TABLE OF CONTENTS

CERTIFICATE OF PROJECT COMPLETION

This certificate proudly attests that the project entitled "APP BOOK" has been successfully accomplished by the dedicated group members and is formally submitted to Aptech Computer Education.

Project Title: APP BOOK
Program: Advanced Diploma in Software Engineering (ADSE)

Members of the Group:
- Kazeem Oyinkansola Mistura
- Oboh Nonso Elijah
- Ayinde Opeyemi Qudus
- Lawal Femi David

We hereby officially recognize the commendable completion of the project by the group members. The project's content and execution align impeccably with the established criteria and standards prescribed for the ADSE program.

AKNOWLEDGMENT

Glory be to God for all that He has accomplished for us. His fortitude and direction have kept us going on this trip. Oh God, the All-Powerful Creator, You have our eternal gratitude!

We would like to take this opportunity to express our sincere gratitude to our parents for their unflagging support, sacrifices, and ongoing inspiration during our journey.

We would like to recognize Mr. Precious Oladele, a respected member of our faculty and a committed mentor who has contributed significantly to our development. Your advice and experience have been priceless.

We would want to express our appreciation to the school counselor for always being there to offer advice and assistance. Your presence has given us comfort.

Finally, we want to express our sincere gratitude to our director.

SYNOPSIS

The "**App Book**" project is a comprehensive and innovative mobile application designed to revolutionize the way people engage with books and literature. In an era marked by digital transformation, this application aims to bridge the gap between traditional reading and modern technology, providing users with an enriched reading experience.

Problem Addressed:

1. Difficulty in Finding Books: The app addresses the common problem of users struggling to find books that match their interests or preferences.

2. Cost of Books: It offers a solution for users who want to read without incurring the cost of purchasing books.

Key Features:

1. Free Books Section:

- This section provides access to curated free websites and platforms where users can read books without any cost.

2. Paid Books Section:

- The paid books section allows users to explore and potentially purchase books if they don't find them for free.

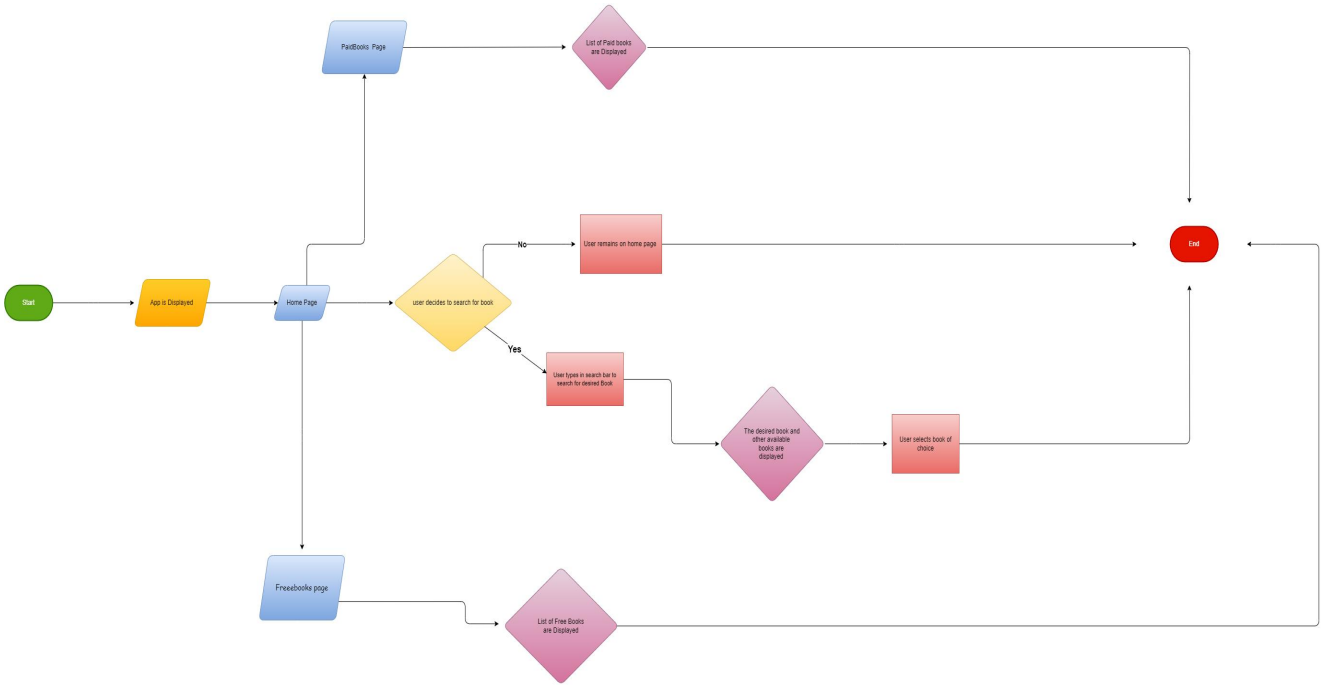
3. Search Functionality:

