

# EPROJECT CITIGUIDE APP

**TEAM MEMBERS NAME :**

- **AAMNA FAISAL**
- **MUHAMMAD SAHIL**
- **MUHAMMAD AMMAR**
- **ANAS SHAHID**

**INSTRUCTOR NAME :  
SIR RIZWAN**



**CITIGUIDE APP**

# ACKNOWLEDGEMENT

I want to extend my heartfelt appreciation to all those who have stood by me morally and contributed to the success of the project.

I am grateful to the eProject Team based at the Head Office for their invaluable guidance and assistance. Additionally, I extend my thanks to the entire staff at my center for not only providing me with the opportunity to collaborate on this project but also for their unwavering support and encouragement throughout the journey.

My sincere gratitude goes out to the eProject Team from Aptech Head Office and my project mentor at the organization for their invaluable guidance and assistance in completing this project.

Finally, I wish to express my deep appreciation to all my colleagues for their valuable insights and helpful feedback, which greatly contributed to the project's success

# INTRODUCTION :

The City Guide mobile app is a digital platform designed to provide residents and tourists with valuable information and recommendations for exploring and enjoying a specific city. It serves as a user-friendly, all-in-one solution for discovering local attractions, restaurants, events, accommodations, and more. The app leverages the power of mobile technology to enhance the city exploration experience and make it more accessible to a wide range of users.

# REQUIREMENT SPECIFICATIONS:

## User Registration and Authentication:

- a. Users can create accounts.
- b. Users can log in securely to access personalized features and settings.
- c. Password reset option should be available.

## City Selection:

- a. Users can browse and select a city from a list of available cities.
- b. The app provides information about each city, including descriptions and images.

## Attraction Listings:

- a. The app displays a list of popular attractions, restaurants, hotels, and events in the selected city.
- b. Users can filter and sort attraction listings based on categories and ratings.
- c. Each listing includes the attraction's name, image, description, contact information, opening hours, and user ratings.

### **Detailed Information:**

- a. Users can view detailed information about each attraction by tapping on a listing.
- b. Detailed information includes additional images, location on the map, user reviews, and a link to the attraction's website.

### **User Reviews and Ratings:**

- a. Users can leave reviews and ratings for attractions they have visited.
- b. Reviews should include text comments and star ratings.
- c. Users can like helpful reviews.

### **Search Functionality:**

- a. Users can search for specific attractions, restaurants, or events by name or keywords.
- b. The app provides search filters to help users find what they're looking for.

### **User Profile and Preferences:**

- a. Users can view and edit their profiles, including their name, profile picture, and contact information.
- b. Users can set their preferences, including their favorite attractions and notification settings.

## **Admin Dashboard:**

- a. An admin dashboard should be available to manage attraction listings, reviews, and notifications.
- b. Admins can add, edit, or remove attractions, events, and other content.

## **Non-Functional Requirements-**

**Responsiveness:** The app should respond to user interactions within 1-2 seconds, ensuring a smooth and lag-free experience.

**Loading Time:** The app's initial loading time should be minimized to ensure users can access it quickly.

**User Interface:** The app's user interface should be intuitive, following best design practices for mobile apps to ensure ease of use.

**Accessible:** The application should have clear and legible fonts, user-interface elements, and navigation elements.

**User-friendly:** The application should be easy to navigate with clear menus and other elements and easy to understand.

**Operability:** The application should operate in a reliably efficient manner.

**Error Handling:** Implement robust error handling to provide clear error messages to users and gracefully handle unexpected situations.

**Scalability:** The application architecture and infrastructure should be designed to handle increasing user traffic, data storage, and feature expansions.

**Security:** The application should implement adequate security measures. For example, only registered users can access certain features.

**User Documentation:** Provide user guides, FAQs, and tutorials to help users understand and navigate the application.

**Developer Documentation:** maintain developer documentation to assist in further development and maintenance.

**Video:** Provide video displaying complete working of the application.

# USER INTERFACE :

SPLASH  
SCREEN

REGISTRATION  
FORM

LOGIN FORM

HOME PAGE

VIEW SCREEN

POST VIEW

SEARCH FILTER

# ADMIN PANEL :

ADMIN  
LOGIN

ADMIN  
DASHBOARD

MANAGE  
DETAILS

VIEW  
DETAILS

DELETE  
DETAILS

EDIT  
DETAILS



The screenshot shows the Android Studio interface. On the left is the Project Navigational Bar, which includes sections for Project, Resource Manager, Project, Commit, Pull Requests, Bookmarks, and Variants. The 'Project' section is expanded, showing the project structure: project02 (containing .dart\_tool, idea, android [project02\_android], assets, build, firebaseWork, ft, ios, lib, test, web, windows, flutter-plugins, flutter-plugins-dependencies, .gitignore, .metadata, analysis\_options.yaml, and project02.iml). The 'lib' folder is also expanded, showing auth, dashboard, screens, and widgets, along with CMakeLists.txt and firebase\_options.dart. The 'test' folder is currently selected. On the right is the main editor window displaying the main.dart file. The code is as follows:

```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:project02/auth/splash_screen.dart';
import 'package:project02/dashboard/admin/dashboard.dart';
import 'firebase_options.dart';

void main() async {
    WidgetsFlutterBinding.ensureInitialized();
    await Firebase.initializeApp(
        options: DefaultFirebaseOptions.currentPlatform,
    );
    runApp(MyApp());
}

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

    // This widget is the root of our application.
    @override
    Widget build(BuildContext context) {
        return MaterialApp(
            title: 'Flutter Demo',
            theme: ThemeData(
                colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
                useMaterial3: true,
            ), // ThemeData
            home: WelcomeScreen(), // MaterialApp
        );
    }
}
```

Below the editor are several tabs: Git, TODO, Profiler, Problems, Terminal, Logcat, Services, App Quality Insights, App Inspection, Run, and Dart Analysis.

This screenshot shows the same Android Studio environment as the first one, but with a different set of tabs at the bottom: Project, Resource Manager, Project, Commit, Pull Requests, Bookmarks, and Variants. The Project tab is selected. The rest of the interface is identical to the first screenshot, including the Project Navigational Bar on the left and the code editor on the right displaying the main.dart file.

project02 lib auth splash\_screen.dart

```
import 'package:flutter/material.dart';
import 'package:project02/auth/authscreen.dart';

class WelcomeScreen extends StatefulWidget {
  const WelcomeScreen({Key? key}) : super(key: key);

  @override
  _WelcomeScreenState createState() => _WelcomeScreenState();
}

class _WelcomeScreenState extends State<WelcomeScreen>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller;
  late Animation<double> _fadeAnimation;
  late Animation<Offset> _slideAnimation;
  late Animation<double> _scaleAnimation;

  @override
  void initState() {
    super.initState();

    _controller = AnimationController(
      vsync: this,
      duration: Duration(seconds: 1),
    ); // AnimationController
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Welcome',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        body: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Image.asset('assets/images/logo.png'),
              Text('Welcome to My App'),
              Text('Version 1.0'),
            ],
          ),
        ),
      ),
    );
  }
}
```

Resource Manager

- project02 C:\Users\DELL\AndroidStudioProjects\project02\_android
- > dart\_tool
- > idea
- > android [project02\_android]
- > assets
- > build
- > firebaseWork
- > fit
- > ios
- > lib
  - auth
    - adminlogin.dart
    - authscreen.dart
    - homescreen.dart
    - splash\_screen.dart
  - Welcome.dart
  - dashboard
  - screens
  - widgets
    - CMakeLists.txt
    - firebase\_options.dart
    - main.dart
- > linux
- > macos
- > test
- > web
- > windows
- flutter-plugins

Project

Commit

Pull Requests

Bookmarks

Id Variants

Git TODO Profiler Problems Terminal App Quality Insights Logcat Services App Inspection Dart Analysis Layout Inspector

project02 lib auth splash\_screen.dart

```
); // AnimationController

_fadeAnimation = Tween<double>(begin: 0.0, end: 1.0).animate(
  CurvedAnimation(parent: _controller, curve: Curves.easeIn),
);

_slideAnimation = Tween<Offset>(begin: Offset(0, 1), end: Offset.zero)
  .animate(CurvedAnimation(parent: _controller, curve: Curves.easeInOut));

_scaleAnimation = Tween<double>(begin: 0.5, end: 1.0).animate(
  CurvedAnimation(parent: _controller, curve: Curves.elasticOut),
);

_controller.forward();
}

@Override
void dispose() {
  _controller.dispose();
  super.dispose();
}

@Override
Widget build(BuildContext context) {
  return MaterialApp(
    title: 'Welcome',
    theme: ThemeData(
      primarySwatch: Colors.blue,
    ),
    home: Scaffold(
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Image.asset('assets/images/logo.png'),
            Text('Welcome to My App'),
            Text('Version 1.0'),
          ],
        ),
      ),
    ),
  );
}
```

Resource Manager

- project02 C:\Users\DELL\AndroidStudioProjects\project02\_android
- > dart\_tool
- > idea
- > android [project02\_android]
- > assets
- > build
- > firebaseWork
- > fit
- > ios
- > lib
  - auth
    - adminlogin.dart
    - authscreen.dart
    - homescreen.dart
    - splash\_screen.dart
  - Welcome.dart
  - dashboard
  - screens
  - widgets
    - CMakeLists.txt
    - firebase\_options.dart
    - main.dart
- > linux
- > macos
- > test
- > web
- > windows
- flutter-plugins

Project

Commit

Pull Requests

Bookmarks

Id Variants

Git TODO Profiler Problems Terminal App Quality Insights Logcat Services App Inspection Dart Analysis Layout Inspector

The screenshot shows the Android Studio interface with the following details:

