МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Львівський національний університет імені Івана Франка

Факультет електроніки та комп'ютерних технологій

Кафедра радіоелектронних і комп'ютерних систем

Допустили до захисту

Завідувач кафедри

_____ проф. Оленич І. Б.

«___» _____20__ р.

Кваліфікаційна робота

Бакалавр

(освітній ступінь)

Створення електронної бібліотеки для мобільних пристроїв з допомогою фреймворку Dart/Flutter

Виконав:
студент IV курсу групи ФЕП-41
спеціальності 121 – Інженерія програмного забезпечення
_____ Лотошинський Андрій Романович

Науковий керівник:
_____ доц. Соколовський Б. С.
«____» _____20__ р.

Рецензент:
_____ проф. Юзевич В. М.

Львів 2023

АНОТАЦІЯ

Дана дипломна робота присвячена створенню електронної бібліотеки книг для мобільних пристроїв. На основі аналізу позитивних та негативних сторін існуючих електронних бібліотек була поставлена мета розробити зручну для користувача мобільну бібліотеку, що володіє багатьма функціональними можливостями.

Для створення мобільної бібліотеки був використаний фреймворк Flutter, а також Google Books API. Розроблено користувацький інтерфейс, який дозволяє зручно створювати та переглядати колекції книг.

Розроблений додаток протестовано на мобільних пристроях та емуляторах, які працюють на платформах Android та iOS.

ABSTRACT

This thesis is devoted to the creation of a digital library of books for mobile devices. Based on the analysis of the positive and negative aspects of existing digital libraries, the goal was to develop a user-friendly mobile library with many functionalities.

The Flutter framework and Google Books API were used to create the mobile library. A user interface was developed that allows you to easily create and view book collections.

The developed application has been tested on mobile devices running on Android and iOS platforms.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

UI – користувацький інтерфейс (від англійського “User interface”)

API – прикладний програмний інтерфейс (від англійського “Application programming interface”

JSON – Нотація об'єктів JavaScript (від англійського JavaScript Object Notation)

Віджет – компоненти для побудову інтерфейсу у Flutter

ВСТУП

Актуальність роботи: З появою сучасних мобільних пристроїв, які стали невід'ємною частиною нашого повсякденного життя, зросла потреба у зручних та функціональних мобільних додатках. Одним з таких додатків є мобільна бібліотека, яка дозволяє користувачам мати доступ до своєї колекції книг прямо на своєму смартфоні або планшеті. Враховуючи цю потребу, розробка ефективного та привабливого мобільного додатку для книжкової колекції стає актуальною задачею.

Мета роботи: Метою даної дипломної роботи є розробка мобільної бібліотеки з книгами з використанням фреймворку Flutter. Робота спрямована на створення функціонального додатку, який забезпечує зручне та ефективне управління книжковою колекцією користувача на мобільних пристроях.

Об'єкт дослідження: Об'єктом дослідження є розробка мобільної бібліотеки з книгами з використанням фреймворку Flutter.

Предмет дослідження: Предметом дослідження є функціональні можливості та інтерфейс розробленої мобільної бібліотеки, її удосконалення та оптимізація для забезпечення задоволення потреб користувачів.

Методи дослідження та апаратура: У процесі роботи використовувалися методи аналізу, проектування та програмування. Для розробки мобільної бібліотеки використовувався фреймворк Flutter, який забезпечує швидку та ефективну розробку мобільних додатків для платформ Android та iOS. Для відналагодження програми використовувались емулятори мобільних пристроїв,та реальний девайс на ОС Android.

Елементи новизни та галузь застосування результатів: Новизною роботи є поєднання функціональності мобільної бібліотеки з використанням потужного фреймворку Flutter. Розроблений додаток може бути використаний як інструмент для організації колекції книг на мобільних пристроях. Також, результати роботи можуть бути використані в галузі розробки мобільних додатків та бібліотек.

Прогнозні припущення про розвиток об'єкту розроблення: Прогнозується, що розроблена мобільна бібліотека може бути подальше розширена та вдосконалена. Можливі напрями розвитку включають додаткові функціональні можливості, такі як синхронізація з хмарними сервісами для резервного копіювання даних та спільного доступу, можливість ділитися книгами з іншими користувачами, тощо. Також, можливим напрямком розвитку є адаптація додатку для інших платформ, таких як планшети та веб-браузери, для ще ширшої доступності та зручного використання користувачами.

РОЗДІЛ 1. АНАЛІЗ СТАНУ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Створення мобільних додатків

Створення мобільних додатків пройшло значний шлях еволюції протягом останніх десятиліть. З першими мобільними пристроями на базі старих операційних систем, розробка додатків була складною і обмеженою. Проте з введенням смартфонів та розширенням їх функціоналу виникла потреба в більш розширених та потужних рішеннях для розробки мобільних додатків.

З'явилися перші мобільні платформи, такі як IOS та Android, які забезпечували фреймворки та інструменти для розробки додатків, пристосованих до їхніх операційних систем. Розробники мали можливість створювати нативні додатки, які повністю використовували можливості цих платформ.

Однак, разом з розширенням ринку мобільних пристроїв і появою нових платформ, виникла потреба в розробці кросплатформних додатків. Кросплатформність дозволяє розробникам створювати додатки, які можуть працювати на різних платформах з використанням спільного коду.

У цьому контексті Flutter виступає як чудове рішення для кросплатформної розробки мобільних додатків. Flutter – це відкритий фреймворк, розроблений компанією Google, який дозволяє створювати красиві та швидкі мобільні додатки для платформ Android та iOS з використанням одного і того ж коду.

Основні переваги Flutter полягають у його швидкості, продуктивності та простоті використання. Він пропонує багатий набір готових компонентів та інтерфейсів, які дозволяють створювати привабливі та інтуїтивно зрозумілі додатки. Крім того, Flutter має гарну підтримку з боку спільноти розробників та активно розвивається.

Завдяки Flutter розробники можуть ефективно використовувати свій час та зусилля, розробляючи додатки для обох платформ одночасно. Це дозволяє зменшити витрати на розробку та підтримку додатків, а також прискорює процес впровадження змін та оновлень.

Усе це робить Flutter чудовим вибором для розробників, які прагнуть створювати високоякісні, кросплатформні мобільні додатки з ефективним

використанням ресурсів. За останні роки Flutter набув значної популярності та зарекомендував себе як сучасне та потужне рішення для розробки мобільних додатків.

Перш за все, існує зростаюча потреба в зручних та функціональних мобільних додатках для книжкової колекції. З популярністю сучасних мобільних пристроїв, люди все частіше використовують їх для читання книг та управління своєю колекцією. Однак, існуючі додатки не завжди задовольняють потреби користувачів щодо зручного організування та управління книгами.

1.2 Огляд існуючих рішень

Деякі існуючі мобільні додатки для книжкової колекції мають обмежену функціональність або не надають достатньо зручного інтерфейсу. Вони можуть бути обмежені в можливостях сортування, пошуку або відображенні інформації про книги. Це створює необхідність в розробці нового додатку, який задовольняв би потреби користувачів у функціональності та зручності.

Огляд існуючих програмних рішень в області мобільних бібліотек з книгами демонструє різноманітні підходи та функціональні можливості. Нижче наведено огляд деяких існуючих програмних рішень.

Goodreads є одним з найпопулярніших веб-орієнтованих сервісів для організації та відстеження книжкових колекцій. Додаток Goodreads надає користувачам можливість додавати книги до своєї колекції, вести статистику читання, робити оцінки та написати відгуки про прочитані книги. Ось деякі переваги та недоліки Goodreads:

Переваги:

- Велика база даних: Goodreads має вражаючу кількість книжок у своїй базі даних, що дозволяє користувачам легко знайти та додати бажані книги до своєї колекції.

- Рейтинги та відгуки: Користувачі можуть переглядати рейтинги та відгуки інших читачів, що допомагає при виборі нових книг для читання.
- Рекомендації: Goodreads пропонує персоналізовані рекомендації на основі переданої інформації про читання користувача, що допомагає знайти нові цікаві книги.
- Соціальний аспект: Додаток має соціальні функції, які дозволяють користувачам спілкуватися з іншими читачами, ділитися враженнями та рекомендаціями.

Недоліки:

- Веб-орієнтованість: Goodreads перш за все є веб-платформою, і мобільний додаток має обмежену функціональність порівняно з веб-версією.
- UI: Користувачі відзначають, що UI у Goodreads може бути складним для навігації та використання, особливо для новачків.
- Відсутність офлайн-режиму: Додаток не підтримує офлайн-режим, що обмежує доступ до книжок без інтернет-з'єднання.
- Платформові обмеження: Goodreads доступний переважно для платформ Android та iOS, що може обмежити доступність для користувачів інших мобільних пристроїв.

Незважаючи на деякі недоліки, Goodreads залишається популярним додатком для організації книжкових колекцій та обміну рекомендаціями між читачами.

Libib є іншим популярним програмним рішенням для каталогізації та організації книжкових колекцій. Цей додаток надає користувачам зручні інструменти для додавання, відстеження та управління книгами. Ось огляд переваг та недоліків Libib:

Переваги:

- Множина типів колекцій: Libib дозволяє створювати різні типи колекцій, такі як книги, фільми, музика тощо, що робить його універсальним інструментом для організації різних типів зібрань.
- Синхронізація в хмарі: Додаток пропонує синхронізацію даних у хмарі, що дозволяє отримувати доступ до колекцій з різних пристроїв і забезпечує збереження даних навіть при заміні пристрою.
- Сканування штрих-кодів: Libib підтримує функцію сканування штрих-кодів, що спрощує додавання книг до колекції шляхом автоматичного розпізнавання.
- Каталогізація та організація: Додаток дозволяє впорядковувати книги за різними категоріями, додавати мітки, створювати список бажань та відстежувати статус прочитання.