- The search box is a crucial feature, enabling users to search for specific books by title, author, genre, or keywords.

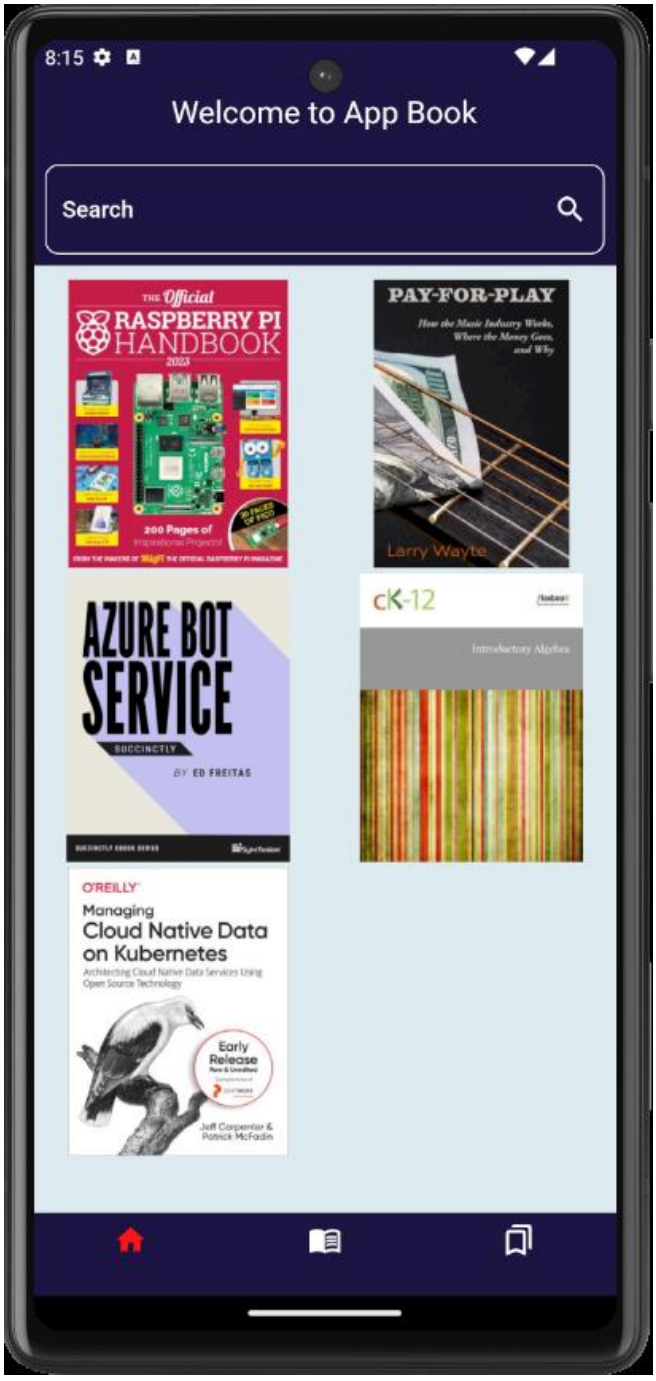
ANALYSIS

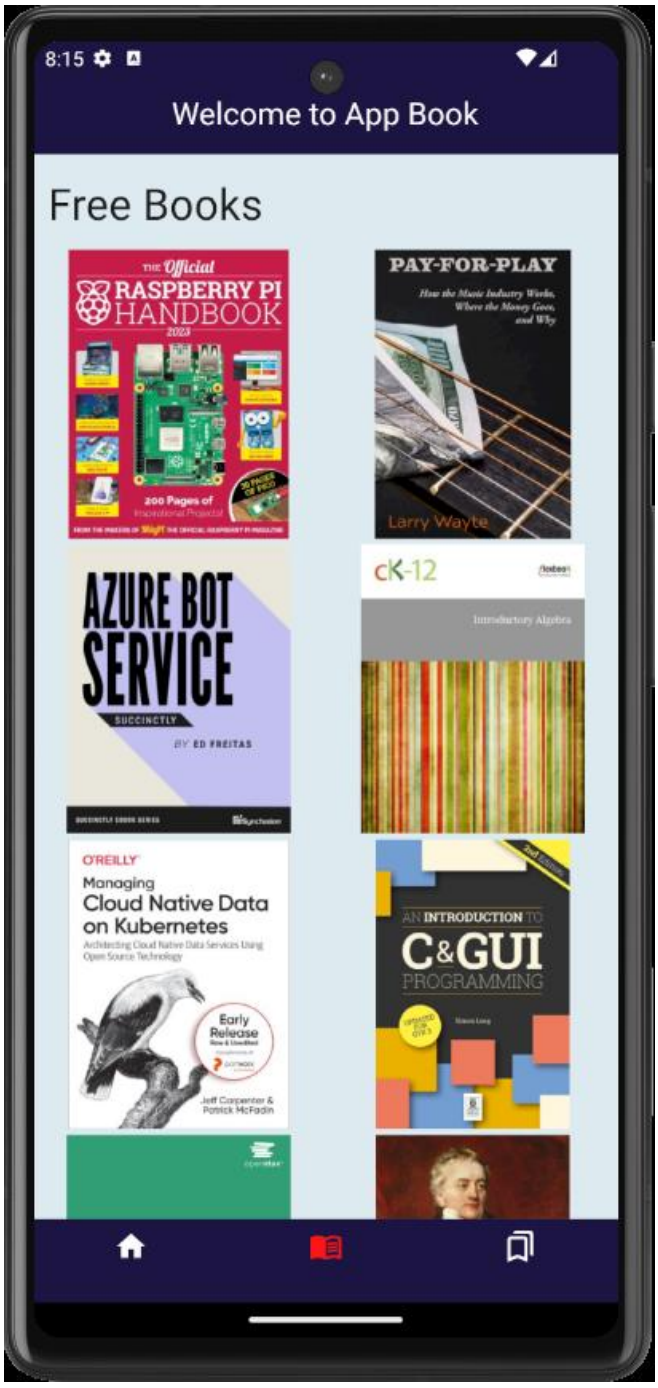
The development of this project was carried out using the Flutter framework and the Visual Studio Code (VS Code) Integrated Development Environment (IDE) for coding.

FLOWCHART


PaidBooks Page
List of Paid books are Displayed
No
User remains on home page
End
Start
App is Displayed
Home Page
user decides to search for book
Yes
User types in search bar to search for desired Book
The desired book and other available books are displayed
User selects book of choice
Freeebooks page
List of Free Books are Displayed

SCREENSHOT


8:15
Welcome to App Book
Search
THE Official RASPBERRY PI HANDBOOK
200 Pages of
PAY-FOR-PLAY
How the Music Industry Works, Where the Money Goes, and Why
Larry Wayte
AZURE BOT SERVICE
SUCCINCTLY
BY ED FREITAS
CK-12
Introductory Algebra
O'REILLY
Managing Cloud Native Data on Kubernetes
Early Release
Jeff Carpenter & Patrick McFadin