- Project Bar:** Shows the project structure: `project02` > `lib` > `auth`. The file `splash_screen.dart` is currently selected.
- Resource Manager:** Shows the file tree for `auth`, including `adminlogin.dart`, `authscreen.dart`, `homescreen.dart`, `splash_screen.dart`, `Welcome.dart`, `dashboard`, `screens`, `widgets`, `CMakeLists.txt`, `firebase_options.dart`, `main.dart`, `linux`, `macos`, `test`, `web`, `windows`, and `flutter-plugins`.
- Code Editor:** Displays the `splash_screen.dart` code. The code defines a `Widget` named `splash_screen` that returns a `MaterialApp` with a `title` of 'Welcome'. It uses a `Scaffold` and a `Stack` to contain a background image and a `Container`. The `Container` has a `BoxDecoration` with a black color and 80% opacity, and a `DecorationImage` with a `AssetImage` of 'city1.jpg' (replaced with your image path). The `colorFilter` is set to `ColorFilter.mode(Colors.black.withOpacity(1), BlendMode.dstAtTop)`. The `Container` also has a `Column` as its child.
- Toolbars and Bottom Navigation:** Includes tabs for `Git`, `TODO`, `Profiler`, `Problems`, `Terminal`, `App Quality Insights`, `Logcat`, `Services`, `App Inspection`, and `Dart Analysis`. A `Layout Inspector` icon is also present.

```
Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Expanded(
      child: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            SlideTransition(
              position: _slideAnimation,
              child: ScaleTransition(
                scale: _scaleAnimation,
                child: FadeTransition(
                  opacity: _fadeAnimation,
                ), // FadeTransition
              ), // ScaleTransition
            ), // SlideTransition
            SizedBox(height: 20),
            FadeTransition(
              opacity: _fadeAnimation,
              child: Text(
                'CITY GUIDE APP',
                style: TextStyle(
                  fontSize: 40,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  ],
);
```

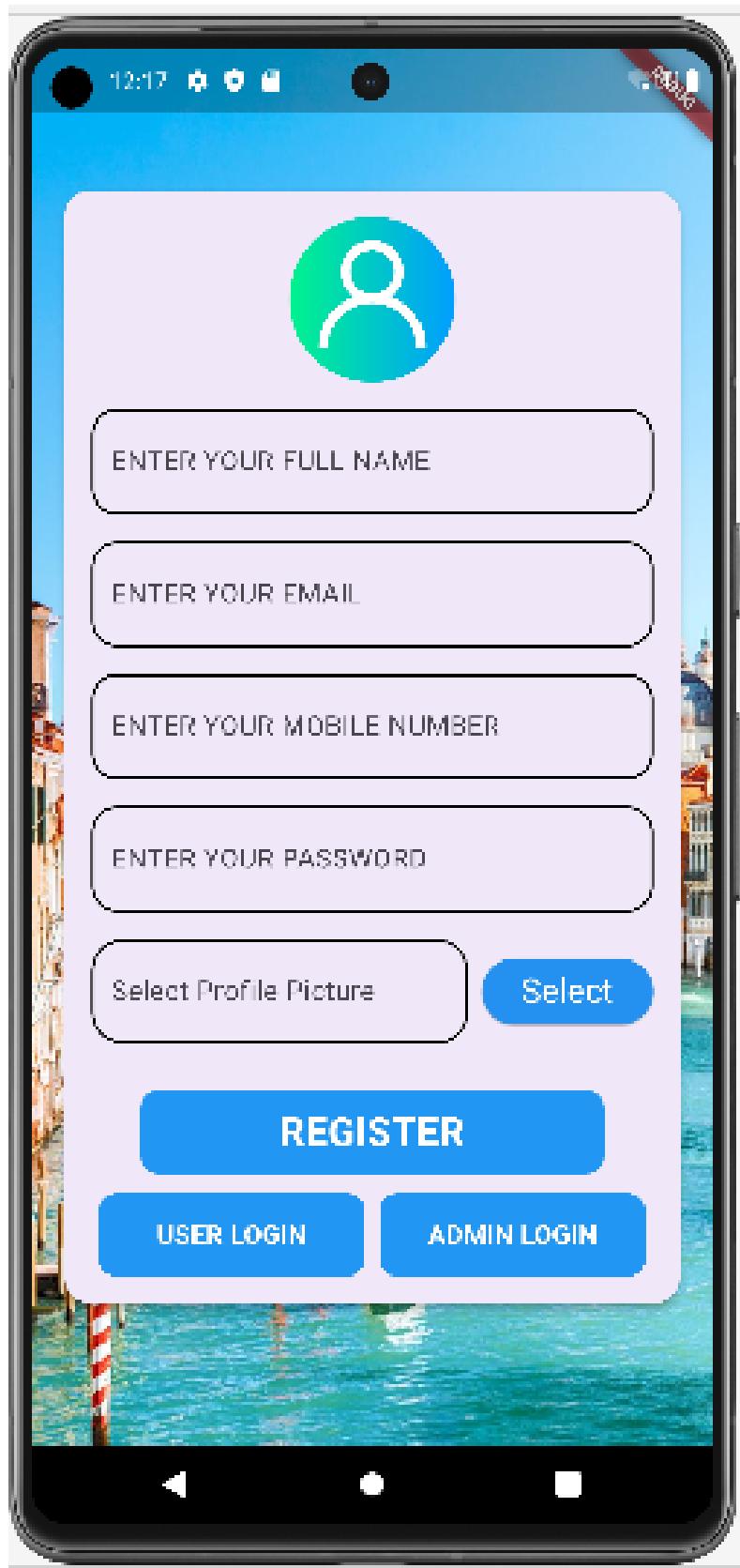
The screenshot shows the Android Studio interface with the following details:

- Project Tree:** On the left, the project structure is shown under the "lib" folder. The "auth" and "splash\_screen.dart" files are highlighted.
- Editor:** The main editor area displays the code for "splash\_screen.dart".
- Code Content:**

```
'CITY GUIDE APP',
style: TextStyle(
    fontSize: 40,
    fontWeight: FontWeight.bold,
    color: Colors.black,
    fontStyle: FontStyle.italic
), // TextStyle
), // Text
), // FadeTransition
SizedBox(height: 10),
SizedBox(height: 30),
FadeTransition(
    opacity: _fadeAnimation,
    child: Padding(
        padding: const EdgeInsets.only(right: 10.0, left: 12.0, top: 16.0),
        child: Center(
            child: Text(
                'Discover cities like never before with our City Guide App. Whether you are exploring hidden
                style: TextStyle(
                    fontSize: 20,
                    color: Colors.black87,
                    fontWeight: FontWeight.bold,
                    fontStyle: FontStyle.italic,
                ), // TextStyle
            ), // Text
        ), // Center
    ), // Padding
), // FadeTransition
), // Container
```
- Bottom Navigation:** Includes tabs for Git, TODO, Profiler, Problems, Terminal, App Quality Insights, Logcat, Services, App Inspection, and Dart Analysis.
- Right Sidebar:** Shows navigation icons for 20, X, 3, and ^.

The screenshot shows the Android Studio interface with the following details:

- Project Bar:** Shows the project structure: `project02` > `lib` > `auth` > `splash_screen.dart`.
- Resource Manager:** Shows various files and folders like `dart_tool`, `idea`, `assets`, `build`, `firebaseWork`, `fl`, `ios`, `lib`, `auth`, `adminlogin.dart`, `authscreen.dart`, `homescreen.dart`, `splash_screen.dart`, `Welcome.dart`, `dashboard`, `screens`, `widgets`, `CMakeLists.txt`, `Firebase_options.dart`, `main.dart`, `linux`, `macos`, `test`, `web`, `windows`, and `flutter-plugins`.
- Code Editor:** The `splash_screen.dart` file is open. The code defines a `Column` widget with a `SizedBox` (height: 50), a `Text` widget with bold italic font, a `Center` widget, a `Padding` widget, a `FadeTransition`, another `SizedBox` (height: 20), and a `Row` widget. The `Row` has `mainAxisAlignment: MainAxisAlignment.spaceEvenly`, `children: [`, a `SlideTransition` with `position: _slideAnimation`, a `Containen` child with `width: 340`, `height: 60`, and an `ElevatedButton` child with an `onPressed` callback that pushes a `MaterialPageRoute` to a `ProjectForm`.
- Bottom Navigation:** Includes Git, TODO, Profiler, Problems, Terminal, App Quality Insights, Logcat, Services, App Inspection, Dart Analysis, and Layout Inspector.



The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the `authscreen.dart` file under the `main.dart` tab. The code defines a `MyApp02` widget that returns a `MaterialApp` with a `AuthScreen` as the home screen. The `AuthScreen` widget uses a `Scaffold` with a `Container` that has a `BoxDecoration` and a `DecorationImage` with an asset image from the `assets/images/bg.jpg` file.

```
import 'dart:io';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:file_picker/file_picker.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:flutter/material.dart';
import 'package:project02/auth/adminLogin.dart';
import 'package:project02/screens/home_page.dart';

class MyApp02 extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: AuthScreen(),
    ); // MaterialApp
  }
}

class AuthScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        decoration: BoxDecoration(
          image: DecorationImage(
            image: AssetImage('assets/images/bg.jpg'),
            fit: BoxFit.cover,
          ), // DecorationImage
        ), // BoxDecoration
        child: Row(
          children: [
            Padding(
              padding: const EdgeInsets.only(right: 40.0, left: 40.0, top: 200.0),
            ), // Padding
          ],
        ), // Row
      ), // Container
    ); // Scaffold
  }
}
```

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the `authscreen.dart` file under its own tab. The code is identical to the one in the previous screenshot, defining the `AuthScreen` widget with its `Scaffold`, `Container`, `BoxDecoration`, and `DecorationImage`.

```
class AuthScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        decoration: BoxDecoration(
          image: DecorationImage(
            image: AssetImage('assets/images/bg.jpg'),
            fit: BoxFit.cover,
          ), // DecorationImage
        ), // BoxDecoration
        child: Row(
          children: [
            Padding(
              padding: const EdgeInsets.only(right: 40.0, left: 40.0, top: 200.0),
            ), // Padding
          ],
        ), // Row
      ), // Container
    ); // Scaffold
  }
}
```