Недоліки:

- Обмежена безкоштовна версія: Деякі функції та обмеження доступні лише в платній версії Libib, що може бути недоцільним для деяких користувачів.
- Обмежена спільнота користувачів: У порівнянні з іншими додатками, Libib має меншу спільноту користувачів, що може обмежити обмін рекомендаціями та взаємодію між користувачами.
- UI та дизайн: Деякі користувачі вказують на те, що UI у Libib може бути менш інтуїтивним та менш привабливим у порівнянні з деякими конкуруючими додатками.

Незважаючи на деякі недоліки, Libib залишається популярним вибором для організації книжкових колекцій зі своїми зручними функціями та можливостями управління.

Book Catalogue також є одним із програмних рішень для каталогізації та управління книжковими колекціями. Цей додаток надає розширені можливості для користувачів, що дозволяє їм зручно відстежувати, додавати та керувати своїми книгами. Ось огляд переваг та недоліків Book Catalogue:

Переваги:

- Додавання книг за ISBN: Book Catalogue дозволяє користувачам швидко додавати книги до своєї колекції, просто скануючи штрих-код або вводячи ISBN.
- Відстеження прочитаних книг: Додаток надає можливість позначати книги, які ви прочитали, та вести статистику свого прогресу читання.
- Відстеження позичених книг: Ви можете відстежувати, кому ви позичили свої книги, і нагадувати собі про повернення.
- Категоризація та сортування: Додаток дозволяє вам створювати категорії, додавати теги та застосовувати різні фільтри для зручної організації та пошуку книг.

Недоліки:

- Обмежений доступ: Book Catalogue доступний лише для платформи Android, що може обмежити доступність для користувачів iOS та інших мобільних пристроїв.
- Обмежені можливості синхронізації: Додаток не має вбудованої функції синхронізації у хмарі, що може ускладнити доступ до даних з різних пристроїв або при заміні пристрою.
- Обмежений соціальний взаємодія: У порівнянні з деякими іншими додатками, Book Catalogue може мати обмежені можливості обміну рекомендаціями та взаємодії з іншими користувачами.
- Також сюди можна віднести і не зручний та не привабливий UI

Незважаючи на деякі недоліки, Book Catalogue є зручним інструментом для організації книжкових колекцій та відстеження читацького прогресу. Він пропонує розширені функції та можливості, які забезпечують ефективне управління вашими книгами.

Shelfie – це додаток, призначений для каталогізації книжкових полиць та керування книжковими колекціями. Він пропонує ряд функцій та можливостей для

зручного організування та відстеження книг. Ось огляд переваг та недоліків додатку Shelfie:

Переваги:

- Сканування штрих-кодів: Shelfie дозволяє користувачам швидко додавати книги до своєї колекції, просто скануючи штрих-код на обкладинці книги.
- Автоматичне заповнення даних: Додаток автоматично отримує та заповнює інформацію про книги, включаючи обкладинки, авторів та описи, що забезпечує швидку та точну каталогізацію.
- Організація полиць та категорій: Ви можете створювати власні полиці та категорії для каталогізації книг за темами, жанрами або будь-якими іншими параметрами.
- Поділ книжок з друзями: Shelfie дозволяє легко позичати книги друзям або робити обмін, зберігаючи записи про позичені книги та нагадуючи про повернення.

Недоліки:

- Доступність платформ: Shelfie доступний лише для платформи iOS, що може обмежити доступність для користувачів Android та інших мобільних пристроїв.
- Відсутність синхронізації: Додаток не має вбудованої функції синхронізації між різними пристроями, що може ускладнити доступ до даних при заміні або втраті пристрою.
- Обмежений соціальний аспект: Shelfie має обмежені можливості спілкування та обміну рекомендаціями з іншими користувачами.

Незважаючи на деякі недоліки, Shelfie є зручним інструментом для каталогізації та управління книжковими полицями, забезпечуючи користувачам зручну організацію та відстеження їх книжок.

Всі ці програмні рішення мають свої переваги та недоліки. Проте, існує потреба в новому мобільному додатку, який би поєднував широкий функціонал,

зручний інтерфейс та кросплатформову підтримку. Розробка мобільної бібліотеки з використанням фреймворку Flutter має потенціал задовольнити цю потребу та надати користувачам новий рівень зручності та функціональності управління їх книжковими колекціями.

1.3 Аналіз літератури для вивчення технології

При підготовці до бакалаврської роботи було підібрано та проаналізовано наступну літературу:

"Flutter in Action" Адама Сміта – це книга, яка надає змістовний посібник з розробки мобільних додатків з використанням фреймворку Flutter. Книга охоплює різні аспекти розробки на Flutter, включаючи основи Flutter, побудову користувацьких інтерфейсів, роботу з віджетами, керування станом та інтеграцію зі службами на боці сервера.

Ось детальний аналіз деяких ключових аспектів, які охоплюються в "Flutter in Action":

- Вступ до Flutter: Книга починається зі знайомства з Flutter та його перевагами як фреймворку для розробки кросплатформових додатків. Вона пояснює архітектуру Flutter, підхід до побудови інтерфейсу на основі віджетів та мову програмування Dart, яка використовується для розробки на Flutter.
- Побудова користувацьких інтерфейсів: "Flutter in Action" розглядає створення користувацьких інтерфейсів з використанням Flutter, охоплюючи такі теми, як віджети компонування, текст і зображення, кнопки, поля введення, списки та навігацію. Вона досліджує дерево віджетів Flutter, демонструючи, як створювати відзивчиві та динамічні інтерфейси користувача.
- Керування станом: Книга досліджує різні підходи до керування станом у додатках на Flutter. Вона охоплює вбудовані опції керування станом, такі як StatefulWidget та InheritedWidget, а також популярні зовнішні бібліотеки для керування станом, такі як Provider, Bloc та MobX. Вона пояснює переваги та компроміси кожного підходу.

- Анімація та жести: Можливості анімації та обробки жестів у Flutter детально пояснюються в книзі. Вона охоплює анімацію з використанням класів AnimationController та Tween, неявну та явну анімацію, а також складні послідовності анімацій. Також обговорюється визнання жестів та обробка взаємодій користувача.
- Мережеве взаємодія та збереження даних: "Flutter in Action" надає вказівки з інтеграції додатків на Flutter зі службами на боці сервера та базами даних. Розглядаються такі теми, як здійснення HTTP-запитів, обробка даних у форматі JSON, робота з REST API та використання популярних бібліотек, таких як Dio та Retrofit. Досліджуються також можливості локального збереження даних, такі як SQLite та спільні налаштування.
- Тестування та налагодження: Книга акцентує увагу на важливості тестування та налагодження додатків на Flutter. Вона охоплює різні техніки тестування, включаючи тестування віджетів, інтеграційне тестування та модульне тестування. Також обговорюються інструменти та техніки налагодження для виявлення та усунення проблем у додатках на Flutter.
- Розгортання та оптимізація продуктивності: Книга завершується вказівками з розгортання додатків на Flutter на різних платформах, включаючи iOS та Android. Розглядаються такі теми, як затемнення коду, оптимізація продуктивності та використання інструментів профілювання Flutter для аналізу та покращення продуктивності додатків.

"Flutter in Action" високо оцінюється як практичний джерело для вивчення розробки на Flutter. Вона надає чіткі пояснення, зразки коду та приклади з реального життя, щоб допомогти читачам осмислити концепції та застосувати їх у власних проектах. Чи ви новачок чи досвідчений розробник, ця книга пропонує цінні інсайти для побудови високоякісних додатків на Flutter.

"Flutter – програмування мобільних додатків на Dart" Джеремі Лі – є книгою, написаною Джеремі Лі, яка надає вичерпний огляд розробки мобільних додатків з використанням фреймворку Flutter та мови програмування Dart. Книга

пропонує чітке пояснення концепцій та практичних прикладів для допомоги читачам засвоїти основи розробки на Flutter.

Ось детальний аналіз деяких ключових аспектів, які охоплюються в книзі "Flutter – програмування мобільних додатків на Dart":

- Вступ до Flutter та Dart: Книга починається зі знайомства з фреймворком Flutter та мовою програмування Dart. Вона пояснює переваги використання Flutter для розробки кросплатформових мобільних додатків та основи мови Dart, яка використовується для програмування на Flutter.
- Основи розробки на Flutter: Книга детально розглядає основні концепції розробки на Flutter, такі як віджети, компонування інтерфейсу, макети, навігація та взаємодія з користувачем. Вона надає приклади коду та практичні завдання для закріплення отриманої інформації.
- Робота зі станом та керування станом: Книга описує різні підходи до керування станом в Flutter та наводить приклади використання таких бібліотек, як Provider та MobX. Вона пояснює, як ефективно керувати станом додатка та підтримувати його синхронізацію з інтерфейсом користувача.
- Робота зі списками та базами даних: Книга досліджує розробку списків та використання баз даних у Flutter. Вона надає приклади використання різних типів списків та демонструє способи підключення до баз даних, таких як SQLite, та взаємодії з ними.
- Анімація та взаємодія з користувачем: Книга пропонує детальний огляд можливостей анімації та взаємодії з користувачем в Flutter. Вона пояснює, як створювати анімації, реагувати на жести користувача та забезпечувати зручну взаємодію з додатком.
- Тестування та налагодження: Книга акцентує увагу на важливості тестування та налагодження Flutter додатків. Вона описує різні підходи до тестування Flutter додатків та надає практичні поради щодо налагодження та виявлення помилок.

- Розгортання додатків: Книга завершується поясненням процесу розгортання Flutter додатків на різних платформах, включаючи Android та iOS. Вона надає вказівки щодо підготовки додатків до релізу та налагодження проблем, що виникають під час розгортання.