8:15
Welcome to App Book
Free Books
THE Official RASPBERRY PI HANDBOOK 2023
200 Pages of
PAY-FOR-PLAY
How the Music Industry Works, Where the Money Goes, and Why
Larry Wayte
AZURE BOT SERVICE
SUCCINCTLY
BY ED FREITAS
CK-12
Introductory Algebra
O'REILLY
Managing Cloud Native Data on Kubernetes
Early Release
Jeff Carpenter & Patrick McFadin
AN INTRODUCTION TO
C&GUI PROGRAMMING


8:15
Managing Cloud Native Data o...
O'REILLY
Managing
Cloud Native Data
on Kubernetes
Architecting Cloud Native Data Services Using
Open Source Technology
Early
Release
Raw & Unedited
Jeff Carpenter &
Patrick McFadin
Title: Managing Cloud Native Data on Kubernetes
Author: Jeff Carpenter, Patrick McFadin
Download: https://www.dbooks.org/managing-cloud-native-data-on-kubernetes-1098111389/

SOURCE CODE
Main

```
import 'package:bookapp/home.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

void main() {
runApp(const MyApp());
}

class MyApp extends StatelessWidget {
const MyApp({super.key});

// This widget is the root of your application.
@override
Widget build(BuildContext context) {
return GetMaterialApp(
title: 'Flutter Demo',
debugShowCheckedModeBanner: false,
theme: ThemeData(
scaffoldBackgroundColor: Color(0xFFEDEECF2),
colorScheme: ColorScheme.fromSeed(seedColor:
Colors.deepPurple),
useMaterial3: true,
),
home: const HomePage(),
);
}
}
```

HOME

```
import 'package:bookapp/FreePage.dart';
import 'package:bookapp/PaidPage.dart';
import
'package:bookapp/controllers/book_controller.dart';
import 'package:bookapp/details.dart';
import 'package:bookapp/models/books_model.dart';
import 'package:bookapp/search.dart';
import 'package:bookapp/home_book.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class HomePage extends StatefulWidget {
const HomePage({super.key});

@override
State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
late final BookController _bookController;
late final TextEditingController _searchController;

@override
void initState() {
_bookController = Get.put(BookController());
_searchController = TextEditingController();
super.initState();
}

final pages = [HomeBook(), FreePage(), PaidPage()];

int currentIndex = 0;
Color selectedItemColor = Color.fromARGB(255, 255, 16,
16);

@override
```

```
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
backgroundColor: Color(0xFF1f1545),
title: Center(
child: Text(
"Welcome to App Book",
style: TextStyle(color: Colors.white),
),
),
),
body: pages[currentIndex],
bottomNavigationBar: BottomNavigationBar(
backgroundColor: Color(0xFF1f1545),
selectedItemColor: Colors.blueGrey,
unselectedItemColor: Colors.indigoAccent,
onTap: (cIndex) {
setState(() {
currentIndex = cIndex;
});
},
type: BottomNavigationBarType.fixed,
items: [
BottomNavigationBarItem(
icon: Icon(
Icons.home,
color: currentIndex == 0 ? selectedItemColor :
Colors.white,
),
label: '',
),
BottomNavigationBarItem(
icon: Icon(
Icons.menu_book_outlined,
color: currentIndex == 1 ? selectedItemColor :
Colors.white,
),
label: '',
```

```
),
BottomNavigationBarItem(
icon: Icon(
Icons.bookmarks_outlined,
color: currentIndex == 2 ? selectedItemColor :
Colors.white,
),
label: '',
),
],
),
);
}
}
```