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the `authscreen.dart` file under the `main.dart` tab. The code implements a registration form with fields for email, password, full name, and phone number. It uses TextEditingController for each field and FirebaseAuth to create a user with email and password. It also handles image selection and upload to Firebase Storage.

```
class RegisterForm extends StatefulWidget {
  @override
  _RegisterFormState createState() => _RegisterFormState();
}

class _RegisterFormState extends State<RegisterForm> {
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  final TextEditingController _fullNameController = TextEditingController();
  final TextEditingController _phoneNumberController = TextEditingController();

  String? _filePath; // Add a variable to store the file path
  File? _image; // Add a variable to store the selected image file

  void register() async {
    try {
      UserCredential userCredential =
        await FirebaseAuth.instance.createUserWithEmailAndPassword(
          email: _emailController.text,
          password: _passwordController.text,
        );

      // Upload image to Firebase Storage
      String? imageUrl;
      if (_image != null) {
        final Reference storageReference = FirebaseStorage.instance
          .ref()
          .child('profile_pictures')
          .child('${userCredential.user!.uid}.jpg');

        await storageReference.putFile(_image!);
        imageUrl = await storageReference.getDownloadURL();
      }

      await FirebaseFirestore.instance
        .collection('users')
        .doc(userCredential.user!.uid)
        .set({
          'email': _emailController.text,
          'fullName': _fullNameController.text,
          'phoneNumber': _phoneNumberController.text,
          'file_path': _filePath, // Save the file path in Firestore
          'profile_picture': imageUrl, // Save the profile picture URL in Firestore
        });

      print('Registration successful: ${userCredential.user?.uid}');
    } catch (e) {
      print(e);
    }
  }
}
```

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the `authscreen.dart` file in its own tab. This version of the code is identical to the one in the first screenshot, implementing a registration form with image upload and Firestore storage.

```
// Upload image to Firebase Storage
String? imageUrl;
if (_image != null) {
  final Reference storageReference = FirebaseStorage.instance
    .ref()
    .child('profile_pictures')
    .child('${userCredential.user!.uid}.jpg');

  await storageReference.putFile(_image!);
  imageUrl = await storageReference.getDownloadURL();
}

await FirebaseFirestore.instance
  .collection('users')
  .doc(userCredential.user!.uid)
  .set({
    'email': _emailController.text,
    'fullName': _fullNameController.text,
    'phoneNumber': _phoneNumberController.text,
    'file_path': _filePath, // Save the file path in Firestore
    'profile_picture': imageUrl, // Save the profile picture URL in Firestore
  });

print('Registration successful: ${userCredential.user?.uid}');
```

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the `authscreen.dart` file under the `main.dart` tab. The code handles file picking and printing registration success/failure messages.

```
235     print('Registration successful: ${userCredential.user?.uid}');
236   } catch (e) {
237     print('Registration failed: $e');
238   }
239 }
240
241 void _selectFile() async {
242   try {
243     FilePickerResult? result = await FilePicker.platform.pickFiles();
244
245     if (result != null) {
246       File file = File(result.files.single.path!);
247       setState(() {
248         _filePath = file.path;
249         _image = file;
250       });
251     } else {
252     }
253   } catch (e) {
254     print('File picking failed: $e');
255   }
256 }
257
258 @override
259 Widget build(BuildContext context) {
```

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the `authscreen.dart` file under its own tab. The code defines a scaffold with a background image and a card containing a column with an image asset.

```
258 @override
259 Widget build(BuildContext context) {
260   return Scaffold(
261     body: Container(
262       decoration: BoxDecoration(
263         image: DecorationImage(
264           image: AssetImage('assets/images/bg.jpg'), // Replace with your image path
265           fit: BoxFit.cover,
266         ), // DecorationImage
267       ), // BoxDecoration
268       child: Padding(
269         padding: const EdgeInsets.all(16.0),
270       child: Center(
271         child: Card(
272           elevation: 5,
273           shape: RoundedRectangleBorder(
274             borderRadius: BorderRadius.circular(15.0),
275           ), // RoundedRectangleBorder
276           child: Padding(
277             padding: const EdgeInsets.all(16.0),
278           child: SingleChildScrollView(
279             child: Column(
280               mainAxisAlignment: MainAxisAlignment.center,
281               children: [
282                 Image.asset(
```

The screenshot shows the Android Studio interface with the project navigation bar on the left and the code editor on the right. The project tree on the left shows a folder structure for 'project02' with subfolders like 'assets', 'build', 'lib', and 'test'. The 'auth' folder under 'lib' contains several files: 'adminlogin.dart', 'authscreen.dart', 'homescreen.dart', 'splash\_screen.dart', and 'Welcome.dart'. The 'main.dart' file is open in the code editor, and the 'authscreen.dart' file is currently selected. The code in 'authscreen.dart' is as follows:

```
201 current_line
282 Image.asset(
283   'assets/images/login.png',
284   height: 100,
285   width: 100,
286 ), // Image.asset
287 SizedBox(height: 16),
288 // Existing form fields...
289 TextFormField(
290   controller: _fullNameController,
291   decoration: InputDecoration(
292     labelText: 'ENTER YOUR FULL NAME',
293     focusedBorder: OutlineInputBorder(
294       borderRadius: BorderRadius.circular(16),
295       borderSide: const BorderSide(color: Colors.black, width: 2.5),
296     ), // OutlineInputBorder
297     enabledBorder: OutlineInputBorder(
298       borderRadius: BorderRadius.circular(16),
299       borderSide: const BorderSide(color: Colors.black, width: 1.5),
300     ), // OutlineInputBorder
301     ), // InputDecoration
302     ), // TextFormField
303     SizedBox(height: 16),
304 // Existing form fields...
305     TextFormField(
306   controller: _emailController,
```

This screenshot shows the same Android Studio interface, but the project tree on the left indicates that 'authscreen.dart' is located directly in the 'lib' directory of the 'project02' project. The code in 'authscreen.dart' is identical to the one shown in the previous screenshot.

```
305 current_line
306 TextFormField(
307   controller: _emailController,
308   decoration: InputDecoration(
309     labelText: 'ENTER YOUR EMAIL',
310     focusedBorder: OutlineInputBorder(
311       borderRadius: BorderRadius.circular(16),
312       borderSide: const BorderSide(color: Colors.black, width: 2.5),
313     ), // OutlineInputBorder
314     enabledBorder: OutlineInputBorder(
315       borderRadius: BorderRadius.circular(16),
316       borderSide: const BorderSide(color: Colors.black, width: 1.5),
317     ), // OutlineInputBorder
318     ), // InputDecoration
319     ), // TextFormField
320     SizedBox(height: 16),
321 // Existing form fields...
322     TextFormField(
323   controller: _phoneNumberController,
324   decoration: InputDecoration(
325     labelText: 'ENTER YOUR MOBILE NUMBER',
326     focusedBorder: OutlineInputBorder(
327       borderRadius: BorderRadius.circular(16),
328       borderSide: const BorderSide(color: Colors.black, width: 2.5),
329     ), // OutlineInputBorder
330     enabledBorder: OutlineInputBorder(
331       borderRadius: BorderRadius.circular(16),
```

```

    ), // InputDecoration
    ), // TextFormField
    SizedBox(height: 16),
    // Existing form fields...
    TextFormField(
        controller: _passwordController,
        decoration: InputDecoration(
            labelText: 'ENTER YOUR PASSWORD',
            focusedBorder: OutlineInputBorder(
                borderRadius: BorderRadius.circular(16),
                borderSide: const BorderSide(color: Colors.black, width: 2.5),
            ), // OutlineInputBorder
            enabledBorder: OutlineInputBorder(
                borderRadius: BorderRadius.circular(16),
                borderSide: const BorderSide(color: Colors.black, width: 1.5),
            ), // OutlineInputBorder
            obscureText: true,
        ), // InputDecoration
        SizedBox(height: 16),
        // File input field...
        Row(
            children: [
                Expanded(
                    child: TextFormField(

```

```

    ), // InputDecoration
    labelText: 'Select Profile Picture',
    border: OutlineInputBorder(
        borderRadius: BorderRadius.circular(20.0),
        borderSide: const BorderSide(color: Colors.black, width: 1.5),
    ), // OutlineInputBorder
    enabledBorder: OutlineInputBorder(
        borderRadius: BorderRadius.circular(20.0),
        borderSide: const BorderSide(color: Colors.black, width: 1.5),
    ), // OutlineInputBorder
    controller: TextEditingController(text: _filePath ?? ''),
    ), // TextFormField

    ), // Expanded
    SizedBox(width: 8),
    ElevatedButton(
        onPressed: _selectFile,
        style: ElevatedButton.styleFrom(
            primary: Colors.blue, // Set fill color to blue
        ),
        child: Text(
            'Select',
            style: TextStyle(
                color: Colors.white,
                fontWeight: FontWeight.normal,

```

The screenshot shows the Android Studio interface with the following details:

- Project Bar:** Shows the project structure: `Project`, `main.dart`, `authscreen.dart`, `style`, `realStyle`, `color: Colors.white, fontWeight: FontWeight.bold, fontSize: 24,`, `// Text`, `), // Text`, `), // TextButton`, `), // Padding`, `Row(`, `mainAxisAlignment: MainAxisAlignment.spaceAround,`, `children: [`, `TextButton(`, `onPressed: () {`, `Navigator.push(`, `context, MaterialPageRoute(builder: (context) => LoginForm()));`, `},`, `style: ElevatedButton.styleFrom(`, `primary: Colors.blue,`, `fixedSize: Size(160, 50),`, `shape: RoundedRectangleBorder(`, `borderRadius: BorderRadius.all(Radius.circular(10))), // RoundedRectangleBorder`, `),`, `child: Text(`, `'USER LOGIN',`, `style: TextStyle(`, `color: Colors.white, fontWeight: FontWeight.bold, fontSize: 16), // TextStyle`, `), // Text`, `), // TextButton`, `TextButton(`.
- Resource Manager:** Shows various build configurations like `project02`, `dart_tool`, `idea`, `android [project02_android]`, `assets`, `build`, `firebaseWork`, `fl`, `ios`, `lib`, `auth`, `adminlogin.dart`, `authscreen.dart`, `homescreen.dart`, `splash_screen.dart`, `Welcome.dart`, `dashboard`, `screens`, `widgets`, `CMakeLists.txt`, `firebase_options.dart`, `main.dart`, `linux`, `macos`, `test`, `web`, `windows`, and `flutter-plugins`.
- Commit:** Shows the current commit status.
- Pull Requests:** Shows the current pull requests.
- Bookmarks:** Shows the current bookmarks.
- Variants:** Shows the current variants.
- Git:** Shows the current git status.
- TODO:** Shows the current TODO items.
- Profiler:** Shows the current profiler status.
- Problems:** Shows the current problems.
- Terminal:** Shows the current terminal status.
- Logcat:** Shows the current logcat status.
- Services:** Shows the current services status.
- App Quality Insights:** Shows the current app quality insights.
- App Inspection:** Shows the current app inspection status.
- Run:** Shows the current run status.
- Dart Analysis:** Shows the current dart analysis status.