"Flutter – програмування мобільних додатків на Dart" є корисним джерелом інформації для розробників, які бажають оволодіти розробкою на Flutter та мовою Dart. Вона надає чітке викладення концепцій та практичних прикладів для розуміння та застосування цих технологій у реальних проектах.

РОЗДІЛ 2. ОПИС ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

2.1 Мова програмування Dart

Мова програмування Dart є однією з основних мов, використовуваних для розробки додатків на платформі Flutter. Вона була створена компанією Google з метою створення ефективної та масштабованої мови для веб та мобільних додатків. Ось деякі ключові аспекти мови програмування Dart:

- Синтаксис та структура: Dart має синтаксис, схожий на інші об'єктно-орієнтовані мови програмування, такі як Java або C#. Він використовує класи, об'єкти, функції, змінні, оператори та інші структури для побудови програм.
- Типи даних та типізація: Dart підтримує строгу типізацію, що означає, що кожна змінна та функція має визначений тип даних. Він пропонує різноманітні типи даних, такі як цілі числа, дійсні числа, рядки, булеві значення, списки, мапи та інші.
- Об'єктно-орієнтоване програмування: Dart підтримує об'єктно-орієнтовану парадигму програмування. Він дозволяє визначати класи, створювати об'єкти, використовувати наслідування, поліморфізм та інші концепції ООП. Важливим аспектом є можливість розширювати класи за допомогою розширень та міксинів.
- Асинхронне програмування: Dart пропонує підтримку асинхронного програмування, що дозволяє ефективно виконувати операції вводу-виводу та мережеві запити без блокування основного потоку виконання програми. Для цього використовуються ключові слова async та await, а також класи Future та Stream. Різницю в роботі синхроного та асинхроного програмування зображено на рисунку 2.1.1.
- Колекції та ітератори: Dart має багатий набір вбудованих класів для роботи з колекціями даних, таких як списки, мапи, набори та черги. Він надає різноманітні методи та операції для роботи з колекціями, включаючи фільтрацію, сортування, мапування та зведення.

– Генератори коду: Dart має вбудовану підтримку для генерації коду за допомогою анотацій та метаданих. Це дозволяє розробникам автоматизувати певні аспекти розробки, такі як серіалізація даних, рефлексія та інші.
– Велика екосистема та підтримка: Dart має широку підтримку з боку спільноти розробників та активну екосистему зі значною кількістю пакетів та бібліотек, які полегшують розробку додатків на Flutter.

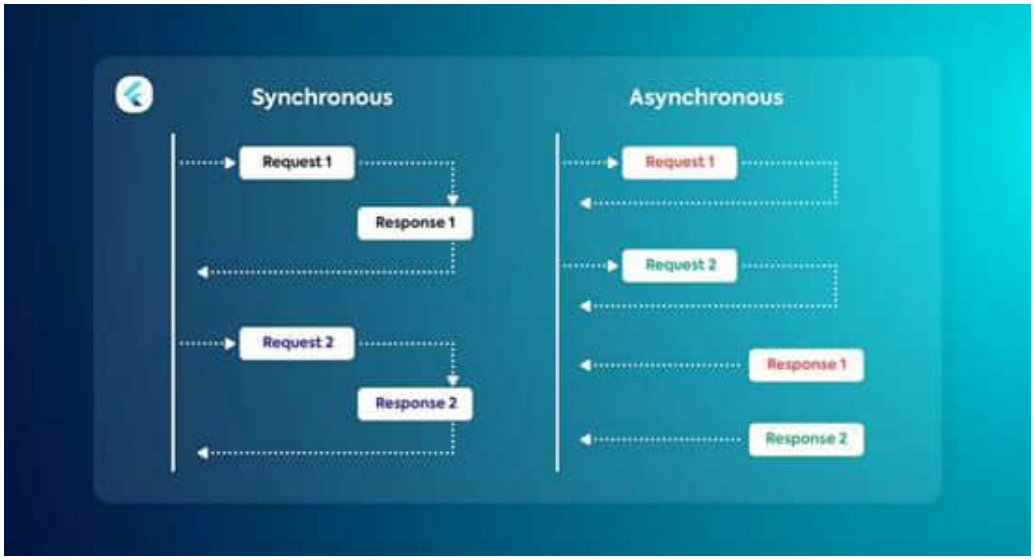

Рис. 2.1. – ілюстрація різниці синхроного та асинхроного програмування

Мова програмування Dart є потужним інструментом для розробки мобільних та веб-додатків. Вона поєднує простоту та елегантність синтаксису з широким спектром можливостей для ефективного створення як невеликих, так і великих проектів.

2.2 Фреймворк Flutter

Flutter є відкритою та прогресивною платформою для розробки мобільних, веб- та настільних додатків. Розроблений компанією Google, він здобуває все більшу популярність у розробників завдяки своїм численним перевагам. Однак, наявні й деякі недоліки. Нижче наведені головні переваги та недоліки фреймворку:

Переваги Flutter:

- Кросплатформеність: Flutter дозволяє розробляти додатки для різних платформ, таких як Android, iOS, веб та навіть настільні операційні системи. Розробники можуть використовувати один кодову базу для створення додатків, що забезпечує ефективність та економію часу.
- Швидкодія: Flutter використовує власний движок відображення, що дозволяє створювати високоефективні додатки з плавною анімацією та швидким реагуванням. Це робить його ідеальним вибором для розробки додатків з багатою взаємодією з користувачем.
- Хороша зручність для розробників: Flutter надає багатий набір готових компонентів та віджетів, які полегшують розробку інтерфейсу користувача. Він також має гарну документацію, активну спільноту та велику кількість ресурсів, що допомагають розробникам швидко освоїти цю платформу.
- Гарний дизайн: За допомогою Flutter розробники можуть створювати додатки зі стильним та сучасним дизайном. Він має вбудовану підтримку для матеріального дизайну, що допомагає створювати додатки, що виглядають привабливо та професійно.
- Гармонійна інтеграція: Flutter добре поєднується з іншими технологіями та фреймворками. Він може використовуватись разом зі специфічними для платформи API, такими як камера, геолокація та інші. Також можна використовувати пакети Dart для роботи з серверною логікою та базами даних.

Недоліки Flutter:

- Розмір додатків: Додатки, розроблені на Flutter, можуть мати трохи більший розмір порівняно з додатками, написаними на нативних мовах. Це через наявність власного двигуна в додатку.
- Недостатня зрілість: Хоча Flutter швидко розвивається та набуває популярності, деякі розробники вважають, що він ще не досяг повної зрілості порівняно з іншими відомими фреймворками.

2.2.1 Архітектура Flutter

Основна ідея розробки користувацького інтерфейсу з використанням Flutter полягає в тому, що весь інтерфейс будується шляхом написання коду. У Flutter немає можливості перетягування та розміщення елементів інтерфейсу на екрані за допомогою миші, замість цього створюється код.

2.2.2 Віджети

У Flutter кожен елемент інтерфейсу є віджетом. Це може бути текст, кнопка, іконка або поле для введення тексту. Усі ці елементи представлені у вигляді віджетів. Наприклад, верхня панель інтерфейсу є віджетом, який містить у собі інші менші віджети. Назви, поля для вводу, кнопки - все це також віджети. Усі компоненти додатку в Flutter побудовані з віджетів, а сторінка додатку також є віджетом, який містить інші віджети.

Розглянемо приклад (рис. 2.2):

Рис. 2.2 – віджети

Верхня панель – це віджет, що містить у собі інші менші віджети (дерево віджетів). Назви, поля для вводу, поле для прикріплення документів, кнопка для відправлення – усе це віджети. Абсолютно весь додаток буде побудований з віджетів, навіть уся сторінка є віджетом, та і весь додаток «загорнутий» у віджет.

Віджет – це шматок коду, який виконує певні інструкції для відображення елемента на екрані користувача. За допомогою віджетів ми будуємо дерево віджетів, де кореневим віджетом є наш додаток, а дочірні віджети відповідають за відображення необхідної сторінки.

2.2.3 Компіляція Flutter/Dart у нативний додаток

Код, написаний на мові Dart, використовує Flutter фреймворк, який включає в себе набір вбудованих віджетів, а також дозволяє створювати власні. Цей код потрібно скомпілювати для створення додатків на платформах Android та iOS. Flutter SDK виконує компіляцію коду Dart в нативний код для кожної з цих платформ. Фреймворк Flutter не використовує платформені примітиви. Наприклад, коли ми додаємо кнопку, Flutter не створює нативний еквівалент кнопки для Android та iOS. Замість цього, Flutter має власний механізм, який керує відображенням кожного пікселя на екрані для користувача. Це дозволяє Flutter максимально контролювати вигляд інтерфейсу додатку.

Великі компанії, що використовують Flutter:

– Google (рис. 2.2.3.1): Flutter був розроблений Google, і компанія використовує його для розробки своїх власних додатків та продуктів, таких як Google Ads, Google Assistant та багато інших.

Рис. 2.3 – логотип Google

– Alibaba (рис. 2.2.3.2): Китайська компанія Alibaba використовує Flutter для створення додатків, що надають користувачам доступ до своїх платформ та сервісів, таких як AliExpress.

Рис. 2.4 – логотип Alibaba

– Tencent (рис. 2.2.3.3): Компанія Tencent використовує Flutter для розробки додатків, включаючи свої месенджери та соціальні медіа-платформи, такі як WeChat та QQ.

Рис. 2.5 – логотип Tencent

– BMW (рис. 2.2.3.4): Автомобільний гігант BMW використовує Flutter для створення додатків та інтерфейсів у своїх автомобілях з мультимедійною системою.