Home_book

```
import 'dart:math';
import'package:bookapp/controllers/book_controller.dart’
;import 'package:bookapp/details.dart';
import 'package:bookapp/models/books_model.dart';
import 'package:bookapp/search.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class HomeBook extends StatefulWidget {
const HomeBook({super.key});

@override
State<HomeBook> createState() => _HomeBookState();
}

class _HomeBookState extends State<HomeBook> {
@override
late final BookController _bookController;

late final TextEditingController _searchController;

@override
void initState() {
_bookController = Get.put(BookController());
_searchController = TextEditingController();
super.initState();
}

Widget build(BuildContext context) {
return SingleChildScrollView(
child: SafeArea(
child: Column(
children: [
Container(
padding: const EdgeInsets.all(8.0),
decoration: BoxDecoration(
```

```
color: Color(0xFF1f1545),
),
child: TextField(
style: TextStyle(color: Colors.white),
controller: _searchController,
decoration: InputDecoration(
hintStyle: TextStyle(color: Colors.white),
border: InputBorder.none,
hintText: 'Search',
suffixIcon: IconButton(
onPressed: () => Get.to(
() => SearchPage(query:
_searchController.text),
),
icon: Icon(
Icons.search,
color: Colors.white,
),
),
enabledBorder: OutlineInputBorder(
borderSide: BorderSide(color: Colors.white),
borderRadius: BorderRadius.circular(10.0),
),
focusedBorder: OutlineInputBorder(
borderSide: BorderSide(color: Colors.white),
borderRadius: BorderRadius.circular(10.0),
),
),
),
),
SizedBox(
height: 10,
),
GetBuilder<BookController>(
init: _bookController,
builder: (controller) {
return controller.isLoading
? const Center(
```

```
child: CircularProgressIndicator(),
)
: GridView.builder(
shrinkWrap: true,
scrollDirection: Axis.vertical,
gridDelegate:
const
SliverGridDelegateWithFixedCrossAxisCount(
crossAxisCount: 2,
mainAxisSpacing: 4,
crossAxisSpacing: 3,
),
itemCount: controller.books.take(5).length,
itemBuilder: (context, index) {
return InkWell(
onTap: () => Get.to(
() => DetailsPage(
book: controller.books[index],
),
),
child: Container(
width: 300,
height: 300,
decoration: BoxDecoration(
image: DecorationImage(
image: NetworkImage(
controller.books[index].image ?? ''),
),
),
),
);
},
);
},
)
],
),
),
```

```
    );
  }
}
```

Freepage

```
import 'dart:math';

import
'package:bookapp/controllers/book_controller.dart';
import 'package:bookapp/details.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class FreePage extends StatefulWidget {
  const FreePage({super.key});

  @override
  State<FreePage> createState() => _FreePageState();
}

class _FreePageState extends State<FreePage> {
  late final BookController _bookController;

  @override
  void initState() {
    _bookController = Get.put(BookController());
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return SingleChildScrollView(
      child: Column(crossAxisAlignment:
CrossAxisAlignment.start, children: [
        Padding(
          padding: const EdgeInsets.only(top: 14.0, left: 10,
bottom: 10),
          child: Text(
            "Free Books",
```

```
          style: TextStyle(fontSize: 30),
        ),
      ),
      SingleChildScrollView(
        child: SafeArea(
          child: GetBuilder(
              init: _bookController,
              builder: (controller) {
                return RefreshIndicator(
                  onRefresh: () => controller.getAllBooks(),
                  child: Column(
                    children: [
                      GetBuilder<BookController>(
                        init: _bookController,
                        builder: (controller) {
                          int numberOfElements =
                              10; // Change this value to the desired
number of elements
                          List randomElements = [];

                          for (int i = 0; i < numberOfElements; i++)
{
                            int randomIndex =

Random().nextInt(controller.books.length);

randomElements.add(controller.books[randomIndex]);
                          }
                          return controller.isLoading
                              ? const Center(
                                  child: CircularProgressIndicator(),
                                )
                              : GridView.builder(
                                  shrinkWrap: true,
                                  scrollDirection: Axis.vertical,
                                  gridDelegate:
                                      const
SliverGridDelegateWithFixedCrossAxisCount(
```

```
                    crossAxisCount: 2,
                    mainAxisSpacing: 4,
                    crossAxisSpacing: 3,
                  ),
                  itemCount: randomElements.length,
                  itemBuilder: (context, index) {
                    return InkWell(
                      onTap: () => Get.to(
                        () => DetailsPage(
                          book: controller.books[index],
                        ),
                      ),
                      child: Container(
                        width: 300,
                        height: 300,
                        decoration: BoxDecoration(
                          image: DecorationImage(
                            image: NetworkImage(controller
                                .books[index].image ??
                              ''),
                          ),
                        ),
                      ),
                    );
                  },
                );
              },
            )
          ],
        ),
      );
    }),
  ),
 ),
 ]),
 );
 }
}
```

