натиснути на кнопку додавання нової книги, то перед користувачем з’являється відповідне вікно, яке зображене на рисунку нижче.

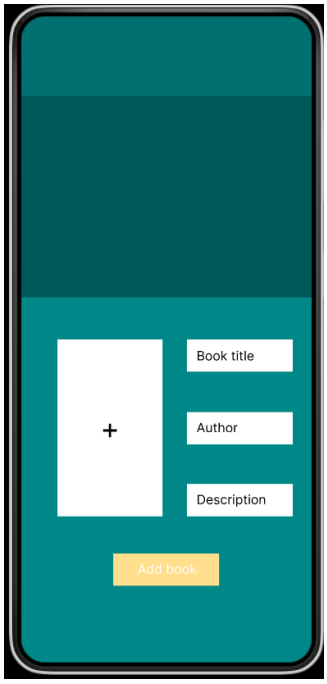


Рисунок 1.8 – Вікно додавання нової книги

Ознайомитись з усіма функціями прототипу можна за посиланням [7].

РОЗДІЛ 2 РОЗРОБКА ТА ВИКОРИСТАННЯ МОБІЛЬНОГО ДОДАТКУ BOOKSHARING

2.1 Обґрунтування вибору засобів розробки мобільного додатку

На сьогоднішній день є багато різних фреймворків для створення мобільних додатків[8], такі як Xamarin, React Native, Ionic, Flutter та багато інших. Але саме Flutter виділяється серед них величезною кількістю бібліотек, які значно полегшують написання додатку і розробнику не потрібно «винаходити велосипед».

Задля збереження та обробки інформації використаємо сервіс Firebase [12]. Він як і Flutter створений компанією Google, тож вони гарно взаємодіють між собою і при розробці виникає мінімум труднощів. Також Firebase надає можливість авторизації та аутентифікації користувача, що нам знадобиться для безпеки персональних даних користувача. Детальніше Flutter та Firebase розглянемо в наступних підрозділах.

2.1.1 Аналіз переваг фреймворку Flutter

Flutter – це сучасний фреймворк, який надає можливість розробникам створювати інтерактивні додатки за допомогою потужних інструментів розробки. Головною ідеєю Flutter є простота [13], лозунг цього фреймворку звучить як – Everything is a widget! (Все є віджетом!). Віджет є незмінним блоком додатку, який є частиною інтерфейсу користувача. Кожен віджет визначає різні структурні елементи такі як кнопка, елементи стилю, елементи розміщення об’єкту і багато інших.

Завдяки двовимірному рендериговому двигуну та попередньо визначеним віджетам, можна з легкістю відтворити будь-який дизайн додатку. Віджети комбінуються в дерево віджетів, і цим самим створюють складні інтерфейси з

безліччю елементів. Коли користувач взаємодіє з додатком, то фреймворк автоматично перебудовує ту частину дерева віджетів, в яку було внесено зміни.

Приклад дерева віджетів зображено на рисунку 2.1.

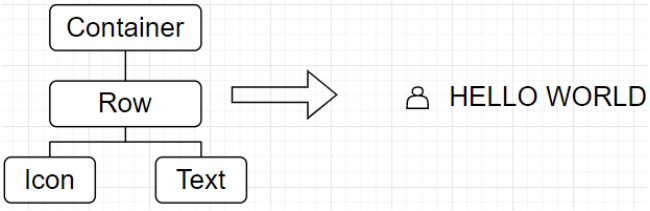


Рисунок 2.1 – Приклад дерева віджетів

Flutter дозволяє розробникам створювати та використовувати маленькі віджети і компонувати їх задля створення більш масштабних інтерфейсів та макетів. Розгляньмо приклад з рисунку 2.1, тут головним віджетом є Container, він використовується задля створення прямокутного елементу. Він за своєю суттю схожий на елемент div в HTML. Нижче знаходиться віджет Row, він відповідає за те, що його нащадки будуть розміщені в горизонтальному напрямку. Його нащадками є віджет Icon який зображає на екрані іконку користувача та віджет Text “HELLO WORLD”. Код до цього дерева віджетів продемонстровано в лістингу 2.1.

Лістинг 2.1 – Приклад композиції дерева віджетів

```
Container(
     padding: EdgeInsets.all(5.0),
     child: Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
               Icon(Icons.person),
               Text('HELLO WORLD'),
               ],
          ),
     );
```

Flutter використовує мову програмування Dart, яка також розроблена компанією Google, та має схожий стиль з мовою JavaScript. Dart поєднує в собі

ряд характеристик, які роблять його привабливим для розробників. Декілька ключових особливостей мови Dart:

- Dart має статичну типізацію, що означає, що тип кожної змінної визначається на етапі компіляції. Це допомагає зменшити кількість помилок в програмах та зробити їх більш гнучкими до змін.
- Dart підтримує асинхронне програмування, що дозволяє розробникам створювати ефективні та швидкодіючі програми з високим рівнем паралельних потоків.
- Гнучкість також є одною з ключових характеристик, за допомогою цієї мови можна створювати як веб-додатки так і мобільні додатки. Dart також підтримує різні платформи, такі як Windows, MacOS, Linux, Android та iOS.
- Dart має простий та зрозумілий синтаксис, що дозволяє швидко вивчити мову та розпочати розробку програм.

Фреймворк Flutter складається з декількох рівнів (Див. рисунок 2.2), де кожен рівень використовує попередній. Верхній рівень використовується розробниками кожного разу, а нижчі рівні використовуються при створенні певних особливостей.

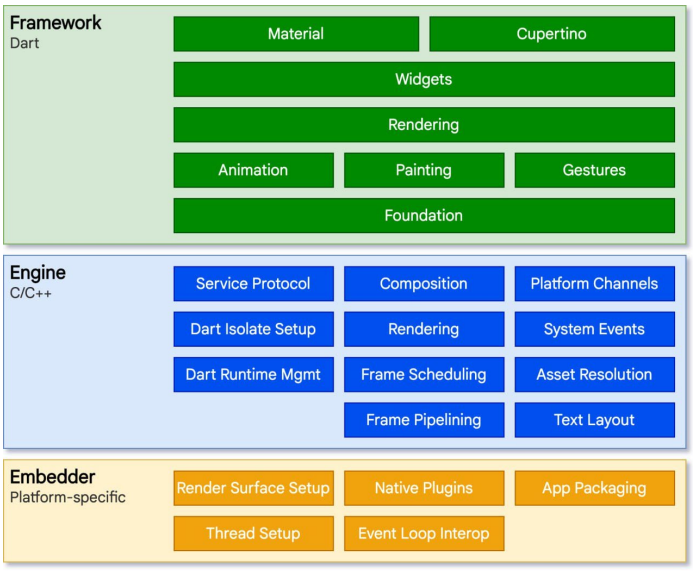


Рисунок 2.2 – Структура фреймворку Flutter

Зазвичай розробники взаємодіють з Flutter через Flutter Framework. Він містить багатий набір бібліотек для платформи, макетів та основ, що складаються з кількох рівнів. Починаючи знизу догори, ми маємо:

- Основні базові класи та служби будівельних блоків, такі як анімація, малювання та реагування на дотик до екрану.
- Рендерний шар надає абстракцію для роботи з макетом. На цьому рівні будується дерево рендерних об’єктів, які автоматично змінюються, якщо користувач вніс якісь зміни.
- Шар віджетів є абстракцією композицій. Кожен рендерний об’єкт в рендерному шарі має відповідний клас в шарі віджетів. Крім того, шар віджетів дозволяє визначити комбінації класів, які можна повторно використовувати.
- Бібліотеки Material та Cupertino надають комплексні набори елементів управління, які використовують примітивні композиції шару віджетів для реалізації дизайну для платформи Android або iOS.

2.1.2 – Структура та можливості платформи Firebase

Firebase [14] – це комплексний набір інструментів та сервісів, що позиціонується як платформа Backend-as-a-Service (BaaS), що надає можливість розробникам створювати та розширювати як мобільні, так і веб додатки. Firebase надає базу даних в реальному часі, автентифікацію, хостинг та інші функції, якими можна управляти за допомогою зручного інтерфейсу .

Firebase синхронізує дані на всіх підключених пристроях. База даних використовує модель даних з документами NoSQL, що дозволяє зберігати дані у зручний та гнучкий спосіб. Дані зберігаються в форматі JSON, база даних також підтримує атомарні операції та повідомлення про події в режимі реального часу.

Firebase також надає потужні сервіси для автентифікації, що надає розробникам можливість легко створювати безпечну автентифікацію користувача у додатку. Firebase підтримує декілька варіантів автентифікації,

таких як електронна пошта та пароль, номер телефону, сторонні сервіси, такі як Facebook, Google, Twitter.

Окрім бази даних та сервісів автентифікації, Firebase також надає хмарне середовище [15], що надає розробникам можливість зберігати великі файли, такі як зображення та відео. Середовище інтегровано з базою даних, що надає швидкий доступ до зберігання та отримання файлів.

Також в Firebase є сервіс Google Cloud Functions. Основна ідея Cloud Functions полягає в тому, щоб дозволити розробникам писати функції, які будуть виконуватись при спеціальних подіях, таких як виклик API, CRUD дії, публікація повідомлень.