Рис. 2.6 – логотип BMW

Це лише кілька прикладів великих компаній, які використовують Flutter. Загалом, Flutter набуває все більшої популярності серед розробників та відомих компаній, завдяки своїм перевагам у швидкості, ефективності та кросплатформеній підтримці.

2.3 Firebase – платформа розробки мобільних та веб застосунків

Firebase є хмарною платформою від Google, яка надає широкий спектр інструментів та сервісів для розробки додатків. Це також один із найпопулярніших виборів для розробників, які використовують Flutter, оскільки Firebase прекрасно поєднується з цим фреймворком. Нижче наведені дані про Firebase та чому він є хорошим вибором для розробки додатків на Flutter:

- Автентифікація та управління користувачами: Firebase надає готові рішення для автентифікації користувачів, включаючи підтримку різних провайдерів, таких як Google, Facebook, Twitter та інші. Ви можете легко налаштувати систему автентифікації для вашого додатку на Flutter та керувати користувачами, забезпечуючи безпеку й зручність.
- База даних в реальному часі: Firebase Realtime Database - це NoSQL база даних, яка надає миттєву синхронізацію даних між клієнтами. Це особливо корисно для розробки реально-часних додатків, таких як чати, співпраця над документами та багатокористувацькі гри. Ви можете легко зберігати та отримувати дані у реальному часі з Firebase Realtime Database у своєму додатку на Flutter.
- Зберігання та синхронізація файлів: Firebase Storage дозволяє зберігати та керувати файлами у хмарі. Ви можете завантажувати, завантажувати та надсилати файли безпосередньо з вашого додатку Flutter до Firebase Storage. Це дозволяє зручно працювати з мультимедійним контентом, таким як зображення, відео, аудіо тощо.
- Серверна логіка: Firebase Cloud Functions дозволяє розробникам писати та розгортати власний серверний код безпосередньо на хмарі Firebase. Це дозволяє виконувати складні операції, обробляти події, взаємодіяти з іншими сервісами та багато іншого. Запускайте функції Firebase безпосередньо з вашого додатку на Flutter, спрощуючи серверну логіку та роблячи її більш масштабованою.
- Аналітика та звітність: Firebase Analytics дозволяє збирати та аналізувати дані про використання вашого додатку. Ви можете отримати інсайти щодо

активності користувачів, взаємодії з додатком, конверсії та багато іншого. Це допомагає вам зрозуміти, як користувачі взаємодіють з вашим додатком та покращити його функціональність.

- Інтеграція з іншими сервісами Google: Firebase також інтегрується з іншими сервісами Google, такими як Google Cloud Platform, Google Ads, Google Maps, Google Sign-In та інші. Це дає вам можливість розширювати функціональність вашого додатку та використовувати додаткові сервіси, щоб поліпшити його експерименти та ефективність.

Загалом, Firebase є потужною та розширюваною хмарною платформою, яка прекрасно поєднується з Flutter. Вона надає готові рішення для багатьох аспектів розробки додатків, таких як автентифікація, бази даних, зберігання файлів та аналітика. Firebase також інтегрується з іншими сервісами Google та надає розробникам широкі можливості для розширення функціональності їх додатків. Це робить його ідеальним вибором для розробки додатків за допомогою фреймворку Flutter.

2.4 Бібліотека Provider

Provider – це пакет, написаний у 2018 році Ремі Русле, схожий на ScopedModel, але функції якого не обмежуються наданням підкласу Model. Це теж обгортка, що укладає InheritedWidget, але провайдер може надавати будь-які об'єкти стану, зокрема, BLoC, потоки, футури та інші. Оскільки провайдер такий простий і гнучкий, Google анонсувала на конференції Google I/O '19, що надалі Provider буде переважним пакетом для управління станом. Зрозуміло, допускається і використання інших пакетів, але, якщо у вас є якісь сумніви, Google рекомендує зупинитися на Provider.

Provider побудований "з віджетами, для віджетів." Provider дає змогу помістити будь-який об'єкт, що має стан, у дерево віджетів і відкрити до нього доступ для будь-якого іншого віджета (нащадка). Також Provider допомагає керувати часом життя об'єктів станів, ініціалізуючи їх із даними і виконуючи очищення після того, як

їх буде видалено з дерева віджетів. Тож Provider підходить навіть для реалізації компонентів BLoC або може слугувати основою для інших рішень з управління станом! Або просто застосовуватися для впровадження залежностей - химерний термін, що має на увазі передачу даних у віджети в такий спосіб, який дає змогу послабити зв'язаність і поліпшити тестованість коду. Нарешті, Provider постачається з набором спеціалізованих класів, завдяки яким використовувати його ще зручніше.

2.5 Google Books Api

У світі цифрового контенту та літератури Google Books API виділяється як потужний інструмент, який надає розробникам доступ до величезної кількості книжок, метаданих та можливостей пошуку. Незалежно від того, чи ви створюєте додаток для читання, проводите дослідження або просто відкриваєте літературний універсум, Google Books API надає невід'ємний ресурс. У цій статті ми розглянемо різноманітні функції та переваги Google Books API та обговоримо, чому це чудовий вибір для інтеграції з Flutter, популярним фреймворком для розробки мобільних додатків з одним кодовою базою.

Доступ до величезної колекції книг

Google Books API дозволяє розробникам отримати доступ до величезного сховища книг, що налічує мільйони назв у різних жанрах, темах та мовах. За допомогою цього API можна здійснювати пошук книг, отримувати інформацію про книги та навіть мати доступ до вмісту книг у загальнодоступному домені. Ця велика колекція надає безліч можливостей для розробників, незалежно від того, чи вони створюють додатки для пошуку книг, освітні платформи або наукові інструменти.

Метадані та деталі книг

Однією з ключових переваг Google Books API є можливість надавати комплексні метадані та деталі про книги. Розробники можуть отримувати інформацію, таку як назва книги, автор, видавецтво, дата публікації, ISBN, опис, рейтинги та навіть обкладинки книг. Ці метадані дозволяють розробникам покращити

свої додатки, відображаючи точну та актуальну інформацію про книги, створюючи залучальний користувацький інтерфейс та покращуючи функціональність пошуку.

Потужність пошуку

Google Books API пропонує потужні можливості пошуку, що дозволяють розробникам виконувати складні запити. Ви можете шукати книги за ключовими словами, іменами авторів, ISBN, категоріями та багато чим іншим. API також підтримує розширені параметри фільтрації, що дозволяють розробникам уточнювати результати пошуку за такими критеріями, як мова, дата публікації та доступність. Ці потужні можливості пошуку дозволяють розробникам створювати інтуїтивні функції пошуку книг у своїх додатках, покращуючи користувацький досвід та забезпечуючи більшу доступність книжкових ресурсів.

Інтеграція з Flutter

Flutter є кросплатформним фреймворком, розробленим Google для створення красивих та високопродуктивних мобільних додатків. Його унікальний підхід "напишіть один раз, запустіть скрізь" дозволяє розробникам створювати додатки як для iOS, так і для Android за допомогою єдиної кодової бази. Інтеграція Google Books API з Flutter відкриває безліч можливостей для створення функціональних і збагачених книжкових додатків.

Плавна робота з мережею та обробка даних

Flutter надає багатий екосистему бібліотек і пакетів, що спрощують інтеграцію зовнішніх API, таких як Google Books API, в ваш додаток. За допомогою бібліотек мережі, таких як Dio або HTTP, розробники можуть без зусиль налагоджувати з'єднання з API, отримувати дані про книги та легко обробляти відповіді. Реактивне програмування в стилі Flutter з використанням пакетів, таких як RxDart або Flutter Bloc, додатково спрощує управління та обробку даних, отриманих з API, забезпечуючи плавний та ефективний користувацький досвід.

Залучальний користувацький інтерфейс

Виразна інтерфейсна система Flutter дозволяє розробникам створювати візуально привабливі та відзивчиві користувацькі інтерфейси. За допомогою Google Books API розробники можуть отримувати обкладинки книг, описи та інші метадані для збагачення відображення книжкової інформації в своїх додатках. Обширна бібліотека віджетів Flutter пропонує налаштовувані компоненти для створення полиць для книг, сторінок деталей книг, інтерфейсів пошуку та інших. Поєднання Google Books API та Flutter дозволяє розробникам створювати захоплюючі інтерфейси, які захоплюють користувачів та забезпечують поглиблену читацьку атмосферу.

Персоналізація та рекомендації

Інтеграція Google Books API з Flutter дозволяє розробникам використовувати велику колекцію книг та дані користувачів для надання персоналізованих досвідів та рекомендацій. Шляхом аналізу вподобань користувачів, історії читання та метаданих книг розробники можуть впроваджувати алгоритми рекомендацій та налаштовувати пропозиції, відповідно до індивідуальних інтересів кожного користувача. Ця персоналізація не лише покращує залучення користувачів, але й стимулює їх відкривати нові книги та авторів, розширюючи їх літературні горизонти.

Google Books API надає безліч функцій і переваг для розробників, які бажають створити книжкові додатки. Його велика колекція книг, розгалужені метадані, потужність пошуку та безпроблемна інтеграція з Flutter роблять його відмінним вибором для розробників, які бажають створити захоплюючі, візуально привабливі та персоналізовані додатки, що задовольняють потреби шанувальників книг у всьому світі.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПОЄКТУ

Для демонстрації можливостей Flutter було вирішено створити простий і корисний додаток під назвою "E-book". Цей додаток призначений для зручного управління та організації електронних книг. Він дозволить користувачам переглядати та легко знаходити електронні книги в одному місці. Користувачі зможуть відмічати улюблені книги та швидко переходити до потрібної сторінки. Додаток "E-book" буде інтуїтивно зрозумілим та привабливим для користувачів, що дозволить їм із задоволенням використовувати його для організації своєї електронної бібліотеки.