project02 > lib > auth > authscreen.dart

```
401 // EXECUTION
412 TextButton(
413   onPressed: () {
414     Navigator.push(
415       context, MaterialPageRoute(builder: (context) => AdminLogin()));
416   },
417   style: ElevatedButton.styleFrom(
418     primary: Colors.blue,
419     fixedSize: Size(160, 50),
420     shape: RoundedRectangleBorder(
421       borderRadius: BorderRadius.all(Radius.circular(10))), // RoundedRectangleBorder
422   ),
423   child: Text(
424     'ADMIN LOGIN',
425     style: TextStyle(
426       color: Colors.white, fontWeight: FontWeight.bold, fontSize: 16), // TextStyle
427     ), // Text
428   ), // TextButton
429 ],
430 ), // Row
431 ],
432 ), // Column
433 ), // SingleChildScrollView
434 ), // Padding
435 ), // Card
436 ), // Center
437 ), // Center
```

Project Manager Project Commit Pull Requests Bookmarks Variants

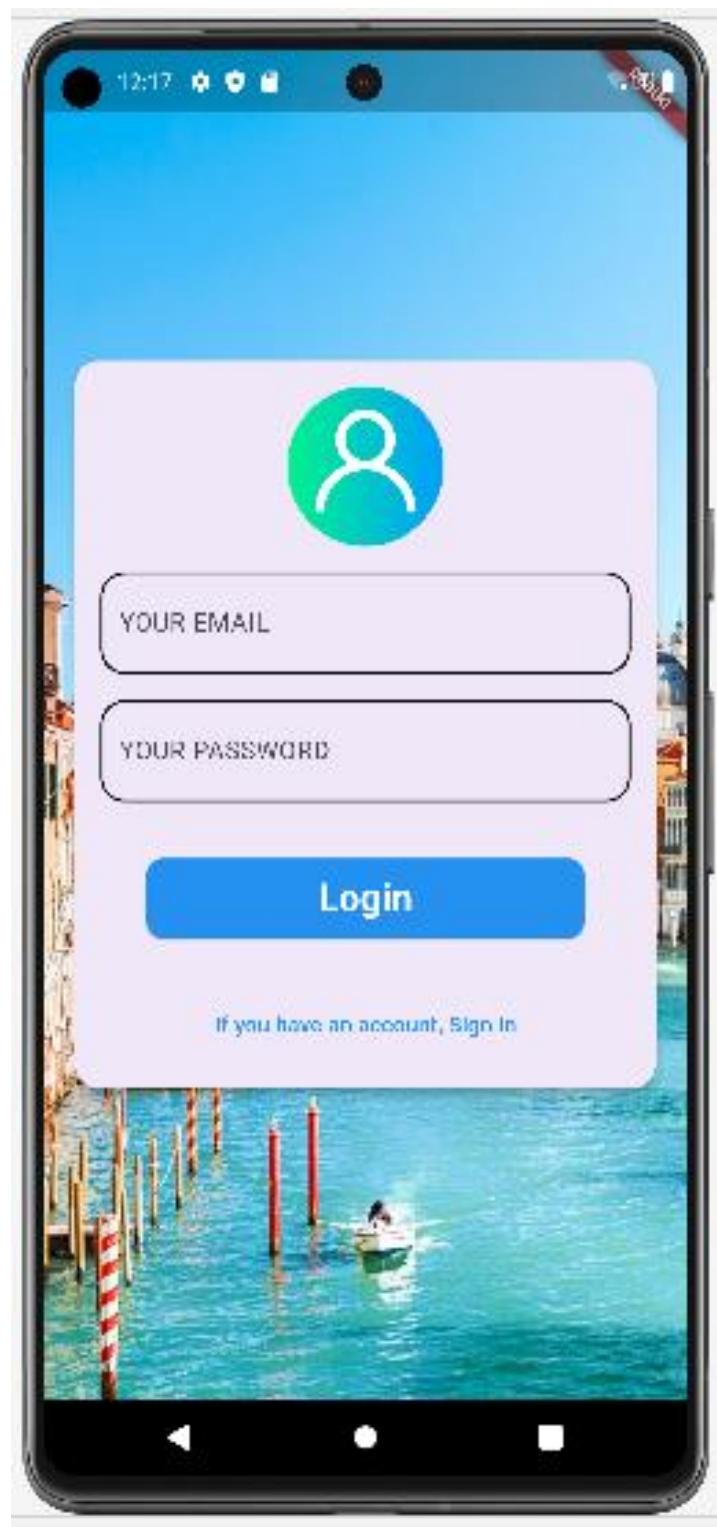
Git TODO Profiler Problems Terminal Logcat Services App Quality Insights App Inspection Run Dart Analysis Layout

project02 > lib > auth > authscreen.dart

```
401 // EXECUTION
412 TextButton(
413   onPressed: () {
414     Navigator.push(
415       context, MaterialPageRoute(builder: (context) => AdminLogin()));
416   },
417   style: ElevatedButton.styleFrom(
418     primary: Colors.blue,
419     fixedSize: Size(160, 50),
420     shape: RoundedRectangleBorder(
421       borderRadius: BorderRadius.all(Radius.circular(10))), // RoundedRectangleBorder
422   ),
423   child: Text(
424     'ADMIN LOGIN',
425     style: TextStyle(
426       color: Colors.white, fontWeight: FontWeight.bold, fontSize: 16), // TextStyle
427     ), // Text
428   ), // TextButton
429 ],
430 ), // Row
431 ],
432 ), // Column
433 ), // SingleChildScrollView
434 ), // Padding
435 ), // Card
436 ), // Center
437 ),
438 ); // Scaffold
439 }
```

Project Manager Project Commit Pull Requests Bookmarks Variants

Git TODO Profiler Problems Terminal Logcat Services App Quality Insights App Inspection Run Dart Analysis Layout



The screenshot shows the Android Studio interface with the project navigation bar on the left and the code editor on the right. The project tree on the left shows a folder structure for 'project02' with subfolders like 'assets', 'build', 'lib', and 'test'. The 'lib/auth' folder contains files such as 'adminlogin.dart', 'authscreen.dart', 'homescreen.dart', 'splash\_screen.dart', and 'Welcome.dart'. The 'lib/main.dart' file is also visible. The code editor displays the 'authscreen.dart' file, which defines a stateful widget 'LoginForm' and its state class '\_LoginFormState'. The code includes validation logic for email and password fields.

```
41: class LoginForm extends StatefulWidget {  
42:   @override  
43:   _LoginFormState createState() => _LoginFormState();  
44: }  
45:  
46: class _LoginFormState extends State<LoginForm> {  
47:   final GlobalKey<FormState> _formKey = GlobalKey<FormState>();  
48:   final TextEditingController _emailController = TextEditingController();  
49:   final TextEditingController _passwordController = TextEditingController();  
50:  
51:   String? _validateEmail(String? value) {  
52:     if (value!.isEmpty) {  
53:       return 'Please enter your email';  
54:     }  
55:     // You can add more advanced email validation logic here if needed  
56:     return null;  
57:   }  
58:  
59:   String? _validatePassword(String? value) {  
60:     if (value!.isEmpty) {  
61:       return 'Please enter your password';  
62:     }  
63:     // You can add more advanced password validation logic here if needed  
64:     return null;  
65:   }  
66:  
67: }  
68:
```

This screenshot is similar to the one above, showing the 'authscreen.dart' file in the 'main.dart' project. The code editor displays the same structure and validation logic for the login form. The project navigation bar on the left shows the same file structure as the first screenshot.

```
69: Widget build(BuildContext context) {  
70:   return Scaffold(  
71:     body: Container(  
72:       decoration: BoxDecoration(  
73:         color: Color.fromRGBO(173, 216, 230, 0.7),  
74:         image: DecorationImage(  
75:           image: AssetImage('assets/images/bg.jpg'),  
76:           fit: BoxFit.cover,  
77:         ), // DecorationImage  
78:       ), // BoxDecoration  
79:     child: Padding(  
80:       padding: const EdgeInsets.all(16.0),  
81:     child: Center(  
82:       child: Card(  
83:         elevation: 5,  
84:         shape: RoundedRectangleBorder(  
85:           borderRadius: BorderRadius.circular(15.0),  
86:         ), // RoundedRectangleBorder  
87:       child: Padding(  
88:         padding: const EdgeInsets.all(16.0),  
89:       child: SingleChildScrollView(  
90:         child: Form(  
91:           key: _formKey,  
92:           child: Column(  
93:             mainAxisAlignment: MainAxisAlignment.center,
```

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor has two tabs: 'main.dart' and 'authscreen.dart'. The 'authscreen.dart' tab is active, displaying Dart code for an authentication screen. The code includes imports for 'Image.asset', 'SizedBox', 'TextFormField', and 'OutlineInputBorder'. It defines two text input fields for email and password, each with specific styling like circular border radii and black outlines. The 'lib/auth/authscreen.dart' file is selected in the project tree.

```
image.asset('assets/images/login.png', height: 100, width: 100), // Image.asset  
SizedBox(height: 16),  
TextFormField(  
  controller: _emailController,  
  decoration: InputDecoration(  
    labelText: 'YOUR EMAIL',  
    focusedBorder: OutlineInputBorder(  
      borderRadius: BorderRadius.circular(16),  
      borderSide: const BorderSide(color: Colors.black, width: 2.5),  
    ), // OutlineInputBorder  
    enabledBorder: OutlineInputBorder(  
      borderRadius: BorderRadius.circular(16),  
      borderSide: const BorderSide(color: Colors.black, width: 1.5),  
    ), // OutlineInputBorder  
    validator: _validateEmail,  
  ), // TextFormField  
SizedBox(height: 16),  
TextFormField(  
  controller: _passwordController,  
  decoration: InputDecoration(  
    labelText: 'YOUR PASSWORD',
```