Основні особливості Cloud Functions:

- Легкість використання. Розробка функцій зворотного виклику займає небагато часу і зусиль. Функції можна писати на кількох популярних мовах програмування, таких як JavaScript , Node.Js, Python і т.д.
- Розподілена обробка подій. Функції виконуються в середовищі хмари, що означає, що вони можуть гнучко масштабуватись в залежності від навантаження.
- Інтеграція з іншими сервісами Google: Google Cloud Functions має глибоку інтеграцію з іншими хмарними сервісами Google, такими як Firebase, Google Cloud Pub/Sub, Google Cloud Storage, Google Cloud Firestore та іншими. Це дозволяє вам створювати потужні функції, які взаємодіють з різними сервісами і виконують складні операції.
- Моніторинг та налагодження: Google Cloud Functions надає інструменти для моніторингу та налагодження ваших функцій. За допомогою них можна переглядати журнали виконання, створювати точки зупину, аналізувати ресурси та отримувати сповіщення про події.

Firebase також пропонує набір інструментів для аналізу використання та продуктивності додатку, надає звіти про помилки, моніторинг роботи додатку. Ці інструменти надають корисну інформацію про додаток, що допомагає

розробникам знаходити та усувати помилки які пов’язані з продуктивністю додатка.

Підсумовуючи, можемо сказати, що Firebase забезпечує широку платформу для створення та розвитку високоякісних мобільних додатків. Його інструменти дозволяють розробникам зосередитись на створенні якісних та оптимізованих додатках.

2.2 – Створення додатку

В цьому підрозділі розглянемо структуру проекту, опишемо його архітектуру, продемонструємо підключення до Firebase, реалізацію аутентифікації та отримання сповіщень.

2.2.1 – Структура проекту

Для того щоб мінімізувати непередбачувану поведінку та зробити проект більш читабельним, було реалізовано трьохрівневу архітектуру.

Трьохрівнева архітектура [16] – це модель побудови програмного забезпечення, що складається з трьох рівнів: інтерфесний рівень, рівень бізнес-логіки та рівня даних.

Інтерфесний рівень відповідає за взаємодію з користувачем. Він забезпечує візуальне представлення даних та управління вводом-виводом інформації. Це рівень включає в себе графічний інтерфейс, який дозволяє користувачам взаємодіяти з додатком.

Бізнес-логіка визначає, яким чином функціонуватиме додаток. Цей рівень включає в себе логіку обробки даних, правила бізнес-процесів та управління додатком. Також цей рівень відповідає за перетворення даних між інтерфейсним та рівнем даних.

Рівень даних відповідає за зберігання та доступ до даних. Цей рівень включає в себе базу даних та налаштування, які забезпечують доступ до даних з

бізнес-логіки. Приклад взаємодії трьохрівневої архітектури зображено на рисунку 2.3

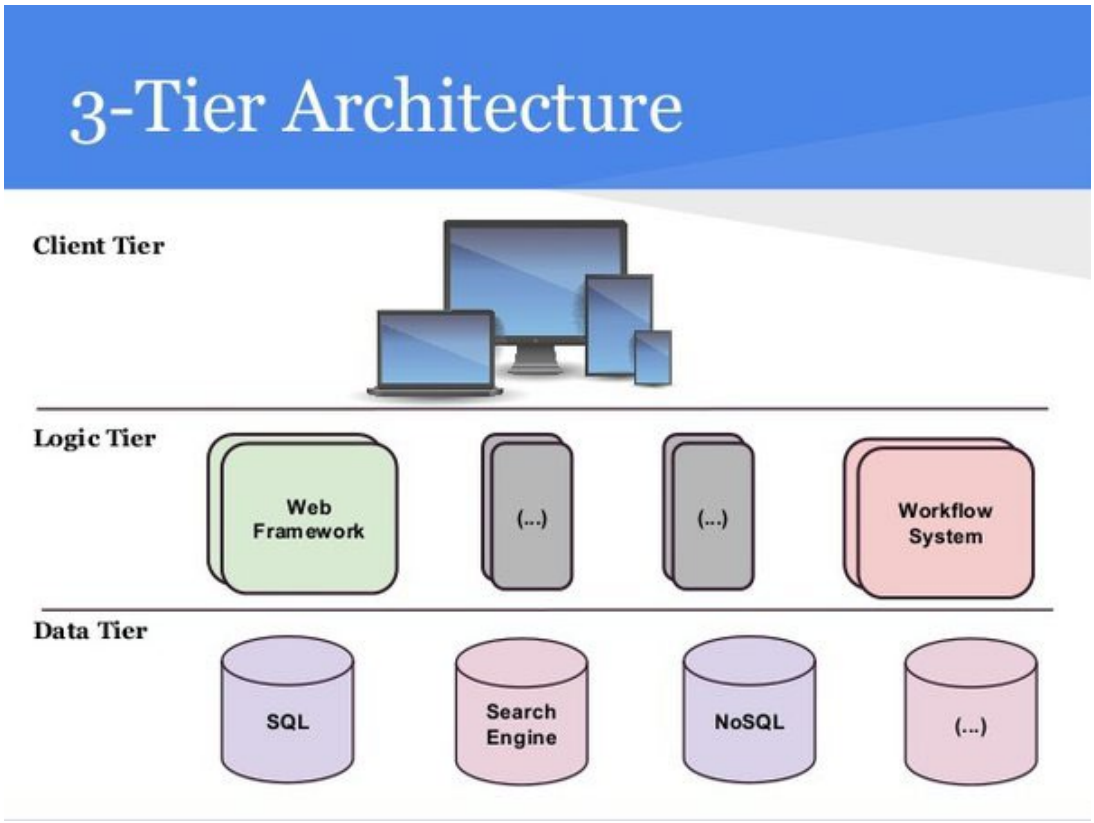


Рисунок 2.3 – Трьохрівнева архітектура

У трьохрівневій архітектурі кожен рівень виконує свої функції та взаємодіє з іншими рівнями, щоб забезпечити функціональність програмного забезпечення. Така архітектура дозволяє забезпечити гнучкість та легкість розширення програмного забезпечення.

Отже, проект розділено на три головні теки:

- Models
- Screens
- Services

В папці Models знаходяться файли які відповідають за представлення моделі в базі даних Firebase (Див. додаток А). В Firebase таблиці називаються колекціями, а кожен запис в них – документами. Документи можуть містити поля з різними типами даних, такими як рядки, числа, булеві значення, дата та інші.

Також важливо зазначити, що документи в Firebase можуть бути взаємозалежними. Наприклад у проекті є дві моделі: «userModel» та «bookModel», остання буде містити в собі поле, яке посилатиметься на документ «userModel». Кожна модель в проекті містить поля з вказаним типом, які будуть додані до конкретної колекції, а також методи для серіалізації та десеріалізації об’єктів. В проекті використовуються такі моделі: «userModel», «bookModel», «bookSwapModel», «friendshipModel», «notificationModel», «reviewModel», «messageModel». Розгляньмо детальніше кожну з них.

Модель «userModel» містить в собі інформацію про користувача. Поля до цієї моделі наведено в таблиці нижче.

Таблиця 2.1 – Поля userModel

Назва поля	**Тип поля**	**Опис**
userNickName	String	Нік користувача
userEmail	String	Електронна адреса користувача
userImage	String	Зображення профілю користувача
userAge	Int	Вік користувача
uid	String	Унікальний ідентифікатор
userToken	String	Токен пристрою користувача
emailVerified	Bool	Перевірка на підтвердження акаунту

Тут потрібно виділити три останні поля. Поле «uid» зберігає в собі унікальний ідентифікатор який генерується Firebase при створенні нового документу в колекції. Тобто «uid» зберігає в собі ідентифікатор саме згенерованого документу.

Поле «userToken» зберігає в собі токен пристрою з якого користувач увійшов додаток, він буде оновлюватись при кожному відкритті додатку. Цей токен ми в подальшому будемо використовувати задля надсилання сповіщень на пристрій користувача.

Поле «emailVerified» буде використовуватись для перевірки чи користувач підтвердив свою електронну адресу.

Модель «bookModel» містить в собі інформацію про книгу додану користувачем. Поля до цієї моделі наведено в таблиці 2.2.

Таблиця 2.2 – Поля bookModel

Назва поля	Тип поля	Опис
name	String	Назва книги
author	String	Автор книги
userId	String	Id користувача
bookId	String	Id книги
available	Bool	Доступність книги
cover	String	Посилання на обкладинку
description	String	Опис книги

В цій моделі цілком зрозуміло, які поля і за що відповідають. Варто виділити поле «available» воно буде використовуватись задля перевірки чи можна цю книгу отримати на обмін. А також від цього параметру буде залежати, чи відображатиметься книга при пошуку. Тож перейдемо до наступної моделі – «notificationModel». Вона відповідає за те яке сповіщення та з якою інформацією отримає користувач. Поля до цієї моделі наведено в таблиці нижче.