За допомогою сервісу Firebase було реалізовано аутентифікацію та збереження улюблених книг.

Для відображення книг у додатку було використано Google Books API, через його зручне інтегрування з Flutter та велику колекцію книг.

3.1 Огляд інтерфейсу користувача

3.1.1 Головний екран

При створені UI, було вирішено створити інтерфейс, який буде інтуїтивно зрозумілим та .

home_page.dart – це файл, в якому реалізовано код для відображення головного екрану додатка. Цей файл містить клас HomePage, який є підкласом класу StatefulWidget, через те що потрібно зберігати та змінювати стан на цьому екрані (рис.3.1 та Рис. 3.2).

Ми також можемо використовувати стан, який дозволяє нам змінювати вигляд екрану під час взаємодії з користувачем та залежно від даних, які отримано від обраного API, та даних про користувача з Firebase.

Наприклад, у AppBar ми бачимо надпис "Welcome to Ebook", який автоматично змінюється в залежності від того, чи користувач увійшов у свій аккаунт. Крім того, при появі нових книг вони автоматично відображаються. Після зміни стану, Flutter автоматично викликає метод build() і оновлює відображення на екрані відповідно до нового стану.

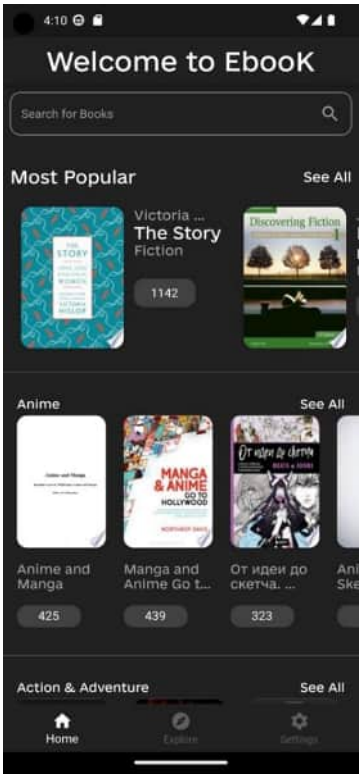

Рис. 3.1 – головний екран

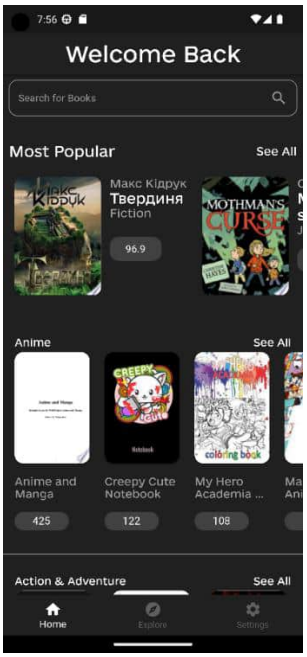

Рис. 3.2 – головний екран

У цій наведеній нижче частині коду (рис 3.3) перевіряється стан користувача.

```
(user == null)
            ? const Text(
                'Welcome to EbooK'
,
                style: TextStyle(
fontFamily: 'e-Ukraine', fontSize:
 30),
              )
            : const Text(
                'Welcome Back',
                style: TextStyle(
fontFamily: 'e-Ukraine', fontSize:
 30),
              ),
```

Рис. 3.3 – частина коду, яка відповідає за зміну стану

Нижче можна побачити поле пошуку, у якому здійснюється пошук за словами. Програма приймає слово (або слова) користувача та відображає книги, в назві або описі яких є ці слова, на окремому екрані search_screen.dart. Результати пошуку та їх вигляд можна побачити на рисунках 3.4 та 3.5.

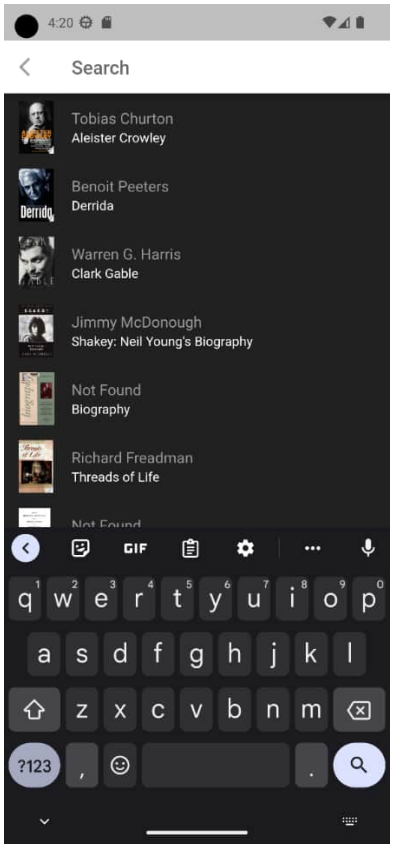

Рис. 3.4 – екран пошуку у звичайному стані

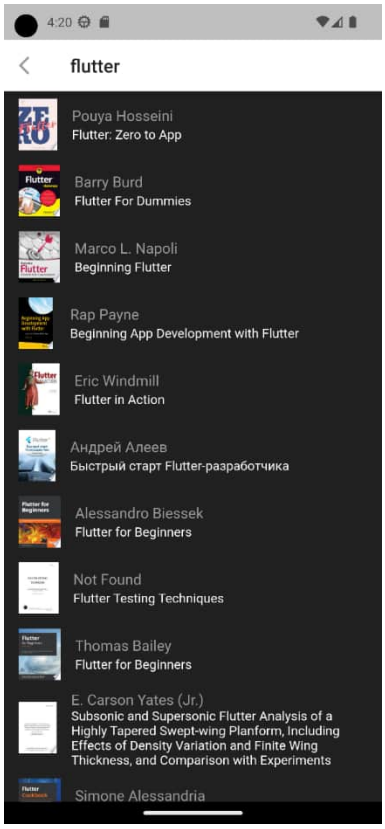

Рис. 3.5 – екран пошуку після здійснення пошуку

Далі можна побачити, як книги відображаються. Спочатку показані найпопулярніші книги на даний момент, а потім книги відсортовані за категоріями. На цьому екрані також показані популярні категорії.

Усі книги розташовані у горизонтальній каруселі, яку можна прокручувати. Кожна карусель може містити максимум 10 книг. Для побудови цих каруселей використано ListView.builder (рис. 3.6).

```
ListView.builder(
                shrinkWrap: true,
                physics: const BouncingScrollPhysics(),
                scrollDirection: Axis.horizontal,
                itemCount: 10,
                itemBuilder: (context, index)
```

Рис 3.6 – елемент коду на якому зображено реалізацію ListView.builder

ListView.builder є одним із віджетів у Flutter, який дозволяє ефективно відображати список елементів на екрані. Цей віджет особливо корисний, коли

потрібно відображати велику кількість елементів або елементи, які можуть динамічно змінюватись.

ListView.builder має два обов'язкових параметри: itemCount і itemBuilder.

- itemCount - це кількість елементів, які мають бути відображені у списку. Зазвичай це значення залежить від кількості даних у джерелі даних.

- itemBuilder - це функція, яка викликається для кожного елемента у списку. Вона приймає два параметри: контекст (BuildContext) і індекс елемента (int).

Одна з головних переваг ListView.builder полягає в тому, що він ефективно використовує пам'ять та ресурси, оскільки будує тільки видимі на екрані елементи, а не всю колекцію даних. При прокрутці списку ListView.builder переосмислює та перебудовує тільки видимі елементи, що дозволяє забезпечити швидку та ефективну прокрутку списку, навіть якщо список містить тисячі елементів.

У середині віджету розміщені картки, які складаються з обкладинки книги, відображення кількості сторінок та назви книги. Ця компоновка дозволяє користувачам швидко отримати важливу інформацію про книгу і зробити відповідний вибір.

Нижче розташована BottomAppBar, яка є аналогом AppBar, але розміщена у нижній частині екрану. BottomAppBar виконує функцію навігації між різними екранами. Натискання на іконку компасу з підписом "Explore" перенаправляє на наступний екран.

Перехід на наступний екран за допомогою BottomAppBar забезпечує зручний спосіб переходу між

різними функціональними частинами додатку та дозволяє користувачам легко виконувати потрібні дії та досліджувати нові можливості.

На рисунку 3.7 зображено екран категорій. На цій сторінці зібрані різні категорії, які допомагають користувачеві знайти більш детальну інформацію про певну тематику. Кожна категорія має свою унікальну назву і супроводжується

зображенням, що ілюструє ту саму тему. Це дозволяє користувачеві швидко визначити бажану категорію за допомогою візуального сприйняття. Категорії представлені у вигляді карток, які організовані за допомогою GridView.builder (рис. 3.8), що дозволяє відображати елементи у вигляді сітки. При натисканні на будь-яку категорію користувач буде спрямований на сторінку, де можна переглянути всі книги, що належать до цієї категорії. Ця функціональність забезпечує зручний спосіб досліджувати книжкові теми та швидко знайти бажану категорію для подальшого вибору книг.