PaidPage

```
import 'dart:math';

import
'package:bookapp/controllers/book_controller.dart';
import 'package:bookapp/details.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class PaidPage extends StatefulWidget {
  const PaidPage({super.key});

  @override
  State<PaidPage> createState() => _PaidPageState();
}

class _PaidPageState extends State<PaidPage> {
  late final BookController _bookController;

  @override
  void initState() {
    _bookController = Get.put(BookController());
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return SingleChildScrollView(
      child: Column(crossAxisAlignment:
CrossAxisAlignment.start, children: [
        Padding(
          padding: const EdgeInsets.only(top: 14.0, left: 10,
bottom: 10),
          child: Text(
            "Paid Books",
            style: TextStyle(fontSize: 30),
```

```
        ),
      ),
      SingleChildScrollView(
        child: SafeArea(
          child: GetBuilder(
              init: _bookController,
              builder: (controller) {
                return RefreshIndicator(
                  onRefresh: () => controller.getAllBooks(),
                  child: Column(
                    children: [
                      GetBuilder<BookController>(
                        init: _bookController,
                        builder: (controller) {
                          int numberOfElements =
                              15; // Change this value to the desired
number of elements
                          List randomElements = [];

                          for (int i = 0; i < numberOfElements; i++)
{
                            int randomIndex =

Random().nextInt(controller.books.length);

randomElements.add(controller.books[randomIndex]);
                          }
                          return controller.isLoading
                              ? const Center(
                                  child: CircularProgressIndicator(),
                                )
                              : GridView.builder(
                                  shrinkWrap: true,
                                  scrollDirection: Axis.vertical,
                                  gridDelegate:
                                      const
SliverGridDelegateWithFixedCrossAxisCount(
                                    crossAxisCount: 2,
```

```
                  mainAxisSpacing: 4,
                  crossAxisSpacing: 3,
                ),
                itemCount: randomElements.length,
                itemBuilder: (context, index) {
                  return InkWell(
                    onTap: () => Get.to(
                      () => DetailsPage(
                        book: controller.books[index],
                      ),
                    ),
                    child: Container(
                      width: 300,
                      height: 300,
                      decoration: BoxDecoration(
                        image: DecorationImage(
                          image: NetworkImage(controller
                              .books[index].image ??
                            ''),
                        ),
                      ),
                    ),
                  );
                },
              );
            },
          )
        ],
      ),
    );
  }),
),
),
]),
);
}
}
```

Search

```
import
'package:bookapp/controllers/book_controller.dart';
import 'package:bookapp/details.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class SearchPage extends StatefulWidget {
  const SearchPage({super.key, required this.query});

  final String query;

  @override
  State<SearchPage> createState() => _SearchPageState();
}

class _SearchPageState extends State<SearchPage> {
  late final BookController _bookController;

  @override
  void initState() {
    _bookController = Get.put(BookController());

WidgetsBinding.instance.addPostFrameCallback((timeSta
mp) {
      _bookController.searchBook(widget.query);
    });
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SingleChildScrollView(
        child: SafeArea(
          child: Column(
            children: [
```

```
            GetBuilder<BookController>(
              init: _bookController,
              builder: (controller) {
                return controller.isLoading
                    ? const Center(
                        child: CircularProgressIndicator(),
                      )
                    : GridView.builder(
                        shrinkWrap: true,
                        scrollDirection: Axis.vertical,
                        gridDelegate:
                            const
SliverGridDelegateWithFixedCrossAxisCount(
                          crossAxisCount: 2,
                          mainAxisSpacing: 4,
                          crossAxisSpacing: 3,
                        ),
                        itemCount: controller.books.length,
                        itemBuilder: (context, index) {
                          return InkWell(
                            onTap: () => Get.to(
                              () => DetailsPage(
                                book: controller.books[index],
                              ),
                            ),
                            child: Container(
                              width: 300,
                              height: 300,
                              decoration: BoxDecoration(
                                image: DecorationImage(
                                  image: NetworkImage(
                                      controller.books[index].image ?? ''),
                                ),
                              ),
                            ),
                          );
                        },
                      );
```

```
              },
            )
          ],
        ),
      ),
    ),
  );
 }
}
```