Таблиця 2.3 – Поля notificationModel

Назва поля	Тип поля	Опис
notificationID	String	Згенероване унікальне id
senderID	String	Id відправника
receiverID	String	Id отримувача
notificationType	String	Тип сповіщення
seenByReceiver	Bool	Перевірка чи бачив користувач сповіщення
deseiredBookID	String	Id книги

Тут потрібно виділити поле «notificationType». Як вже було сказано в першому розділі, сповіщення у нас можуть бути трьох видів: запит на обмін книгами, запит в друзі, та нагадування про книгу. З цих трьох типів виділяється «запит в друзі», оскільки він не використовує поле «deseiredBookId.» Для того щоб це реалізувати, використаємо особливість мови Dart – null safety [17].

Мова програмування Dart має свою особливість – систему типізації з підтримкою null. Ця особливість називається null safety і вона забезпечує більш надійне та безпечне програмування.

За замовчуванням змінні в Dart можуть мати значення null, що означає відсутність значення. Це може призвести до непередбачуваної поведінки програми, коли нульове посилання використовується некоректно. Тому, щоб уникнути цього, у Dart було введено систему типів з підтримкою null. За допомогою анотації “?” можна позначити, що змінна може містити null. Також, за допомогою оператора “!” можна позначити, що змінна точно не має значення null.

Поставимо навпроти типу String атрибут «?», а в конструкторі не вказуємо аргумент required. Що дозволить нам створювати об’єкт класу без поля «desiredBookID» та мати уніфіковані сповіщення. Також користувачу буде надходити сповіщення при отриманні нового повідомлення, але для цього у нас буде окрема модель та окрема логіка.

Наступна наша модель –«bookSwapModel», вона відповідає за обмін книгою між двома користувачами. В таблиці нижче наведено поля до нашої моделі.

Таблиця 2.4 – Поля bookSwapModel

Назва поля	**Тип поля**	**Опис**
1	**2**	**3**
bookSwapID	String	Згенероване унікальне id
swapReqID	String	Id сповіщення

1	2	3
receiverID	String	Id отримувача
senderID	String	Id відправник
deseiredBookID	String	Id книги

В цій моделі також все зрозуміло з назв полей. По своїй суті модель складається з посилань на інші документи. Вона відповідатиме за підтверджений обмін книгами. Перейдемо до наступної моделі «friendshipModel». Ця модель зберігатиме в собі інформацію про зв'язок між двома користувачами. Поля цієї моделі представлено в таблиці нижче.

Таблиця 2.5 – Поля friendshipModel

Назва поля	Тип поля	Опис
friendshipID	String	Згенероване унікальне id
firstUser	String	Id першого користувача
secondUser	String	Id другого користувача

Тут також все просто, в цій колекції зберігаються унікальні ідентифікатор двох користувачів та згенерований ідентифікатор документу, який в подальшому буде використовуватись для особистого чату між користувачами. Перейдемо до моделі «messageModel». Поля до цієї моделі наведено нижче.

Таблиця 2.6 – Поля messageModel

Назва поля	Тип поля	Опис
message	String	Текст повідомлення
senderID	String	Id відправника
sendTime	DateTime	Час відправки

Ця модель містить в собі інформацію про відправлене повідомлення. Ключовим полем тут є «sendTime», оскільки завдяки ньому ми будемо відображати повідомлення в правильному порядку. Останньою моделлю є «reviewModel», вона буде містити в собі інформацію про відгук до книги. Поля до неї наведено в таблиці нижче.

Таблиця 2.7 – Поля reviewModel

Назва поля	**Тип поля**	**Опис**
bookId	String	Id книги
userId	String	Id користувача
review	String	Відгук до книги
rating	Double	Рейтинг книги

Тут варто виділити поле «rating», в подальшому за допомогою програмного коду та сторонніх віджетів воно буде обмежуватись від нуля до п’яти.

Перейдемо до теки Screens. В цій папці у нас будуть розміщені віджети та екрани, які будуть відповідати за відображення інтерфейсу користувача. Для прикладу розглянемо уривок коду, який відповідає за вхід в додаток. Код до цього віджету наведено в лістингу 2.1, на рисунку 2.6 зображено його візуалізацію.

Рисунок 2.6 – Форма авторизації

Лістинг 2.1 – Уривок коду loginWidget

```
Padding(
   padding: const EdgeInsets.all(8.0),
   child: TextFormField(
   decoration: textInputDecoration.copyWith(hintText: «E-Mail»),
   validator: (value) => value!.isEmpty ? 'Введіть email' : null,
   style: const TextStyle(),
   onChanged: (val) {
      setState(() {
        email = val;});},),),
const SizedBox(height: 15),
Padding(
   padding: const EdgeInsets.all(8.0),
   child: TextFormField(
   decoration: textInputDecoration.copyWith(hintText: «Пароль»),
   obscureText: true,
   validator: (value) =>
   value!.length < 8 ? 'Введіть пароль з 8+ символів' : null,
   style: const TextStyle(),
   onChanged: (val) {
      setState(() {
        password = val;
       });},),),
```

На цьому уривку зображено форму для входу в додаток. Тут віджет Padding відповідає за відступ в середині контейнеру. Атрибут validator перевіряє введений текст на поставлені умови. І кожного разу коли користувач вносить зміни в віджеті TextFormField, то за допомогою атрибуту onChanged викликається анонімна функція і в змінній встановлюється нове значення.

В папці Services у нас знаходяться файли, які будуть взаємодіяти з базою даних та виконувати CRUD операції. Для прикладу візьмемо файл «authService» в якому знаходяться функції, які відповідають за вхід, вихід та реєстрацію в додатку. Розглянемо детальніше функцію входу в додаток.(Див. лістинг 2.2)

Лістинг 2.2 – Функція входу в додаток

```
Future signInWithPassword(String email, String password) async {
         try {
     UserCredential            result            =            await
_auth.signInWithEmailAndPassword(
     email: email, password: password);
           User? User = result.user;
```

```
            return _userFromFirebase(user);}
          on FirebaseAuthException catch € {
            String errorMessage;
            if (e.code == 'user-not-found') {
              errorMessage = 'Користувача з таким email не існує';}
            else if (e.code == 'wrong-password') {
              errorMessage = 'Неправильний пароль';}
            else {
              errorMessage = e.message ?? e.toString();}
            Fluttertoast.showToast(
              msg: errorMessage,
              gravity: ToastGravity.BOTTOM,
              backgroundColor: const Color.fromARGB(255, 187, 38,
38), textColor: Colors.white,);}}
```

Тут з форми авторизації ми отримуємо два параметри: електронну адресу та пароль. Отримавши дані, ми пробуємо увійти в додаток використавши метод signInWithEmailAndPassword, який є методом firebase і надається пакетом firebase_auth. Результатом цього методу буде авторизований користувач або null, що означитиме, що такого користувача не існує. Якщо ж при авторизації виникнуть помилки, то про це додаток сповістить користувача за допомогою віджету Fluttertoast.showToast, він надається пакетом fluttertoast. Ознайомитись зі всім кодом можна за посиланням [18].

2.2.2 – Підключення проекту до Firebase

Для того щоб підключити Firebase до додатку, необхідно перейти в консоль Firebase і обрати операційну систему на якій буде працювати додаток. Після цього на екрані з'являється вікно реєстрації додатку, воно продемонстровано на рисунку нижче.

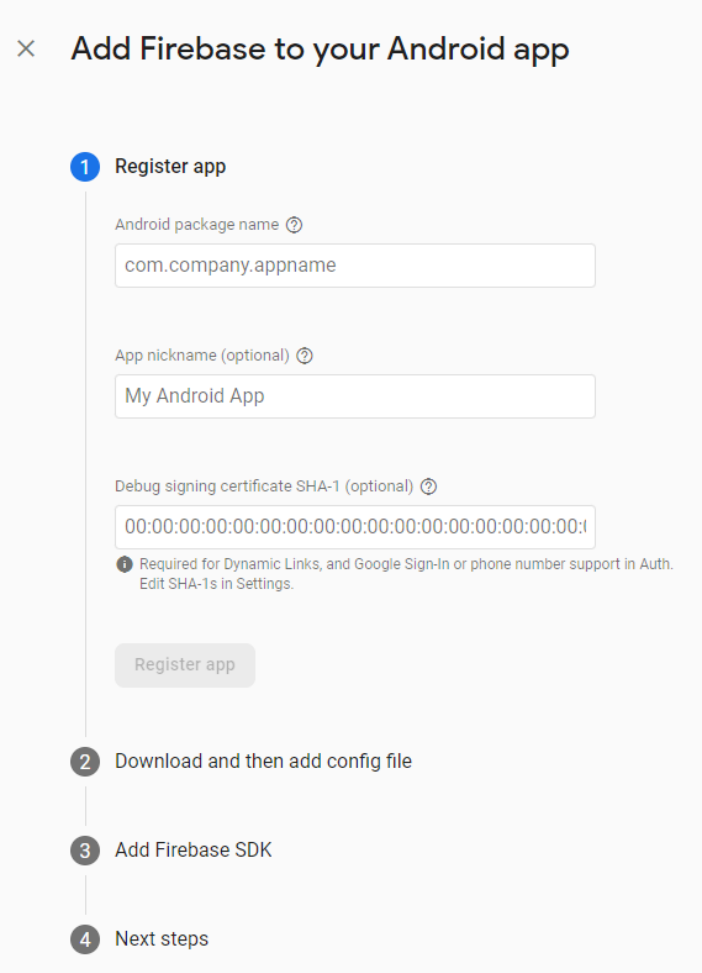


Рисунок 2.7 – Вікно реєстрації застосунку

Тут необхідно ввести назву додатку та його сертифікаційний ключ, його можна знайти в проекті за шляхом: android/app/build.gradle та знайти рядок під назвою aplicationID. Після вводу всіх даних, Firebase згенерує файл google–services.json який необхідно завантажити в проект за шляхом android/app. І фінальним кроком, необхідно додати в головну функцію проекту два рядки, які будуть використовуватися для ініціалізації Firebase :

```
WidgetsFlutterBinding.ensureInitialized();
     await Firebase.initializeApp();
```

2.2.3 – Реалізація надсилання сповіщень

Щоб реалізувати можливість надсилання сповіщень, необхідно скористатись сервісом Cloud Functions в Firebase. За допомогою цього сервісу

можна створювати та розгортати функції зворотного виклику безпосередньо в Firebase.