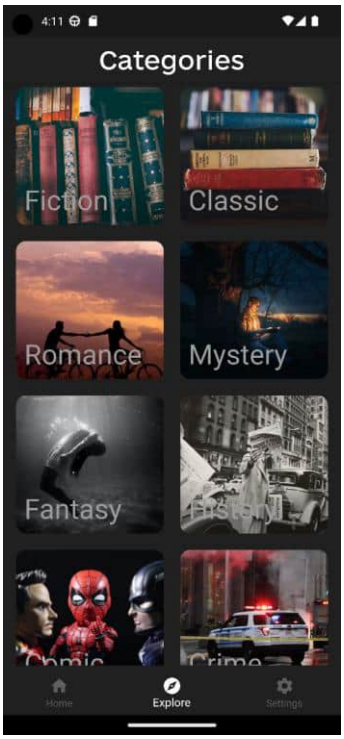

Рис. 3.7 – екран Explore

```
GridView.builder(
          physics: const BouncingScrollPhysics(),
          itemCount: categoriesName.length,
          gridDelegate: const
SliverGridDelegateWithFixedCrossAxisCount(
              childAspectRatio: 16 / 15,
              crossAxisCount: 2,
              crossAxisSpacing: 20,
              mainAxisSpacing: 20),
          itemBuilder: (context, index)
```

Рис. 3.8 – елемент коду, який демонструє GridView(categories.dart)

На сторінці, зображеній на рисунку 3.9, елементи відображаються за допомогою GridView.builder, що надає сітчасту організацію елементів. Кожен елемент представляє окрему книгу і має зовнішній вигляд, який включає обкладинку книги. Якщо деяка книга не має обкладинки, то на її місці буде відображено першу сторінку вмісту книги. Це забезпечує більш гнучке та привабливе візуальне представлення книжок у списку категорій, дозволяючи користувачам легко розпізнати книги та здійснювати вибір на основі зображень та вмісту.

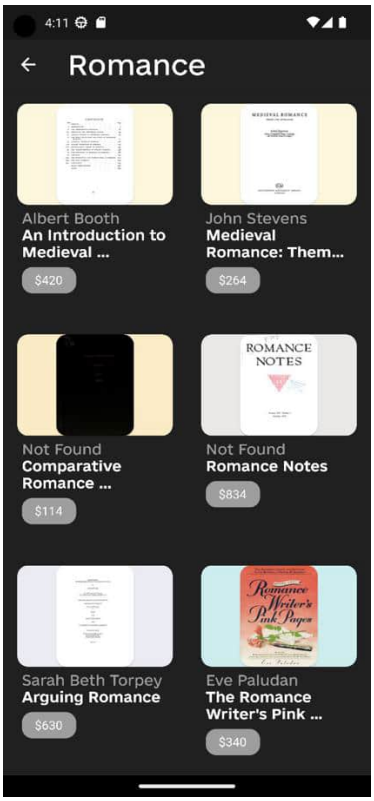

Рис. 3.9 – екран категорії Romance

3.1.2 Детальний екран книги

У будь-якому місці додатку, незалежно від контексту, при натисканні на будь-яку книгу користувача перенаправлятимуть на екран з детальною інформацією про книгу, який зображений на рисунку 3.10 Ця функціональність створює єдиний і послідовний шлях для отримання деталей про книгу в додатку, незалежно від того, з якого екрану чи розділу користувач натиснув на книгу. Такий підхід полегшує

навігацію та забезпечує зручний спосіб дізнатись більше про книгу з будь-якого місця в додатку.

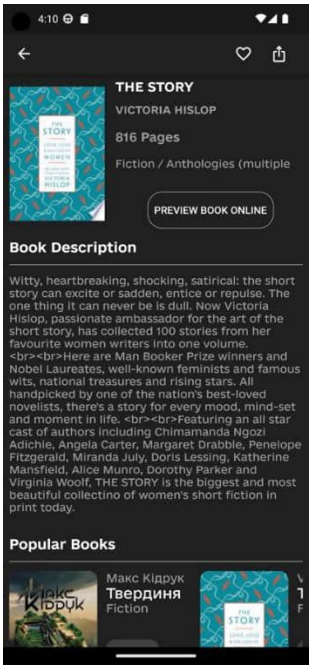

Рис. 3.10 – екран з детальною інформацією про книгу

На цьому екрані змінюється верхня панель (AppBar), яка містить три інтуїтивно зрозумілі іконки, що виконують певні функції.

- Стрілочка: Ця іконка дозволяє повернутися на попередній екран.

- Сердечко: Ця іконка дозволяє додавати книги до списку улюблених.

- Shared_Icon: Ця іконка дозволяє поділитись книгою з іншими людьми.

Після AppBar розташована обкладинка книги, а поряд з нею є детальна інформація про книгу, така як назва, автор, кількість сторінок та жанри, до яких вона відноситься. Також присутня мінімалістична кнопка, яка відкриє сторінку книги у браузері за замовчуванням, де можна переглянути книгу у демо-версії або, якщо книга безкоштовна, повністю її прочитати у зручному форматі.

Нижче розташований текст, який є описом книги, що відкрили. Цей текст автоматично адаптується до будь-якої кількості символів і гармонійно виглядає на будь-якому пристрої. Це досягається завдяки віджету Flutter, який автоматично

розтягується висотою, необхідною для відображення тексту, і включає його в себе (використовуючи віджет Text()).

На завершення сторінки розташована горизонтальна "карусель" з популярними книгами, яку вже було описано раніше.

3.1.3 Екран налаштувань

Перейти до сторінки налаштувань можна натиснувши в BottomAppBar на іконку шестерні та підписом “Settings” (рис. 3.11).

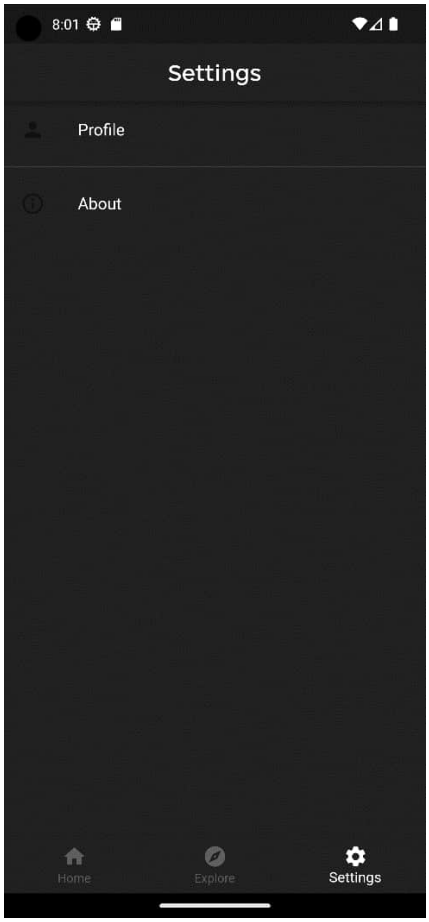

Рис. 3.11 – екран налаштувань

На цьому екрані розташовані два розділи, які надають різні функціональні можливості:

- Профіль (Profile): Цей розділ присвячений профілю користувача. Залежно від стану входу в обліковий запис буде відображатись різна інформація та

доступний різний функціонал. Якщо користувач ще не увійшов у додаток, йому буде запропоновано зареєструватись або увійти в існуючий обліковий запис.

– Про додаток (About): Цей розділ містить загальну інформацію про додаток, таку як копірайт, опис функціональності та можливості. Тут користувач може знайти додаткові відомості про додаток і його розробників.

Якщо користувач натисне на розділ "Профіль" (Profile), залежно від його стану (чи він увійшов в обліковий запис чи ні), він побачить різні екрани. Розпочати можна з розгляду сценарію, коли користувач ще не увійшов у власний акаунт, а тільки опинився на екрані входу (рис. 3.12).

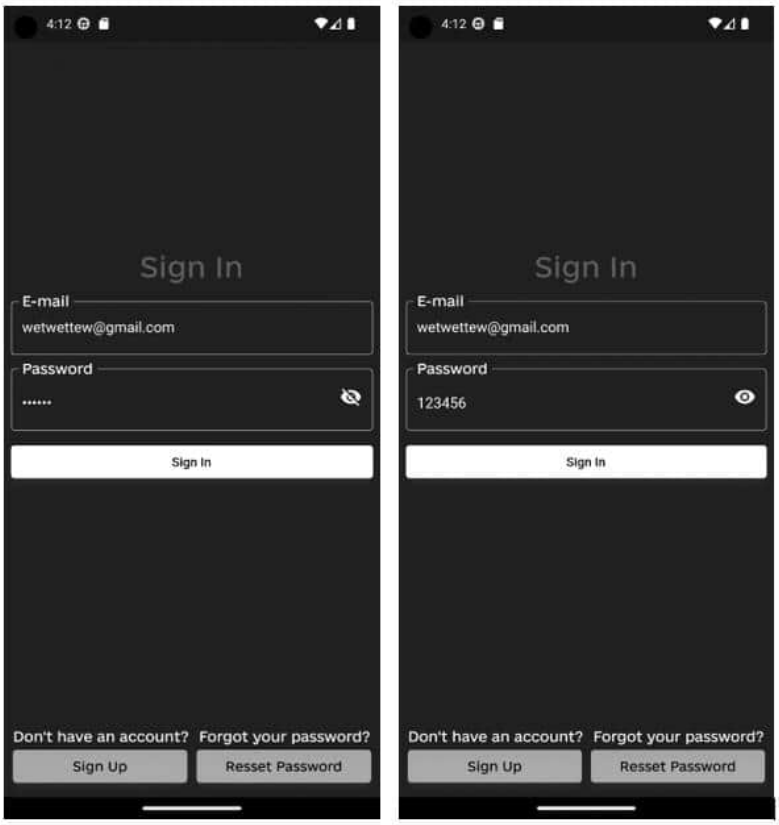

Рис. 3.12 – екран входу

На цьому екрані розташовані два поля для введення даних, які потрібні для входу в обліковий запис. В одному з полів, де користувач вводить свій пароль, є кнопка, що дозволяє змінювати стан поля. При натисканні на іконку закритого ока, введені символи стають видимими, а сама іконка змінюється на відкрите око.

Якщо користувач правильно введе всі дані і він вже зареєстрований, то його перенаправлять на головний екран додатку. Проте якщо користувач ще не зареєстрований, то він може натиснути на кнопку "Sign up" в лівому нижньому куті. Це відкриє екран реєстрації, де користувач зможе створити свій обліковий запис. Цей процес дозволяє новим користувачам зареєструватися та отримати доступ до функціоналу додатку.