Details

```
import 'package:bookapp/models/books_model.dart';
import 'package:flutter/material.dart';

class DetailsPage extends StatelessWidget {
 final Book book;

 const DetailsPage({super.key, required this.book});

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    backgroundColor: Colors.brown,
    title: Text(
     book.title.toString(),
     style: const TextStyle(
      color: Colors.white,
     ),
    ),
   ),
   body: Column(
    children: [
     Container(
      width: double.infinity,
      height: 250,
      decoration: BoxDecoration(
       image: DecorationImage(
        image: NetworkImage(
         book.image.toString(),
        ),
        fit: BoxFit.fill,
       ),
      ),
     ),
```

```
SizedBox(
  height: 20,
),
Row(
  children: [
    Text(
      'Title: ',
      style: TextStyle(
        fontSize: 20,
      ),
    ),
    Expanded(
      child: Text(
        book.title.toString(),
        style: TextStyle(
          fontSize: 20,
          fontWeight: FontWeight.bold,
        ),
      ),
    ),
  ],
),
SizedBox(
  height: 10,
),
Row(
  children: [
    Text(
      'Author: ',
      style: TextStyle(
        fontSize: 20,
      ),
    ),
    Expanded(
      child: Text(
        book.authors.toString(),
        style: TextStyle(
          fontSize: 20,
```

```
                fontWeight: FontWeight.bold,
              ),
            ),
          ),
        ],
      ),
      SizedBox(
        height: 10,
      ),
      Row(
        children: [
          Text(
            'Download: ',
            style: TextStyle(
              fontSize: 20,
            ),
          ),
          Expanded(
            child: Text(
              book.url.toString(),
              style: TextStyle(
                fontSize: 20,
                fontWeight: FontWeight.bold,
              ),
            ),
          ),
        ],
      ),
    ],
  ),
 );
 }
}
```

Books_models

```
// To parse this JSON data, do
//
//     final bookModel = bookModelFromJson(jsonString);

import 'dart:convert';

BookModel bookModelFromJson(String str) =>
BookModel.fromJson(json.decode(str));

String bookModelToJson(BookModel data) =>
json.encode(data.toJson());

class BookModel {
  List<Book>? books;

  BookModel({
    this.books,
  });

  factory BookModel.fromJson(Map<String, dynamic>
json) => BookModel(
        books: json["books"] == null
            ? []
            : List<Book>.from(json["books"]!.map((x) =>
Book.fromJson(x))),
      );

  Map<String, dynamic> toJson() => {
        "books": books == null
            ? []
            : List<dynamic>.from(books!.map((x) =>
x.toJson())),
      };
}
```

```
class Book {
  String? id;
  String? title;
  String? subtitle;
  String? authors;
  String? image;
  String? url;

  Book({
    this.id,
    this.title,
    this.subtitle,
    this.authors,
    this.image,
    this.url,
  });

  factory Book.fromJson(Map<String, dynamic> json) =>
Book(
        id: json["id"],
        title: json["title"],
        subtitle: json["subtitle"],
        authors: json["authors"],
        image: json["image"],
        url: json["url"],
      );

  Map<String, dynamic> toJson() => {
        "id": id,
        "title": title,
        "subtitle": subtitle,
        "authors": authors,
        "image": image,
        "url": url,
      };
}
```