Для того щоб підключити Cloud Functions до свого проекту необхідно перейти в Firebase консоль і активувати його для проекту. Після цього безпосередньо в проект необхідно додати залежність cloud_functions. Для прикладу продемонструємо функцію (Див. додаток Б). , яка відповідає за надсилання сповіщення при отриманні нового повідомлення. Розглянемо детальніше декілька моментів з коду.

Функція буде спрацьовувати при створенні нового документу в колекції «message/{friendshipID}/dialogue/{messageID}», за допомогою цього коду:

```
exports.onNewMessage = functions.firestore
    .document(«message/{friendshipID}/dialogue/{messageID}»)
    .onCreate(async (snapshot, context) => {
```

За допомогою коду продемонстрованого нижче, отримуємо дані зі створеного документу та отримуємо параметр «friendshipID» з його контексту.

```
Const mes = snapshot.data();
const fID = context.params.friendshipID;
```

У коді продемонстрованому нижче перевіряємо наявність документу дружби за допомогою ідентифікатора «fID». Якщо такий документ не знайдено, логується повідомлення про відсутність дружби та функція припиняє виконання.

```
Const fRef = ad.firestore().collection(«friendship»).doc(fID);
const friendshipDoc = await fRef.get();
const fsh = friendshipDoc.data();

if (!fsh) {
  console.log(`Friendship with ID ${fID} does not exist`);
  return;
}
```

На нижче наведеному фрагменті кода визначаємо ідентифікатор отримувача повідомлення, порівнюючи ідентифікатор відправника

повідомлення «mes.senderID» з ідентифікаторами користувачів у документі дружби. Ідентифікатор отримувача зберігається у змінній «rID» і логується в консоль.

```
Const rID = fsh.user1_ID === mes.senderID ? fsh.user2_ID : fsh.user1_ID;
console.log(`Recipient ID: ${rID}`);
```

Тепер підготуємо наше сповіщення для відправлення на пристрій користувача (Див. лістинг 2.3). Щоб дізнатись на який пристрій необхідно відправити сповіщення, будемо використовувати токен пристрою. Він оновлюється кожного разу, коли користувач видаляє додаток, або заходить з іншого пристрою.

Лістинг 2.3 – Підготовка та відправка сповіщення

```
const payload = {
  notification: {
    title: `Нове повідомлення від ${sendData.name}`,
    body: mes.message,
    clickAction: «FLUTTER_NOTIFICATION_CLICK»,
  },
  data: {
    screen: «messages»,
    click_action: «FLUTTER_NOTIFICATION_CLICK»,
    message_id: snapshot.id,
    friend_id: fID,
  },
};

     const resp = await ad.messaging().sendToDevice(recp.userToken, payload);
     console.log(`Message sent to device with token ${recp[«userToken»]}`);
     console.log(`Messaging API response: ${JSON.stringify(resp)}`);
```

У цьому фрагменті коду створюється об’єкт «payload», який містить дані для сповіщення та додаткову інформацію. Після цього сповіщення відправляється на пристрій отримувача за допомогою Firebase Messaging API. Результат відправки логується у консоль.

2.3 – Реліз мобільного додатку BookSharing

Розробка мобільних додатків стає більш популярною та вимагає старанного підходу до процесу релізу [19]. Правильний випуск додатку є ключовим фактором для досягнення успіху на ринку програмного забезпечення. У цьому розділі ми розглянемо основні кроки та вимоги, які необхідно виконати для успішного випуску додатку.

Перш за все необхідно мати акаунт девелопера в Google Play Market. Щоб його отримати необхідно скористатись сервісом Google Play Console. Тут потрібно ввести всю необхідну інформацію про розробника. Вікно вводу інформації наведено на рисунку нижче.

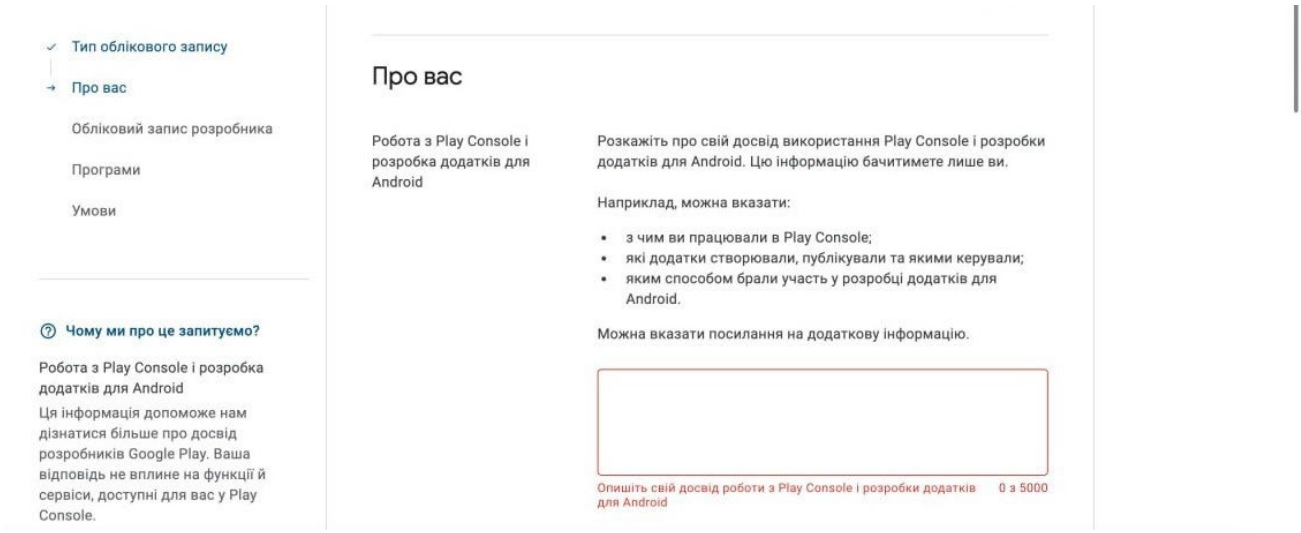


Рисунок 2.8 – Створення облікового запису

Після вводу всієї інформації, необхідно оплатити єдино разовий платіж розміром 25-ти доларів. Після успішної сплати, можна користуватись всіма функціями Google Play Console, але також необхідно підтвердити свою особистість завантаживши фото власного паспорту.

Тепер можна завантажити додаток в Play Market. Щоб розпочати процес створення нового додатку необхідно обрати пункт «Створити додаток». Тут необхідно заповнити поля з основною інформацією, вказати назву додатку, мову за замовчуванням, чи є додаток безкоштовним і т.д. Після заповнення основної

інформації, переходимо до вказання більш детальної. Цей процес є доволі громіздким, він складається з таких етапів: «Політика конфіденційності», «Доступ до додатку», «Цільова аудиторія», «Безпека даних» і т.д. Всі пункти які необхідно заповнити наведено на рисунку нижче.