This screenshot shows the same Android Studio session with the 'authscreen.dart' file now open in its own tab. The code is identical to the previous screenshot, showing the Dart code for the authentication screen. The 'lib/auth/authscreen.dart' file is selected in the project tree.

```
decoration: InputDecoration(  
  labelText: 'YOUR PASSWORD',  
  focusedBorder: OutlineInputBorder(  
    borderRadius: BorderRadius.circular(16),  
    borderSide: const BorderSide(color: Colors.black, width: 2.5),  
  ), // OutlineInputBorder  
  enabledBorder: OutlineInputBorder(  
    borderRadius: BorderRadius.circular(16),  
    borderSide: const BorderSide(color: Colors.black, width: 1.5),  
  ), // OutlineInputBorder  
  obscureText: true,  
  validator: _validatePassword,  
), // TextFormField  
SizedBox(height: 25),  
Padding(  
  padding: const EdgeInsets.all(10.0),  
  child: ElevatedButton(  
    onPressed: () {  
      if (_formKey.currentState!.validate()) {  
        // Form is validated, perform login  
        _login();  
      }  
    },  
    style: ElevatedButton.styleFrom(  
      primary: Colors.blue,
```

The screenshot shows the Android Studio interface with the project navigation bar at the top. The left sidebar displays the project structure under 'lib/auth'. The main editor window shows the code for 'authscreen.dart'.

```
project02 / lib / auth / authscreen.dart
main.dart x authscreen.dart x
```

```
style: ElevatedButton.styleFrom(
    primary: Colors.blue,
    fixedSize: Size(280, 50),
    shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.all(Radius.circular(10)),
    ), // RoundedRectangleBorder
),
child: Text(
    'Login',
    style: TextStyle(color: Colors.white, fontWeight: FontWeight.bold, fontSize: 24),
), // Text
), // ElevatedButton
), // Padding
SizedBox(height: 20),
TextButton(
onPressed: () {
    Navigator.push(context, MaterialPageRoute(builder: (context) => RegisterForm()));
},
child: Text(
    'If you have an account, Sign In',
    style: TextStyle(color: Colors.blue),
),
), // Text
), // TextButton
], // Column
), // Form
```

The bottom navigation bar includes Git, TODO, Profiler, Problems, Terminal, Logcat, Services, App Quality Insights, App Inspection, Run, Dart Analysis, and Layout Inspector.

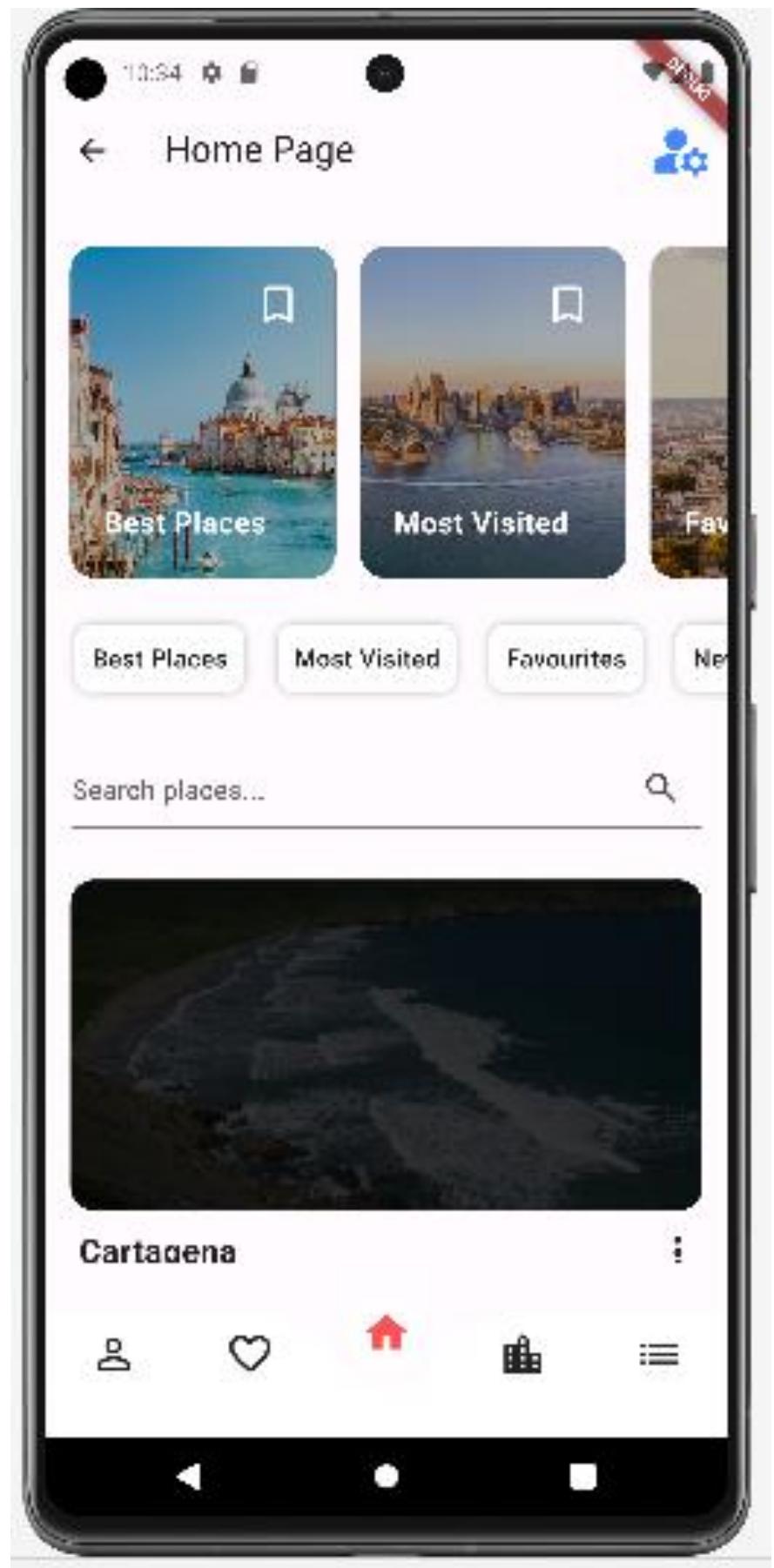
This screenshot is identical to the one above, showing the same code for 'authscreen.dart' in the 'main.dart' project. The code defines a column with a padding, a sized box, and a text button.

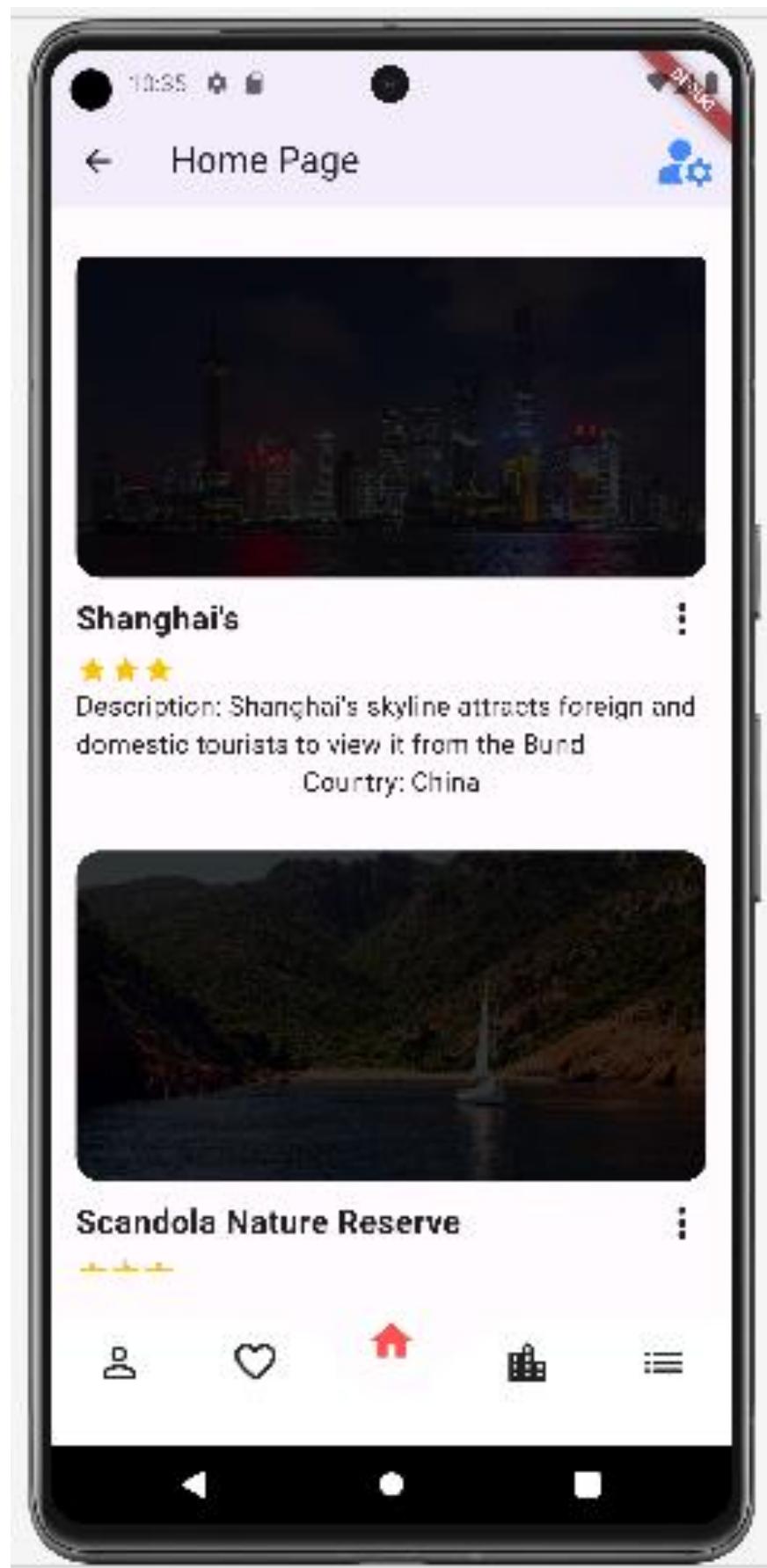
```
project02 / lib / auth / authscreen.dart
main.dart x authscreen.dart x
```