Book_controller

```
import 'dart:convert';

import 'package:bookapp/models/books_model.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:http/http.dart' as http;

class BookController extends GetxController {
  bool isLoading = false;
  var books = <Book>[];

  void load() {

WidgetsBinding.instance.addPostFrameCallback((timeSta
mp) {
      isLoading = !isLoading;
      refresh();
    });
  }

  @override
  void onInit() async {
    getAllBooks();
    super.onInit();
  }

  Future<String> getAllBooks() async {
    try {
      load();
      var response = await http.get(
        Uri.parse('https://www.dbooks.org/api/recent'),
      );
      if (json.decode(response.body)['status'] == 'ok') {
        print("books:
${json.decode(response.body)['books']}");
```

```
      var content = json.decode(response.body)['books'];
      for (var item in content) {
        books.add(Book.fromJson(item));
      }
      load();
      return "Success";
    } else {
      print(json.decode(response.body));
      load();
      return json.decode(response.body).toString();
    }
  } catch (e) {
    print(e.toString());
    load();
    return "Something went wrong";
  }
 }

 Future<String> searchBook(String query) async {
  try {
    var response = await http.get(

Uri.parse('https://www.dbooks.org/api/search/$query'),
    );
    if (json.decode(response.body)['status'] == 'ok') {
      print("books:
${json.decode(response.body)['books']}");
      var content = json.decode(response.body)['books'];
      for (var item in content) {
        books.add(Book.fromJson(item));
      }
      load();
      return "Success";
    } else {
      print(json.decode(response.body));
      load();
      return json.decode(response.body).toString();
    }
```

```
    } catch (e) {
      print(e.toString());
      load();
      return "Something went wrong";
    }
  }
}
```

User Guide

Welcome to the AppBook app! This guide will help you make the most of your reading experience and navigate the app's features.

Table of Contents

Developer's Guide

Prerequisites:

Before you begin developing for "**App-book**" in Flutter, make sure you have the following prerequisites:

- Flutter SDK and Dart installed
- Development environment (e.g., Android Studio, VS Code)
- Git for version control
- Access to necessary APIs (e.g., Google Books API, Stripe API)

Project Structure:

App-book Flutter app follows a standard project structure:

- `lib/`: Contains the Dart code for the app.
- `assets/`: Stores static assets like images and fonts.
- `test/`: For unit and widget tests.
- `android/` and `ios/`: Native code and configuration for Android and iOS.

Technologies and Libraries:

Key technologies and libraries used in the App-book Flutter app include:

- Flutter for cross-platform development.
- Dart for app logic.
- Provider or Bloc for state management.
- HTTP package for API requests.
- Shared preferences for local storage.

Getting Started

1. Clone the repository from GitHub.
2. Install Flutter dependencies using `flutter pub get`.
3. Configure API keys and other environment-specific settings.
4. Run the app using `flutter run`.

App Architecture

App-book follows a modular architecture, typically utilizing one of the recommended Flutter state management approaches (Provider, Bloc, etc.). The app architecture consists of:

- **UI Layer:** Widgets for the user interface.
- **Business Logic:** Handling data, API calls, and state management.
- **Data Layer:** Managing data sources (APIs, local storage).

Authentication

If the app includes user accounts, authentication can be implemented using Firebase Authentication or other methods. Ensure secure authentication practices.

API Integration

- Google Books API: Used for fetching book information and covers.
- Stripe API: Integrated for payment processing in the Paid Books section.

UI Components

The app's UI includes:

- `BookList`: Displays a list of books with details.
- `SearchBox`: Provides search functionality.
- `UserShelf`: Manages user bookshelves and personal library.

Testing

Testing is crucial to ensure app stability. Unit tests are written for logic, and widget tests for UI components. Use `flutter test` to run tests.

Deployment

When deploying the app to production, consider hosting options and configuring build settings for both Android and iOS platforms.

Support and Troubleshooting

For support and troubleshooting, refer to the documentation or reach out to the development team. Document common issues and resolutions in the `docs/` folder.

Future Development

Future development plans include enhancing the app's search functionality, improving performance, and expanding the book catalog. Collaboration with the development team is essential for app growth.