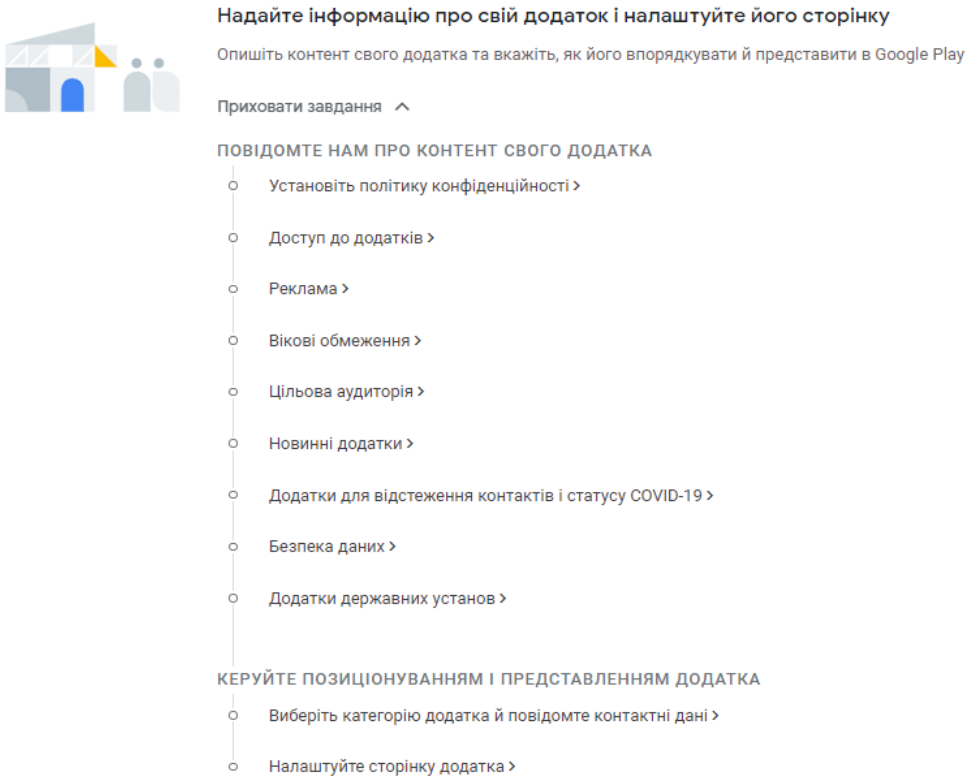


Рисунок 2.9 – Пункти реєстрації додатку

Тут ключовими пунктами є «Політика конфіденційності» та «Безпека даних». Для того щоб їх заповнити необхідно завантажити два файли: «Privacy policy» та «Terms&Conditions». В цих файлах має знаходитись інформація про те, яку інформацію користувача використовує додаток, як вона зберігається, як зв’язатись з розробником у разі порушення прав і т.д. Для того щоб згенерувати ці файли, використаємо спеціалізований для цього сайт[20]. Вікно генерації наведено на рисунку нижче.

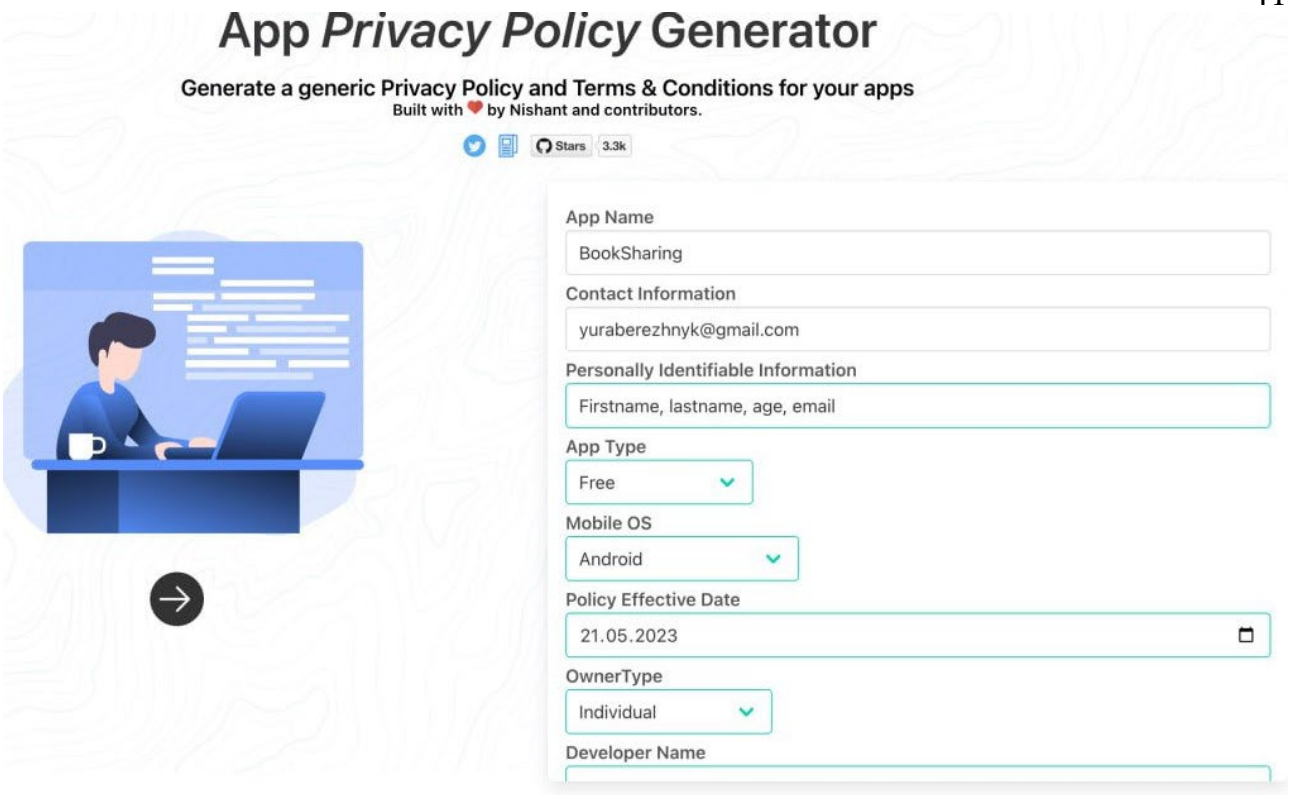


Рисунок 2.10 – Генерація файлів

Заповнивши всю інформацію, нам необхідно завантажити на сайт файл релізу додатку. Для цього нам необхідно згенерувати файл з розширенням «.aab». Щоб мати можливість генерувати файл релізу, насамперед в проекті необхідно згенерувати ключ додатку. Для його генерації виконаємо цю команду в терміналі:

```
keytool -genkeypair -alias mykeyalias -keyalg RSA -keysize 2048 -validity 10000 -keystore keystore.jks
```

Тепер згенеруємо файл релізу використавши команду:

```
flutter build apk release
```

Завантаживши файл релізу та заповнивши всю необхідну інформацію, потрібно зачекати декілька днів, щоб додаток пройшов перевірку.

2.4 – Демонстрація роботи додатку BookShairing

Відкривши додаток вперше, перед користувачем з’являється екран входу в додаток (Див. рисунок 2.11).

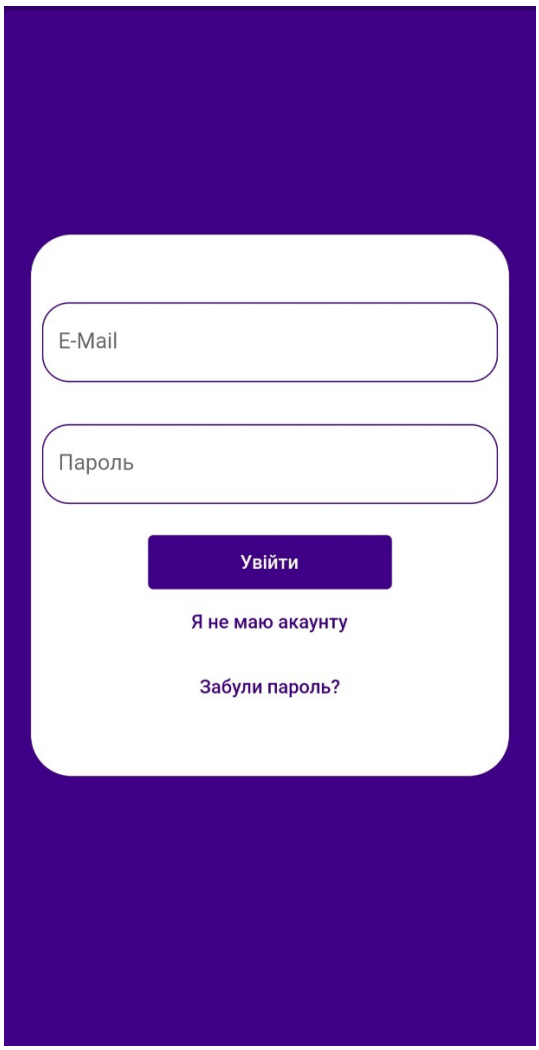


Рисунок 2.11 – Екран входу в додаток

Якщо користувач забув свій пароль для входу, то він може на тиснути на кнопку «Забули пароль», і йому на електронну адресу прийде лист для скидання паролю. Якщо ж користувач незареєстрований, то потрібно натисну «Я не маю акаунту» і відкриться екран для реєстрації(Див. рисунок 2.12).