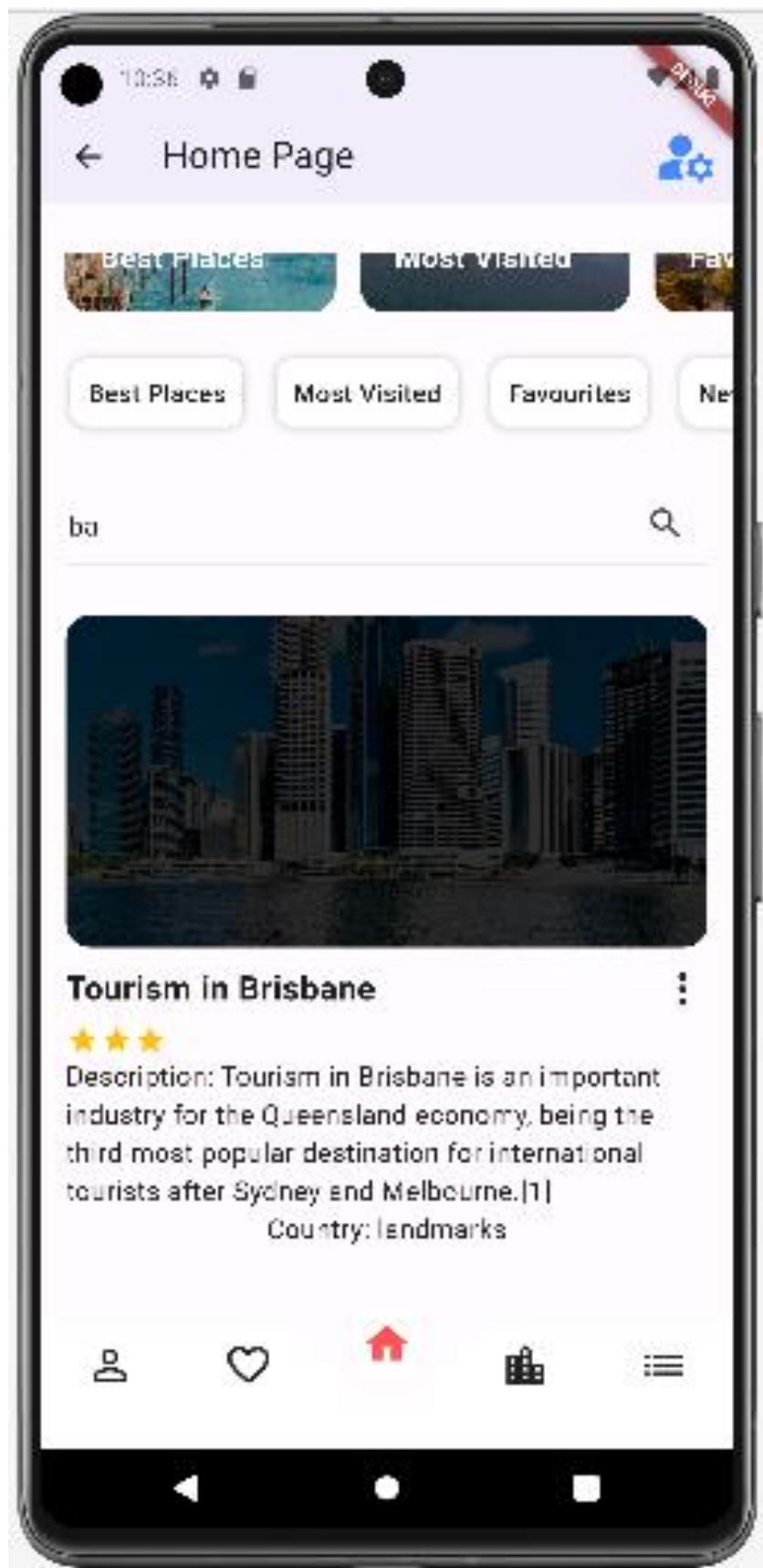
```
, // Text
], // Column
), // Form
), // SingleChildScrollView
), // Padding
), // Card
), // Center
), // Padding
), // Container
); // Scaffold
}

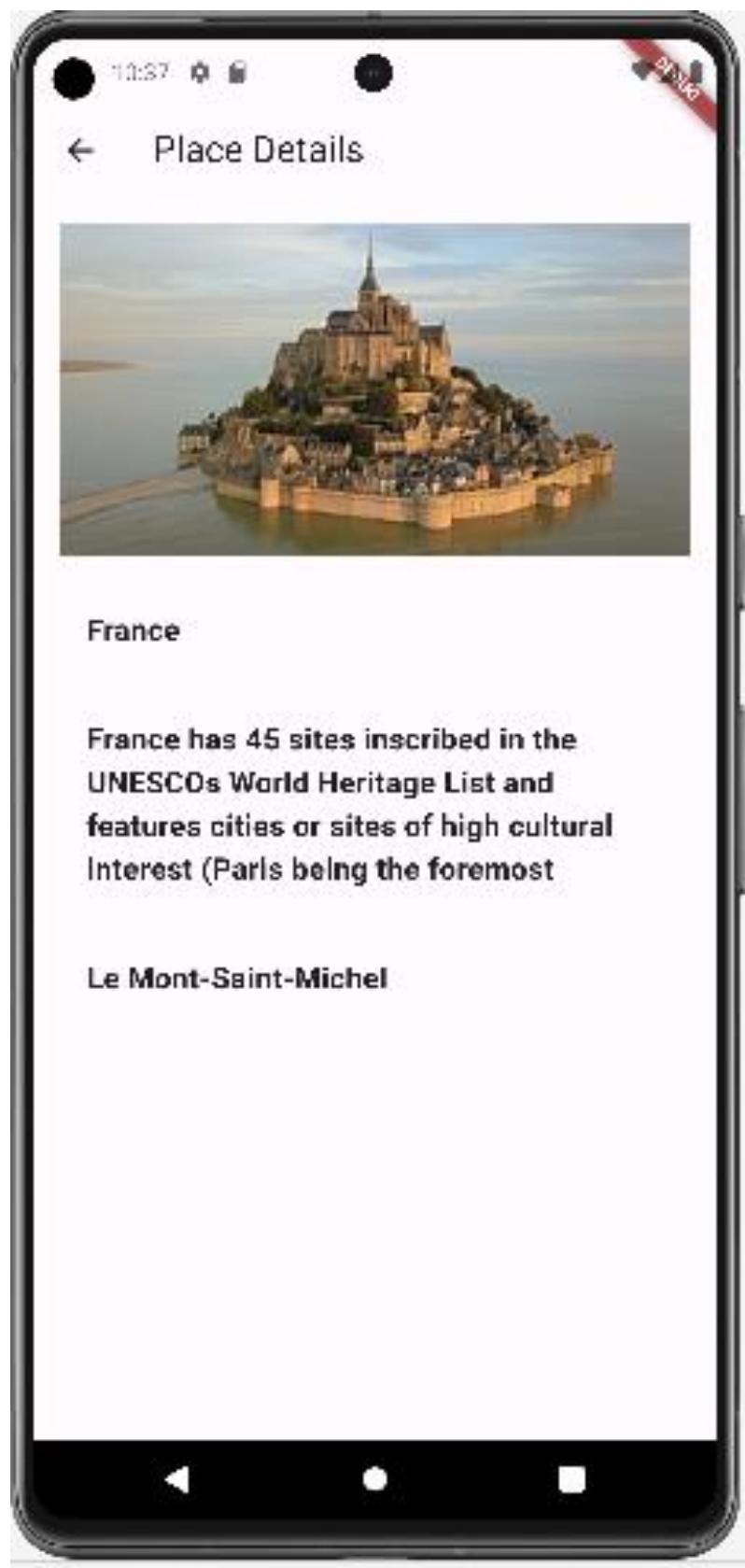
void _login() async {
UserCredential userCredential = await FirebaseAuth.instance.signInWithEmailAndPassword(
email: _emailController.text,
password: _passwordController.text,
);
if (userCredential != null) {
Navigator.push(context, MaterialPageRoute(builder: (context) => HomePage()));
}
}
```

The bottom navigation bar includes Git, TODO, Profiler, Problems, Terminal, Logcat, Services, App Quality Insights, App Inspection, Run, Dart Analysis, and Layout Inspector.









project02 / lib / screens / home\_page.dart

```
home_page.dart x main.dart x
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:project02/dashboard/admin/firestore_service.dart';
import 'package:project02/screens/post_screen.dart';
import 'package:project02/widgets/home_app_bar.dart';
import 'package:project02/widgets/home_bottom_bar.dart';

class HomePage extends StatefulWidget {
    final String? userProfileImageUrl; // Pass the user profile image URL to the HomePage

    const HomePage({Key? key, this.userProfileImageUrl}) : super(key: key);

    @override
    _HomePageState createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
    final TextEditingController _searchController = TextEditingController();
    final FirestoreService firestoreService = FirestoreService();
    var category = ['Best Places', 'Most Visited', 'Favourites', 'New Added', 'Hotels', 'Restaurants'];
    List<Map<String, dynamic>> _originalPlaces = [];

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                title: Text('Home Page'),
                actions: [
                    // Display user's profile picture in the app bar
                    if (widget.userProfileImageUrl != null)
                        CircleAvatar(
                            backgroundImage: NetworkImage(widget.userProfileImageUrl),
                        ),
                    IconButton(
                        icon: Icon(Icons.manage_accounts, size: 40, color: Colors.blueAccent),
                        onPressed: () {
                            // Navigate to the post screen
                            Navigator.push(
                                context,
                                MaterialPageRoute(builder: (context) => PostScreen()),
                            );
                        },
                    ),
                ],
            ),
            body: SafeArea(
                child: Padding(
                    padding: EdgeInsets.symmetric(vertical: 30),
                    child: SingleChildScrollView(
                        child: Column(