Екран, що зображено на рисунку 3.13, має схожий стиль з попереднім, але має додатковий функціонал. Внизу екрану є кнопка, яка призначена для випадків, коли користувач помилково потрапив на цю сторінку. Коли користувач введе всі необхідні дані, він буде перенаправлений на сторінку підтвердження електронної адреси. Це дозволяє забезпечити безпеку та достовірність електронних адрес користувачів перед дозволом їм використовувати функціонал додатку.

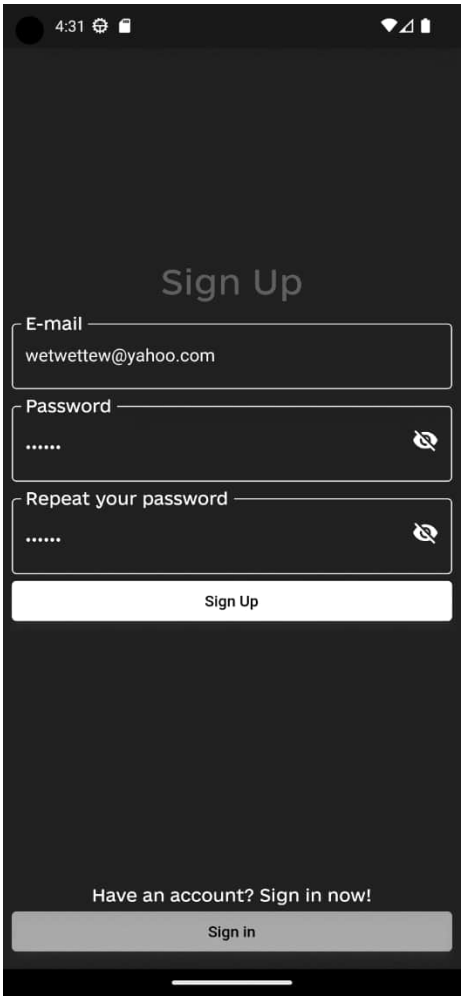

Рис. 3.13 – екран реєстрації

На цьому екрані (рис. 3.15) можна побачити дві кнопки, які є неактивними декілька секунд. Якщо користувач, через якісь причини не отримав одразу повідомлення про підтвердження, тоді у нього є можливість натиснути кнопку “Resend” з іконкою повідомлення. Після цього система відправить повторне повідомлення. Якщо ж користувач передумає відправляти повідомлення, у нього є можливість натиснути кнопку “Cancel”, та повернутись на екран реєстрації.

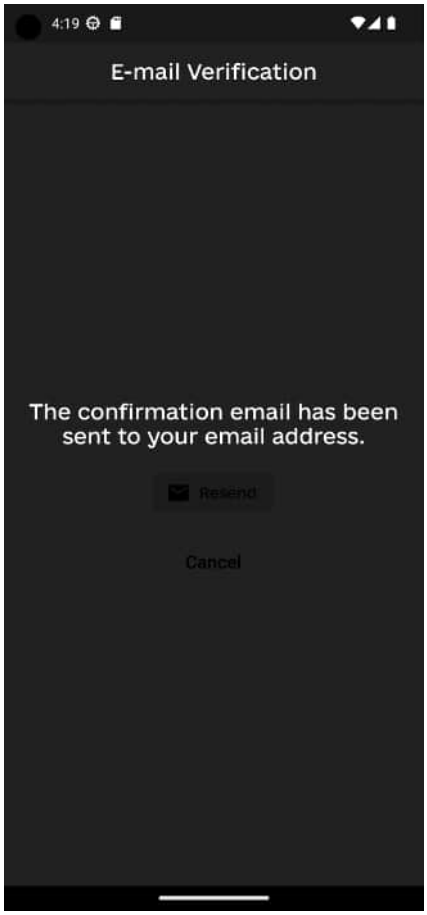

Рис. 3.15– Екран підтвердження

Після успішного підтвердження, користувача буде направлено на головну сторінку.

Тепер у налаштуваннях додатку екран “Profile” буде мати такий вигляд, як зображено на рисунку 3.16.

На екрані, що на рисунку 3.16, є дві текстові кнопки. “Sign Out” – дозволяє вийти із аккаунта, після натискання на цю кнопку користувач буде направлений на

головну сторінку. Друга кнопка “Reset Password” створена для того, щоб змінити пароль (рис. 3.17).

Рис. 3.16 – екран Profile

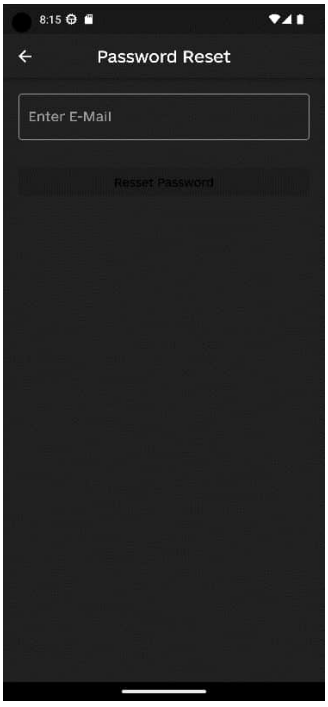

Рис. 3.17 – Екран Password Reset

Після введення E-mail, користувач отримає повідомлення, у якому буде посилання із шаблоном зміни паролю, після успішної зміни паролю користувача буде направлено на головну сторінку, де внизу екрану він побачить повідомлення що все пройшло успішно (рис 3.18).

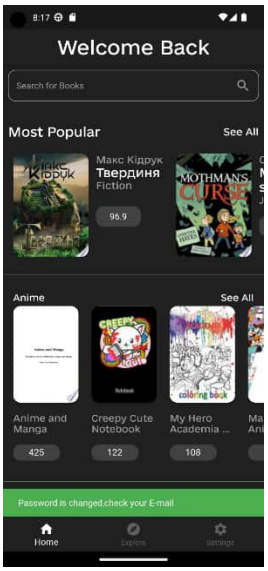

Рис. 3.18 – приклад повідомлення

Реалізація таких повідомлень є дуже зручної у Flutter завдяки віджету SnackBar (рис 3.19).

```
class SnackBarService {
  static const errorColor = Colors.red;
  static const okColor = Colors.green;

  static Future<void> showSnackBar(
      BuildContext context, String message, bool error) async {
    ScaffoldMessenger.of(context).removeCurrentSnackBar();

    final snackBar = SnackBar(
      content: Text(message),
      backgroundColor: error ? errorColor : okColor,
    );

    ScaffoldMessenger.of(context).showSnackBar(snackBar);
  }
}
```

Рис. 3.19 – приклад реалізації SnackBar

3.2 Огляд взаємодії API та Firebase Authentication

3.2.1 Приклад відображення книг за допомогою Google Books API

Для прикладу відображення книг візьмемо книги категорії “Adventure”. Знайти даний код можна у файлі adventure.dart.

У даному коді використовується FutureBuilder для побудови віджету, який залежить від асинхронної операції отримання даних (getBookData3()). FutureBuilder приймає future - об'єкт Future, і будує віджети на основі стану snapshot.

Описати кожен випадок стану snapshot можна так:

- snapshot.hasError: Якщо сталася помилка під час виконання асинхронної операції, виводиться повідомлення "Opps! Try again later!" (за допомогою Center та Text).
- snapshot.hasData: Якщо дані успішно отримані, будується ListView.builder.
    - shrinkWrap встановлено на true, щоб віджет займав тільки стільки місця, скільки йому потрібно.
    - physics встановлено на BouncingScrollPhysics(), щоб додати ефект пружини при прокручуванні.
    - scrollDirection встановлено на Axis.horizontal, щоб список прокручувався горизонтально.
    - itemCount встановлено на 10, щоб відобразити лише перші 10 елементів.
    - itemBuilder викликається для кожного елемента списка та повертає Container, який містить дані про книгу. Цей контейнер містить зображення книги (Image) та іншу інформацію, таку як заголовок та кількість сторінок. При натисканні на книгу, викликається DetailsScreen, передаючи ідентифікатор книги.
- В іншому випадку, коли дані ще завантажуються, відображається CircularProgressIndicator (крутильний індикатор завантаження) за допомогою Center.

Цей віджет використовує стан snapshot для відображення різних варіантів вмісту, в залежності від стану завантаження даних.

У папці моделі є файл api.dart, який відповідає за опрацювання даних для кожної категорії. Також у цій директорії знаходяться файли book.dart та detail.dart, які є opensource бібліотеками. Вони відповідають за опрацювання get запитів та розшифрування json.

3.2.2 Аутентифікація користувача за допомогою FireBase Auth

Firebase Authentication є сервісом аутентифікації користувачів, який надається Firebase. Взаємодія з Firebase Authentication в Flutter передбачає використання пакету firebase_auth, який надає зручний спосіб роботи з аутентифікацією у вашому додатку.

Для початку потрібно добавити залежність в pubspec.yaml файлі проекту.

Далі потрібно імпортувати пакет firebase_auth у файлі Dart, де планується використовувати Firebase Authentication. Наступним кроком буде ініціалізація Firebase (рис 3.20).

```
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(options:
DefaultFirebaseOptions.currentPlatform);
  runApp(const MyApp());
}
```

Рис. 3.20 – ініціалізація Firebase

Перейдемо до процесу реєстрації, який відбувається “під капотом”, приведений код (рис. 3.21) відповідає за функціональність реєстрації користувача у додатку з використанням Firebase Authentication. Варто розглянути кожен крок окремо:

– Перший рядок final navigator = Navigator.of(context); отримує об'єкт Navigator, який використовується для навігації між екранами.