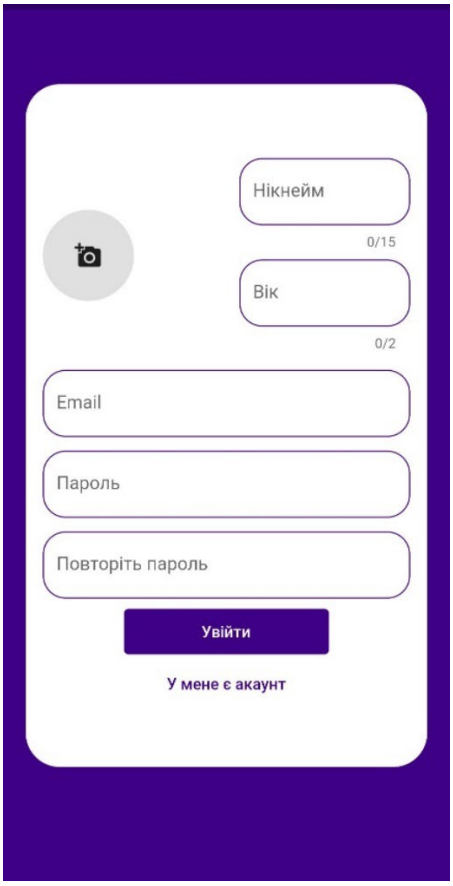


Рисунок 2.12 – Екран реєстрації

Після вводу всієї необхідної інформації, користувачу на електронну адресу приходить лист в якому необхідно перейти по посиланню, щоб підтвердити раніше вказану адресу.(Див. рисунок 2.13).

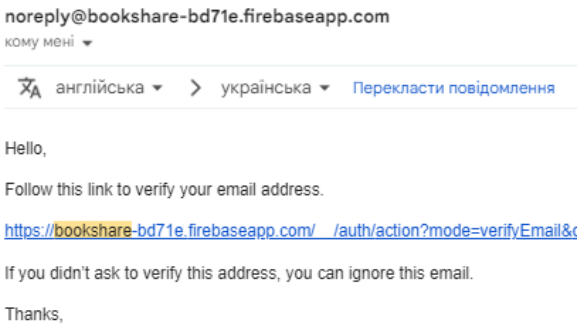


Рисунок 2.13 – Лист верифікації

Після успішної верифікації, на екрані користувача з'являється головна сторінка додатку(Див. рисунок 2.14).

Рисунок 2.14 – Головний екран додатку

Натиснувши на кнопку «+» на екрані з’явиться віджет для додавання в бібліотеку нової книги. (Див. рисунок 2.15).

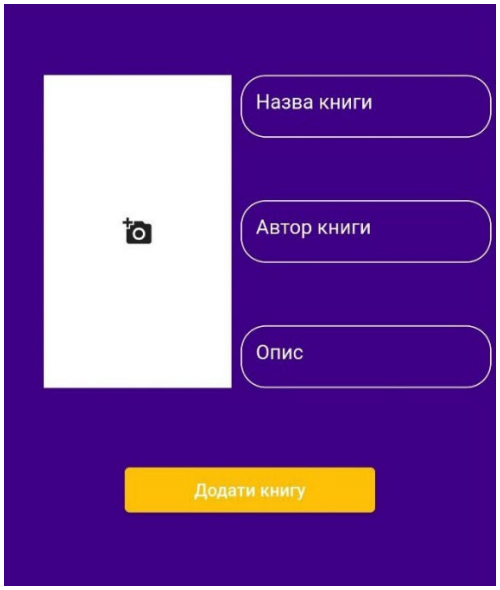


Рисунок 2.15 – Віджет додавання книги

Додавши нові книги, вони з’являться в бібліотеці користувача і будуть відображені на головній сторінці додатку (Див. рисунок 2.16).

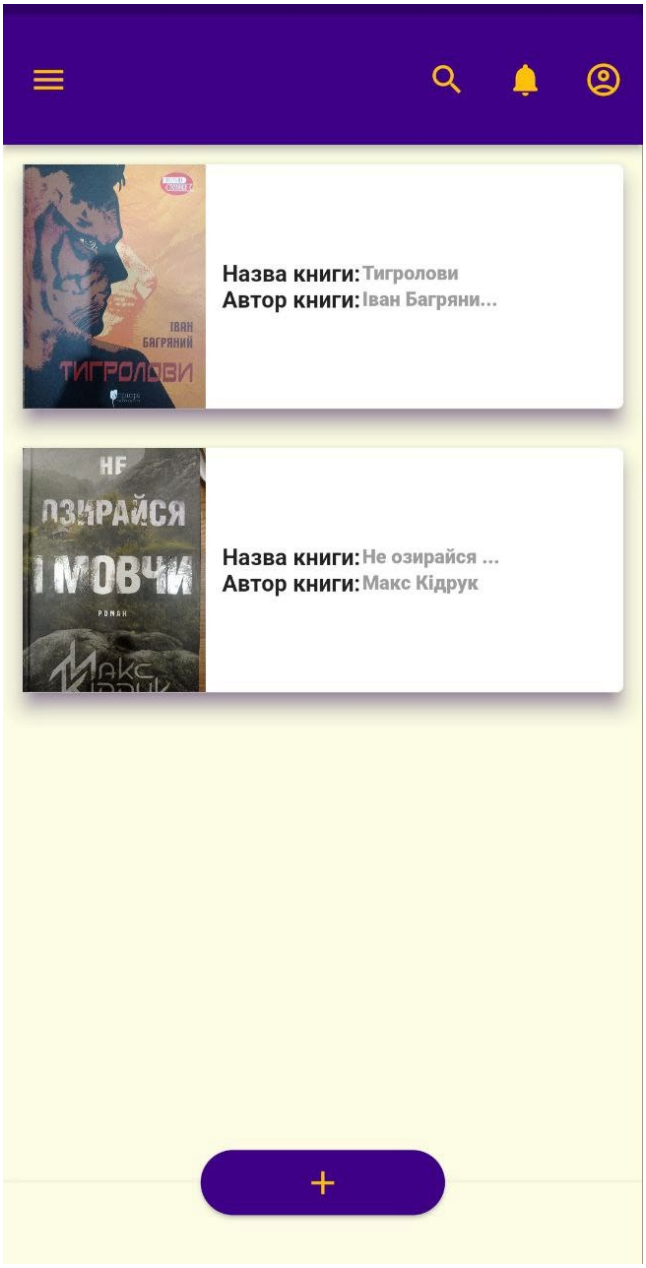


Рисунок 2.16 – Додані в бібліотеку книги

Натиснувши на кнопку в лівому верхньому куті відкриється бокова панель для навігації. Перейдемо в розділ налаштування, щоб переглянути інформацію про додаток.(Див. рисунок 2.17).

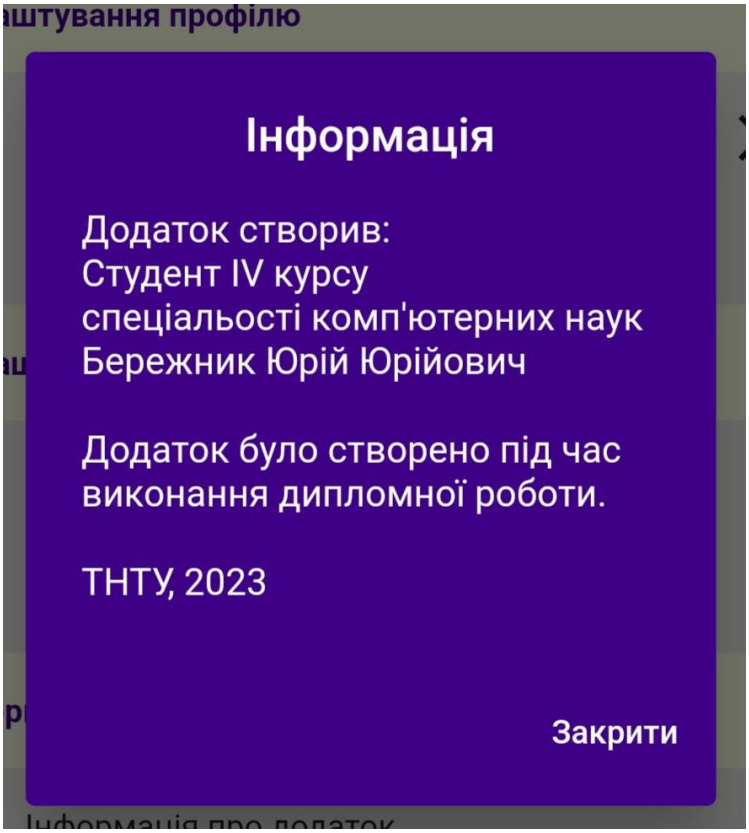


Рисунок 2.17 – Інформація про додаток

Ознайомитись зі всім функціоналом мобільного додатку можна перейшовши за QR-кодом нижче.

Рисунок 2.18 – Посилання на BookShairing в Google PlayMarket

РОЗДІЛ 3 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Діяльність її види та розуміння в безпеці праці

Безпека праці є невід'ємною складовою успішної та продуктивної роботи в будь-якій організації. Вона має велике значення для забезпечення безпеки та здоров'я працівників, а також для підтримки стабільності та розвитку підприємств. Розуміння різних видів діяльності і їх впливу на безпеку праці є ключовим аспектом ефективного управління.

Тема дослідження цього розділу полягає у розкритті різних видів діяльності та їх розумінні в контексті безпеки праці. Метою цього розділу є розкриття особливостей і потенційних ризиків, пов'язаних з різними видами діяльності, а також розуміння заходів, необхідних для забезпечення безпеки праці у цих галузях.