```

project02 / lib / screens / home\_page.dart

```
return Scaffold(
            appBar: AppBar(
                title: Text('Home Page'),
                actions: [
                    // Display user's profile picture in the app bar
                    if (widget.userProfileImageUrl != null)
                        CircleAvatar(
                            backgroundImage: NetworkImage(widget.userProfileImageUrl),
                        ),
                    IconButton(
                        icon: Icon(Icons.manage_accounts, size: 40, color: Colors.blueAccent),
                        onPressed: () {
                            // Navigate to the post screen
                            Navigator.push(
                                context,
                                MaterialPageRoute(builder: (context) => PostScreen()),
                            );
                        },
                    ),
                ],
            ),
            body: SafeArea(
                child: Padding(
                    padding: EdgeInsets.symmetric(vertical: 30),
                    child: SingleChildScrollView(
                        child: Column(

```

The screenshot shows the Android Studio interface with the following details:

- Project Bar:** Pixel 2 API 30 (mobile), main.dart, Pixel 7 Pro API 30.
- Resource Manager:** Shows the project structure:
  - Project
  - build
  - assets
  - build
  - firebaseWork
  - fl
  - ios
  - lib
    - auth
    - dashboard
    - screens
      - home\_page.dart
      - post\_screen.dart
      - welcome\_screen.dart
  - widgets
    - CMakeLists.txt
    - firebase\_options.dart
    - main.dart
  - linux
  - macos
  - test
  - web
  - windows
  - flutter-plugins
  - flutter-plugins-dependencies
  - .gitignore
  - metadata
  - analysis\_options.yaml
  - project02.iml
- Code Editor:** The code for `home_page.dart` is displayed. It contains a `SafeArea` widget with padding and a `SingleChildScrollView` containing a `Column` with a `Container` height of 200. A `ListView.builder` with 6 items is nested within this. Each item is wrapped in an `InkWell` with an `onTap` handler that pushes a `MaterialPageRoute` to the `PlaceDetailsPage`.
- Toolbars:** Standard Android Studio toolbars for file operations, search, and navigation.

The screenshot shows the Android Studio interface with the following details:

- Project Tree:** The left sidebar displays the project structure under "lib/screens". Key files shown include `home_page.dart`, `post_screen.dart`, `welcome_screen.dart`, `main.dart`, and `test`.
- Code Editor:** The main window shows the `home_page.dart` file. The code defines a `StatefulWidget` with a `State` class. It includes a `Column` widget with a `Container` and a `Spacer()`. The `Container` has a `Icon` with a `size: 30,` and a `color: Colors.white,` and a `alignment: Alignment.topRight,`.
- Toolbars and Status Bar:** The bottom of the screen features standard Android Studio toolbars for Git, TODO, Profiler, Problems, Terminal, App Quality Insights, Logcat, Services, App Inspection, and Dart Analysis. A "Layout Inspector" icon is also present.

The screenshot shows the Android Studio interface with the following details:

- Project Bar:** Shows the project structure: `project02` / `lib` / `screens` / `home_page.dart`.
- Resource Manager:** Lists files and folders: `build`, `assets`, `ios`, `lib` (containing `auth`, `dashboard`, `screens` (containing `home_page.dart`, `post_screen.dart`, `welcome_screen.dart`), `widgets`, `CMakeLists.txt`, `firebase_options.dart`, `main.dart`), `linux`, `macos`, `test` (selected), `web`, `windows`, `flutter-plugins`, `flutter-plugins-dependencies`, `gitignore`, `metadata`, `analysis_options.yaml`, and `project02.iml`.
- Code Editor:** The `home_page.dart` file is open, showing Dart code for a home page screen. The code includes imports for `Spacer`, `Container`, `Text`, `TextStyle`, `Colors`, `FontWeight`, `InkWell`, `Column`, `Container`, `ListView.builder`, `SizedBox`, and `SingleChildScrollView`. It defines a `Row` with children including `Text` and `Image`, and a `Column` with children including `Text` and `Image`.
- Bottom Navigation Bar:** Includes icons for Git, TODO, Profiler, Problems, Terminal, App Quality Insights, Logcat, Services, App Inspection, and Dart Analysis.

The screenshot shows the Android Studio interface with the following details:

- Project Tree:** The left sidebar displays the project structure under "Project". Key files shown include `android`, `assets`, `build`, `firebaseWork`, `fl`, `ios`, `lib` (containing `auth`, `dashboard`, and `screens`), `main.dart`, `post_screen.dart`, `welcome_screen.dart`, `widgets`, `CMakeLists.txt`, `firebase_options.dart`, `main.dart`, `linux`, `macos`, `test` (selected), `web`, `windows`, `flutter-plugins`, `flutter-plugins-dependencies`, `gitignore`, `metadata`, `analysis_options.yaml`, and `project02.iml`.
- Code Editor:** The main window shows the `home_page.dart` file. The code defines a horizontal scroll view with a padding of 8 units and a row of six categories. Each category is a `Container` with a white color, circular border radius of 10, and a black shadow with a blur radius of 4. The text inside each container has a font weight of `w500` and a font size of 15.
- Bottom Bar:** The navigation bar includes icons for Git, TODO, Profiler, Problems, Terminal, App Quality Insights, Logcat, Services, App Inspection, and Dart Analysis.

This screenshot shows the Android Studio interface with the code editor open to the file `home_page.dart`. The code defines a `Widget` named `_buildPlacesList()` which returns a `FutureBuilder`. The `future` is `firestoreService.getPlaces()`, and the `builder` handles the state of the `Future`. If the connection is `waiting`, it shows a `CircularProgressIndicator`. If there's an `error`, it displays the error message. If there's no data or the data is empty, it shows a message indicating no places are available. Otherwise, it iterates over the `places` list and applies a search filter based on the `_searchController`.

```
Widget _buildPlacesList() {
    return FutureBuilder<List<Map<String, dynamic>>>(
        future: firestoreService.getPlaces(),
        builder: (context, snapshot) {
            if (snapshot.connectionState == ConnectionState.waiting) {
                return Center(
                    child: CircularProgressIndicator(),
                ); // Center
            } else if (snapshot.hasError) {
                return Center(
                    child: Text('Error: ${snapshot.error}'),
                ); // Center
            } else if (!snapshot.hasData || snapshot.data!.isEmpty) {
                return Center(
                    child: Text('No places available.'),
                ); // Center
            } else {
                List<Map<String, dynamic>> places = snapshot.data!;
                _originalPlaces = List.from(places);

                String query = _searchController.text.toLowerCase();
                if (query.isNotEmpty) {
                    places = places.where((place) =>
                        (place['name']?.toLowerCase() ?? '').contains(query) ||
                        (place['address']?.toLowerCase() ?? '').contains(query));
                }
            }
        },
    );
}
```

This screenshot continues the code from the previous one. It shows the continuation of the `_buildPlacesList()` method. After filtering the places, it creates a `ListView` to display each place as a `PlaceItem` widget. The `PlaceItem` widget is defined in the `home_page.dart` file and contains a `Text` widget for the place name.

```
Widget _buildPlacesList() {
    return FutureBuilder<List<Map<String, dynamic>>>(
        future: firestoreService.getPlaces(),
        builder: (context, snapshot) {
            if (snapshot.connectionState == ConnectionState.waiting) {
                return Center(
                    child: CircularProgressIndicator(),
                ); // Center
            } else if (snapshot.hasError) {
                return Center(
                    child: Text('Error: ${snapshot.error}'),
                ); // Center
            } else if (!snapshot.hasData || snapshot.data!.isEmpty) {
                return Center(
                    child: Text('No places available.'),
                ); // Center
            } else {
                List<Map<String, dynamic>> places = snapshot.data!;
                _originalPlaces = List.from(places);

                String query = _searchController.text.toLowerCase();
                if (query.isNotEmpty) {
                    places = places.where((place) =>
                        (place['name']?.toLowerCase() ?? '').contains(query) ||
                        (place['address']?.toLowerCase() ?? '').contains(query));
                }
            }
        },
    );
}

class PlaceItem extends StatelessWidget {
    final Map<String, dynamic> place;
    final void Function(Map<String, dynamic>) onTap;

    PlaceItem({required this.place, required this.onTap});

    @override
    Widget build(BuildContext context) {
        return ListTile(
            title: Text(place['name']),
            subtitle: Text(place['address']),
            onTap: () => onTap(place),
        );
    }
}
```

The screenshot shows the Android Studio interface with the following details:

- Project Tree:** The left sidebar displays the project structure under "Project". Key files shown include `main.dart`, `home_page.dart`, `post_screen.dart`, `welcome_screen.dart`, `main.dart` (under lib/widgets), `MakeLists.txt`, `firebase_options.dart`, `main.dart` (under test), `test`, `web`, `windows`, `flutter-plugins`, `flutter-plugins-dependencies`, `gitignore`, `metadata`, `analysis_options.yaml`, and `project02.iml`.
- Code Editor:** The main area shows the `home_page.dart` file. The code implements a search functionality using a query parameter and displays a list of places using a `ListView.builder`. It includes a `Column` with an `InkWell` widget that triggers a `Navigator.push` to a `PlaceDetailsPage`.
- Toolbars:** The bottom toolbar includes icons for Git, TODO, Profiler, Problems, Terminal, App Quality Insights, Logcat, Services, App Inspection, and Dart Analysis.
- Right Panel:** A vertical panel on the right shows build statistics: 3 warnings, 52 errors, and 5 critical errors.