- Наступний рядок final isValid = formKey.currentState!.validate(); перевіряє, чи є форма валідною, використовуючи formKey.
- Перевірка if (_passwordTextInputController.text != _passwordTextRepeatInputController.text) перевіряє, чи обидва введення пароля співпадають. Якщо ні, то показується спливаюче повідомлення за допомогою SnackBarService.showSnackBar().
- Блок try - catch використовується для спроби створити нового користувача з використанням createUserWithEmailAndPassword() методу FirebaseAuth.instance. Електронна пошта і пароль передаються як параметри з відповідних полів введення.
- Якщо під час реєстрації виникає помилка FirebaseAuthException, перевіряється код помилки за допомогою e.code і програма відповідно реагує. У випадку, якщо код помилки є 'email-already-in-use', з’являється повідомлення про помилку за допомогою SnackBarService.showSnackBar(). У іншому випадку, якщо код помилки невідомий, також показується відповідне повідомлення.
- Якщо реєстрація пройшла успішно, хорошим рішенням є використати navigator.pushNamedAndRemoveUntil() для переходу на інший екран, і на додачу видалення попередніх екранів зі стеку навігації. Це сприяє покращенню швидкодії та ефективності програми, зменшуючи навантаження на ресурси.

```
Future<void> _signUp() async {
    final navigator = Navigator.of(context);

    final isValid = formKey.currentState!.validate();
    if (!isValid) return;
    if (_passwordTextInputController.text !=
        _passwordTextRepeatInputController.text) {
      SnackBarService.showSnackBar(
        context,
        'Passwords should be the same',
        true,
      );
      return;
    }
    try {
      await FirebaseAuth.instance.createUserWithEmailAndPassword(
        email: _emailTextInputController.text.trim(),
        password: _passwordTextInputController.text.trim(),
      );
    } on FirebaseAuthException catch (e) {
      // ignore: avoid_print
      print(e.code);
      if (e.code == 'email-already-in-use') {
        SnackBarService.showSnackBar(
          context,
          'This E-mail is already in use',
          true,
        );
        return;
      } else {
        SnackBarService.showSnackBar(
          context,
          'Unknown error,please contact customer support',
          true,
        );
      }
```

Рис. 3.21 – функція реєстрації користувача

3.2.3 Консоль керування Firebase Authentication

Консоль Firebase Authentication є інтерфейсом, який дозволяє адміністраторам керувати користувачами, проводити аутентифікацію та налаштовувати їх параметри в Firebase.

У консолі Firebase Authentication можна здійснювати такі дії як:

– Керування користувачами: Можна переглядати, додавати, редагувати та видаляти користувачів вашого додатку. Кожен користувач має унікальний ідентифікатор користувача (UID), який використовується для ідентифікації користувача в системі (рис 3.22).

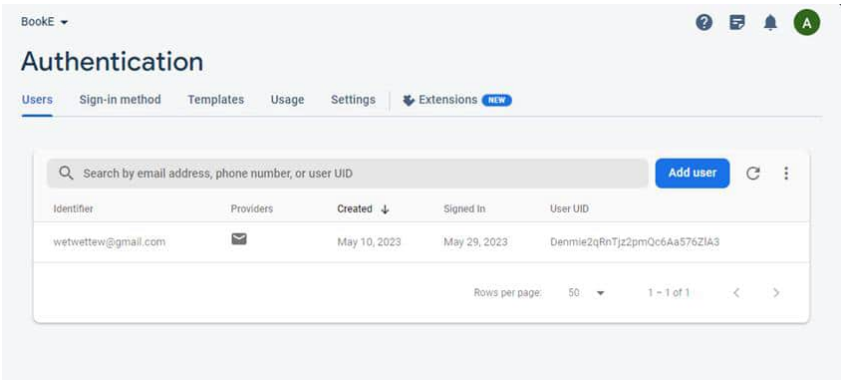

Рис. 3.22 – UI консолі

– Перегляд інформації про користувачів: Можна переглядати деталі кожного користувача, такі як електронна пошта, інформація про останній вхід, статус підтвердження електронної пошти та інше.

– Налаштування методів аутентифікації: Можна налаштовувати доступні методи аутентифікації для додатку. Firebase Authentication надає різноманітні методи аутентифікації, такі як електронна пошта/пароль, Google, Facebook, Twitter, GitHub та інші. Є можливість активувати та налаштувати будь-які з цих методів для використання користувачами (рис. 3.23).

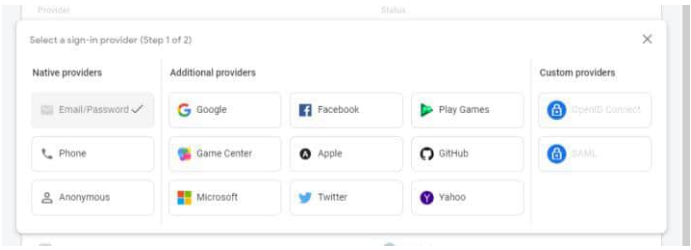

Рис. 3.23 – Можливі методи аутентифікації

– Налаштування сторінок підтвердження (рис 3.24): Можна налаштовувати сторінки підтвердження для методів аутентифікації, які вимагають підтвердження електронної пошти або номера телефону користувача.

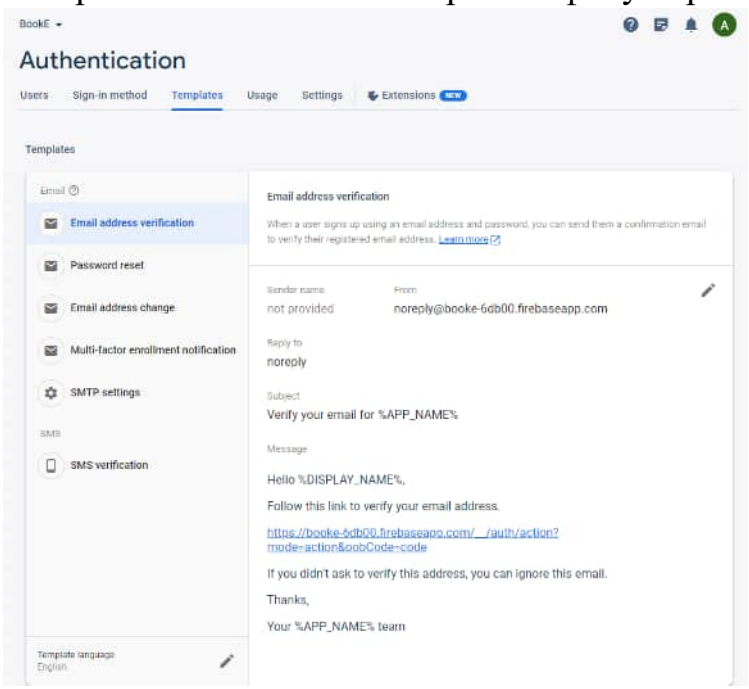

Рис. 3.24 – шаблони сторінки підтвердження

– Управління обмеженнями доступу: Можна налаштовувати обмеження доступу до додатку для певних користувачів або груп користувачів. Наприклад, обмежити доступ лише для певних електронних адрес або використовувати правила Firebase для точного контролю доступу.

ВИСНОВОК

1. Зроблено огляд існуючих мобільних додатків для електронних бібліотек, на основі якого сформована мета і концепція даного проєкту.
2. З використанням фреймворку Dart/Flutter створено прототип мобільної бібліотеки, яка може підтримуватись на платформах Android та iOS.
3. Здійснено інтеграцію додатку з Google Books API, що забезпечило можливість отримувати йому інформацію про книги.
4. Розроблено багатофункціональний користувацький інтерфейс, який дозволяє легко організовувати роботу з колекціями книг.
5. Для забезпечення безпеки та автентифікації користувачів у додатку була використана Firebase Authentication.
6. Проведено тестування розробленого додатку.
7. Даний додаток володіє гнучкістю щодо введення змін в нього, такі як:
    a) Покращення інтерфейсу.
    b) Взаємодія між користувачами(така собі невелика соціальна мережа про книги).
    c) Збільшення колекції книг шляхом надання можливості користувачам завантажувати їхні власні роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Buckett C. Dart in Action. –Manning Publications Co, 2013.– 424 p.*
2. *Windmill E. Flutter in Action.–Manning Publications Co, 2020. –368p .*
3. *R. K. Kushwaha, Jend Lal Singh. Use of Mobile Library Services and Technology//International Journal of Advanced Research in Science, Communication and Technology. –2022.–Vol.2, No.2. – P. 217–219.*
4. *Jashandeep Singh, Swapnil Srivastva. Flutter and Firebase Making Cross-Platform application development Hassle-Free. International Research Journal of Modernization in Engineering Technology and Science. – 2022. –Vol.4, No.4 .– P.1819-1827.*
5. *Flutter documentation. Flutter documentation | Flutter. URL: https://docs.flutter.dev.*
6. *Dart documentation. Dart programming language | Dart. URL: https://dart.dev/guides.*
7. *Firebase Documentation. Firebase. URL: https://firebase.google.com/docs.*
8. *Using the API | Google Books APIs | Google for Developers. Google for Developers. URL: https://developers.google.com/books/docs/v1/using*
9. *Meiller D. Modern App Development with Dart and Flutter 2: A Comprehensive Introduction to Flutter. de Gruyter GmbH, Walter, 2021. 249 c.*
10. *Alessandria S., Kayfitz B. Flutter Cookbook: Over 100 Proven Techniques and Solutions on Mobile Development with Flutter and Dart. Packt Publishing, Limited, 2021.*
11. *Zammetti F. Practical Flutter: Improve your Mobile Development with Google's Latest Open-Source SDK. Apress, 2019. 416 c.*