3.1.1 Виробнича діяльність

Виробнича діяльність [22] є однією з найпоширеніших галузей економіки, яка включає процеси виробництва товарів або надання послуг. Аналіз потенційних ризиків, пов'язаних з виробничою діяльністю, є важливим етапом в управлінні безпекою праці. Ризики можуть включати механічні травми, вплив шкідливих речовин, електричні та пожежні небезпеки, стресові ситуації та багато інших факторів, які можуть шкодити здоров'ю та безпеці працівників.

Заходи щодо запобігання небезпекам та підвищення безпеки в виробничій діяльності повинні базуватися на принципах передбачення, запобігання та контролю. Організації повинні проводити оцінку ризиків, розробляти та впроваджувати відповідні стратегії та процедури з метою зниження ризиків. Це може включати впровадження безпечних робочих методів та процедур, надання необхідного оснащення та захисних засобів, навчання працівників правилам

безпеки, системи контролю та моніторингу, а також регулярну оцінку та оновлення заходів безпеки праці.

Виробнича діяльність вимагає постійного удосконалення безпеки та впровадження новітніх технологій та методів, які забезпечують безпеку працівників на робочому місці. Крім того, важливо підтримувати розуміння працівників щодо безпеки та сприяти створенню безпечної робочої культури в організації

3.1.2 Будівельна діяльність та її розуміння в безпеці праці:

Будівельна діяльність [23] є складним та ризикованим процесом, який включає будівництво, реконструкцію та ремонт будівель та споруд. Вона має свої особливості, які потребують спеціальної уваги щодо безпеки праці. Будівельна діяльність включає в себе використання важкого обладнання, роботу на висоті, взаємодію з небезпечними матеріалами та можливість виникнення непередбачуваних ситуацій.

Аналіз потенційних ризиків, пов’язаних з будівельною діяльністю, допомагає виявити небезпеки, які можуть виникнути на будівельному майданчику. Це можуть бути падіння з висоти, травми, пов’язані з важким обладнанням та матеріалами, електричні та пожежні небезпеки, а також ризики, пов’язані зі змінами в робочих умовах та непередбачуваними факторами. Цей аналіз дозволяє розробити стратегії та заходи, спрямовані на запобігання небезпекам та забезпечення безпеки праці на будівельних майданчиках.

Заходи щодо забезпечення безпеки праці на будівельних майданчиках повинні враховувати особливості цієї сфери діяльності. Це можуть бути встановлення правил безпеки, проведення навчання та інструктажу для працівників, надання необхідного захисного спорядження та обладнання, системи контролю та моніторингу, організація робочих місць та процесів з урахуванням безпеки, а також впровадження ефективної системи управління безпекою праці.

3.1.3 Промислова діяльність

Промислова діяльність [24] охоплює широкий спектр галузей, таких як хімічна промисловість, машинобудування, електроенергетика, нафтогазова промисловість та багато інших. Кожна з цих галузей має свої особливості, процеси та обладнання, які можуть впливати на безпеку праці.

Важливим кроком у забезпеченні безпеки праці в промисловій діяльності є виявлення потенційних небезпек та аналіз ризиків, що виникають у цих галузях. Небезпеки можуть включати викиди шкідливих речовин, вибухи, травми, пов’язані зі збільшеною механічною силою, електричні небезпеки та інші ризики, пов’язані з конкретними процесами та умовами праці. Аналіз ризиків дозволяє оцінити ймовірність виникнення цих небезпек та їх потенційний вплив на працівників.

Заходи щодо попередження та управління ризиками в промисловій діяльності є критичними для забезпечення безпеки праці. Це може бути розробка та впровадження стандартів безпеки, встановлення процедур безпечної роботи, надання інструктажу працівникам, контроль і моніторинг за дотриманням правил безпеки, проведення систематичного технічного обслуговування та перевірок обладнання, а також сприяння культурі безпеки праці в організації.

Промислова діяльність має великий вплив на суспільство та економіку, але одночасно може бути пов’язана зі значними ризиками для працівників. Забезпечення безпеки праці в цих галузях є важливим завданням, яке вимагає постійного удосконалення, використання сучасних технологій та врахування специфіки кожної галузі промисловості.

У підсумку, розуміння різних видів діяльності в контексті безпеки праці та впровадження відповідних заходів є необхідним для забезпечення безпеки праці та здоров’я працівників. Це вимагає постійного вдосконалення процесів, навчання та свідомості всіх учасників.

3.2 Загальні вимоги безпеки з охорони праці для користувачів ПК

У сучасному робочому середовищі ПК використовуються все більше і більше. Вони стали необхідним інструментом у багатьох сферах діяльності, включаючи офісну роботу, інформаційні технології, дизайн, освіту та багато іншого. Однак, разом з цим з'являються потенційні ризики, пов'язані з їх використанням [25].

Незабезпечене або неправильне використання ПК може призвести до різних проблем здоров'я та безпеки працівників. Наприклад, тривала робота за комп'ютером може спричинити напругу м'язів і суглобів, що призводить до м'язово-скелетних захворювань. Використання неправильно налаштованого монітора, клавіатури або стільця може спричинити проблеми з поставою спини.

Вимоги безпеки з охорони праці для користувачів ПК відіграють важливу роль у забезпеченні безпеки та здоров'я працівників. Дотримання цих вимог сприяє зменшенню ризиків та забезпечує комфортні умови праці. Далі розглянемо аспекти загальних вимог безпеки з охорони праці для користувачів ПК, що допоможе працівникам працювати в ефективний, комфортний і безпечний спосіб.

Ергономіка робочого місця відіграє важливу роль у забезпеченні безпеки та здоров'я працівників, які використовують ПК Монітор повинен бути розташований на рівні очей працівника. Це допомагає уникнути неправильної постави шиї і зменшує напругу шийних м'язів. Клавіатура та миша повинні бути розташовані таким чином, щоб руки працівника знаходилися в комфортному положенні. Розташування рук повинно бути рівним, а зап'ястя повинні бути вирівняні.

Стіл повинен бути належної висоти, щоб працівнику було зручно працювати. Він повинен бути достатньо просторим для розташування всіх необхідних пристроїв. Стілець повинен мати зручну підтримку для спини і можливість регулювання висоти. Правильна постава і підтримка спини сприяють запобіганню м'язово-скелетних захворювань.

Робоче місце повинно мати належне освітлення, щоб уникнути надмірного або недостатнього освітлення, які можуть спричинити втому очей. Необхідно уникати блисків на екрані монітора, організовуючи місце роботи таким чином, щоб уникнути прямого впливу світла на екран. Регулярні перерви та вправи розтягування можуть допомогти зменшити напругу на м’язах і суглобах. Це особливо важливо при тривалій роботі за ПК.

Безпека електроживлення є одним із найважливіших аспектів охорони праці, оскільки неправильне використання електрообладнання може призвести до серйозних наслідків, таких як пожежі, ураження електричним струмом чи пошкодження обладнання.

Процедури безпечного включення та вимикання електрообладнання повинні бути відомі всім працівникам. Необхідно дотримуватись правил використання зарядних пристроїв, таких як не залишати їх без нагляду під час зарядки, не використовувати пошкоджені пристрої та використовувати лише оригінальні зарядні пристрої.

Електрообладнання повинно бути належним чином заземлене, щоб уникнути уражень електричним струмом. Проводка повинна бути правильно встановлена і має утримуватись у належному стані без пошкоджень.

Перенапруга може виникати внаслідок блискавки, несправностей в електричній мережі чи інших факторів. Застосування стабілізаторів напруги та захисних пристроїв допоможе забезпечити стабільну напругу та захистити обладнання від можливих пошкоджень.

Знання правил безпеки щодо електроживлення та використання відповідних заходів захисту допомагають зменшити ризики пожеж, уражень електричним струмом та пошкодження електрообладнання, забезпечуючи безпеку працівників та надійну роботу обладнання.

Таким чином, виконання загальних вимог безпеки з охорони праці для користувачів ПК є необхідним і ефективним заходом, який допомагає підвищити якість працівника, забезпечити його здоров’я та сприяє досягненню успіху в роботі.

ВИСНОВКИ

Результатом кваліфікаційної роботи є практично створений мобільний додаток для обміну книгами «BookShairing». При розробці даного мобільного додатку було використано фреймворк Flutter та мову програмування Dart, а також платформу Firebase.

В першому розділі кваліфікаційної роботи:

- розглянуто існуючі альтернативи;
- сформовано вимого до мобільного додатку;
- побудовано діаграми використання;
- розроблено прототип мобільного додатку.

В другому розділі кваліфікаційної роботи:

- розглянуто фреймворк Flutter та платформу Firebase;
- показано структуру проекту;
- запропоновано приклади з процесу розробки мобільного додатку;
- наведено інструкції щодо релізу мобільного додатку;
- продемонстровано роботу додатку,

У розділі «Безпека життєдіяльності, основи охорони праці » висвітлено такі питання:

- Діяльність її види та розуміння в безпеці праці.
- Загальні вимоги з безпеки праці для користувачів ПК;

ПЕРЕЛІК ДЖЕРЕЛ

1. Flutter [Електронний ресурс] // Режим доступу: https://docs.flutter.dev/resources/faq

2. Сайт BooksAround [Електронний ресурс] // Режим доступу: https://www.booksaround.net/?lang=en

3. Інтерв'ю редакції Wonder [Електронний ресурс] // Режим доступу: https://www.wonderzine.com.ua/wonderzine/life/life/7353-do-zakladok-zastosunok

4. Сайт BookSwap [Електронний ресурс] // Режим доступу: https://bookswap.co.uk/

5. Сайт BookCrosing [Електронний ресурс] // Режим доступу: https://www.bookcrossing.com/

6. Навіщо потрібно прототипування [Електронний ресурс] // Режим доступу: https://dou.ua/lenta/articles/prototyping-for-managers/

7. Прототип мобільного додатку BookShairing [Електронний ресурс] // Режим доступу: http://surl.li/gdjyc

8. A comparison between the top of mobile app development frameworks [Електронний ресурс] // Режим доступу: https://www.fingent.com/blog/react-native-flutter-ionic-xamarin-a-comparison-between-the-top-mobile-app-development-frameworks/

9. Порівняння використання фреймворків за останні 4 роки [Електронний ресурс] // Режим доступу: https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/

10. Порівняння фреймворків за 2021 рік [Електронний ресурс] // Режим доступу: https://strapi.io/blog/comparing-mobile-development-platforms-in-2021

11. Вибір найкращого фреймворку за 2020 рік [Електронний ресурс] // Режим доступу: https://radixweb.com/blog/cross-platform-app-development-frameworks

12. Сайт Firebase [Електронний ресурс] // Режим доступу: https://firebase.google.com/docs

13. Архітектура Flutter [Електронний ресурс] // Режим доступу: https://docs.flutter.dev/resources/architectural-overview

14. Детальна інформація про Firebase [Електронний ресурс] // Режим доступу: https://appmaster.io/blog/what-is-firebase

15. Firestore tutorial [Електронний ресурс] // Режим доступу: https://www.kodeco.com/26435435-firestore-tutorial-for-flutter-getting-started

16. Модель трьохрівневої архітектури [Електронний ресурс] // Режим доступу: https://www.techopedia.com/definition/24649/three-tier-architecture

17. Null safety [Електронний ресурс] // Режим доступу: https://dart.dev/null-safety

18. Код проекту BookShairing [Електронний ресурс] // Режим доступу: https://github.com/YuraYB3/BookSharing

19. Build and release [Електронний ресурс] // Режим доступу: https://docs.flutter.dev/deployment/android

20. App Privacy Policy Generator [Електронний ресурс] // Режим доступу: https://app-privacy-policy-generator.firebaseapp.com/

21. BookShairing [Електронний ресурс] // Режим доступу: https://play.google.com/store/apps/details?id=com.yuraberezhnyk.booksshare&hl=uk&gl=US

22. Виробнича діяльність [Електронний ресурс] // Режим доступу: https://osvita.ua/vnz/reports/econom_pidpr/20893/

23. Будівельна діяльність [Електронний ресурс] // Режим доступу: https://studfile.net/preview/3009462/page:14/

24. Промислова діяльність [Електронний ресурс] // Режим доступу: https://ukrstat.gov.ua/operativ/pro_stat/Prosto/prom/Prom_vur.pdf

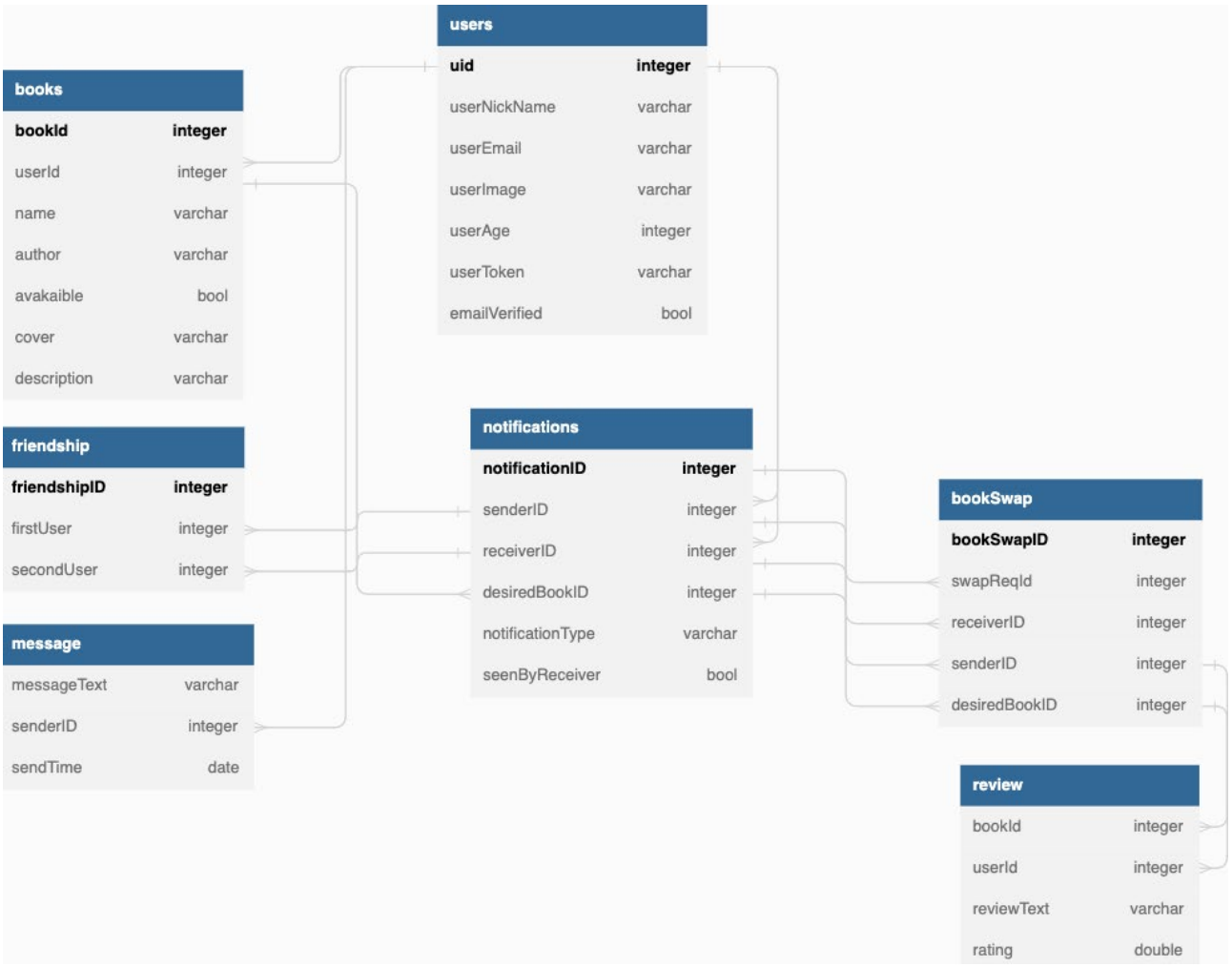
25. Правила користування ПК [Електронний ресурс] // Режим доступу: https://pedcollege.kiev.ua/index.php/77-robota-koledzhu/okhorona-pratsi/589-pravyla-bezpechnoi-roboty-na-kompiuteri

26. Сайт для побудови діаграм [Електронний ресурс] // Режим доступу: https://app.diagrams.net/

ДОДАТКИ

Додаток А

Зв’язок між таблицями



Додаток Б

Функція onNewMessage

```
exports.onNewMessage = functions.firestore
    .document("message/{friendshipID}/dialogue/{messageID}")
    .onCreate(async (snapshot, context) => {
      const mes = snapshot.data();
      const fID = context.params.friendshipID;
      console.log(`New message  ${fID}`);
      console.log(`Message details: ${JSON.stringify(mes)}`);
      try {
        const fRef =
        ad.firestore().collection("friendship").doc(fID);
        const friendshipDoc = await fRef.get();
        const fsh = friendshipDoc.data();
        if (!fsh) {
          console.log(`Friendship with ID ${fID} does not exist`);
          return;
        }
        const rID = fsh.user1_ID === mes.senderID ? fsh.user2_ID :
fsh.user1_ID;
        console.log(`Recipient ID: ${rID}`);
        const                          sRef                          =
ad.firestore().collection("users").doc(mes.senderID);
        const senderDoc = await sRef.get();
        const sendData = senderDoc.data();
        const rRef = ad.firestore().collection("users").doc(rID);
        const recipientDoc = await rRef.get();
        const recp = recipientDoc.data();
        if (!recp) {
          console.log(`Recipient with ID ${rID} does not exist`);
          return;
        }
        const payload = {
          notification: {
            title: `Нове повідомлення від ${sendData.name}`,
            body: mes.message,
            clickAction: "FLUTTER_NOTIFICATION_CLICK",
          },
          data: {
            screen: "messages",
            click_action: "FLUTTER_NOTIFICATION_CLICK",
            message_id: snapshot.id,
            friend_id: fID, },};
        const              resp              =              await
ad.messaging().sendToDevice(recp.userToken, payload);
        console.log(`Message   sent   to   device   with   token
${recp["userToken"]}`);
        console.log(`Messaging            API            response:
${JSON.stringify(resp)}`);
      } catch (error) {
        console.log(`Error sending notification: ${error}`);
      }});
```