The screenshot shows the Android Studio interface with the code editor open to the `home_page.dart` file. The left sidebar displays the project structure, including `lib/screens/home_page.dart`, which is currently selected. The code editor shows the following snippet:

```
    ), // ColorFilter.mode
    ), // DecorationImage
    ), // BoxDecoration
    ), // Container
    ), // InkWell
    Padding(
      padding: EdgeInsets.only(top: 10),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: [
          Text(
            place['name'] ?? '',
            style: TextStyle(
              fontSize: 20,
              fontWeight: FontWeight.w600,
            ), // TextStyle
          ), // Text
          Icon(Icons.more_vert, size: 30),
        ],
      ), // Row
    ), // Padding
    SizedBox(height: 5),
    Row(
      children: [
        Icon(

```

project02 / lib / screens / home\_page.dart

```
Project Manager | Resource Manager | Project | Commit | Pull Requests | Bookmarks | Variants | Git | TODO | Profiler | Problems | Terminal | App Quality Insights | Logcat | Services | App Inspection | Dart Analysis | Layout Inspector
```

```
265     color: Colors.amber,
266     size: 20,
267     ), // Icon
268     Text(
269       place['cityName'] ?? '',
270       style: TextStyle(
271         fontWeight: FontWeight.w500,
272       ), // TextStyle
273     ) // Text
274   ],
275   ), // Row
276   Text(
277     'Description: ${place['description'] ?? ''}',
278     style: TextStyle(
279       fontSize: 16,
280     ), // TextStyle
281   ), // Text
282   Text(
283     'Country: ${place['country'] ?? ''}',
284     style: TextStyle(
285       fontSize: 16,
286     ), // TextStyle
287   ), // Text
288   ],
289 ), // Column
290 ),
291 );
292 );
293 }
294 ),
295 );
296 }
297
298 void _filterPlaces(String query) {
299   setState(() {
300     _buildPlacesList();
301   });
302 }
303
304 class PlaceDetailsPage extends StatelessWidget {
305   @override
306   Widget build(BuildContext context) {
307     return Scaffold(
308       appBar: AppBar(
309         title: Text('Place Details'),
310       ), // AppBar
311       body: SingleChildScrollView(
312         padding: EdgeInsets.all(16),
313       ),
314     );
315   }
316 }
```

project02 / lib / screens / home\_page.dart

```
Project Manager | Resource Manager | Project | Commit | Pull Requests | Bookmarks | Variants | Git | TODO | Profiler | Problems | Terminal | App Quality Insights | Logcat | Services | App Inspection | Dart Analysis | Layout Inspector
```

```
288   ],
289   ),
290   );
291 );
292 );
293 }
294 ),
295 );
296 }
297
298 void _filterPlaces(String query) {
299   setState(() {
300     _buildPlacesList();
301   });
302 }
303
304 class PlaceDetailsPage extends StatelessWidget {
305   @override
306   Widget build(BuildContext context) {
307     return Scaffold(
308       appBar: AppBar(
309         title: Text('Place Details'),
310       ), // AppBar
311       body: SingleChildScrollView(
312         padding: EdgeInsets.all(16),
313       ),
314     );
315   }
316 }
```

project02 / lib / screens / home\_page.dart

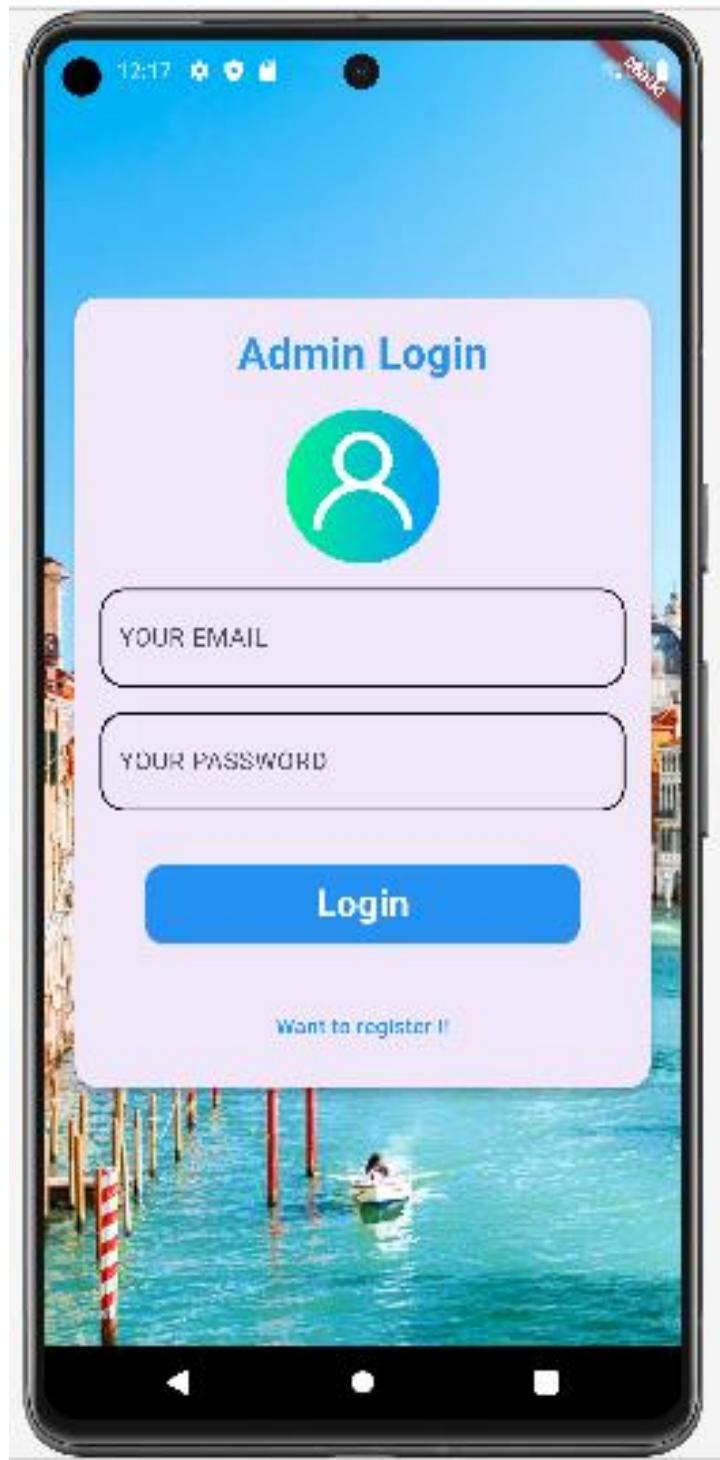
```
302 }
303 }
304 }
305 class PlaceDetailsPage extends StatelessWidget {
306   @override
307   Widget build(BuildContext context) {
308     return Scaffold(
309       appBar: AppBar(
310         title: Text('Place Details'),
311       ), // AppBar
312       body: SingleChildScrollView(
313         padding: EdgeInsets.all(16),
314         child: Column(
315           crossAxisAlignment: CrossAxisAlignment.start,
316           children: [
317             Image.asset(...), // Image.asset
318             SizedBox(height: 16),
319             Padding(
320               padding: const EdgeInsets.all(16.0),
321               child: Text(
322                 'France',
323                 style: TextStyle(fontSize: 18,fontWeight: FontWeight.bold),
324               ), // Text
325             ), // Padding
326             SizedBox(height: 8),
327             Padding(
328               padding: const EdgeInsets.all(16.0),
329               child: Text(
330                 'Le Mont-Saint-Michel',
331                 style: TextStyle(fontSize: 18,fontWeight: FontWeight.bold),
332               ), // Text
333             ), // Padding
334           ],
335         ),
336       ), // SingleChildScrollView
337     ); // Scaffold
338   }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
```

Project Manager    Resource Manager    Project    Commit    Pull Requests    Bookmarks    Variants    Git    TODO    Profiler    Problems    Terminal    App Quality Insights    Logcat    Services    App Inspection    Dart Analysis    Layout Inspector

project02 / lib / screens / home\_page.dart

```
311   ),
312   ),
313   ),
314   ),
315   ),
316   ),
317   ),
318   ),
319   ),
320   ),
321   ),
322   ),
323   ),
324   ),
325   ),
326   ),
327   ),
328   ),
329   ),
330   ),
331   ),
332   ),
333   ),
334   ),
335   ),
336   ),
337   ),
338   ),
339   ),
340   ),
341   ),
342   ),
343   ),
344   ),
345   ),
346   ),
347   ),
348   ),
349   ),
350   ),
351   ),
352   ),
353   }
```

Project Manager    Resource Manager    Project    Commit    Pull Requests    Bookmarks    Variants    Git    TODO    Profiler    Problems    Terminal    App Quality Insights    Logcat    Services    App Inspection    Dart Analysis    Layout Inspector



The screenshot shows the Android Studio interface with the project navigation bar at the top. The left sidebar shows the project structure under 'lib/auth'. The main editor window displays the code for 'adminlogin.dart'. The code defines a stateful widget 'AdminLogin' and its state class '\_AdminLoginState'. It includes imports for flutter/material.dart, firebase\_auth/firebase\_auth.dart, and project02/dashboard/admin/dashboard.dart. The build method creates a Scaffold with a Container containing a Card. The Card has a BoxDecoration with a BoxFit.cover image from assets/images/bg.jpg. The text 'Admin Login' is displayed in a large, bold, black font.

```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:project02/dashboard/admin/dashboard.dart';
import 'package:project02/screens/home_page.dart'; // Import your HomePage widget
import 'package:flutter/material.dart';

class AdminLogin extends StatefulWidget {
  @override
  _AdminLoginState createState() => _AdminLoginState();
}

class _AdminLoginState extends State<AdminLogin> {
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();

  final String adminEmail = 'admin123@gmail.com';
  final String adminPassword = 'admin123';

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        decoration: BoxDecoration(

```

This screenshot continues the code from the previous one. It shows the continuation of the 'build' method, which adds a Column with a MainAxisSize.center alignment and children. One child is a Text widget with the text 'Admin Login' and a TextStyle of 'fontStyle: FontStyle.italic, fontSize: 28'. The other child is a SingleChildScrollView containing a Column with a MainAxisAlignment center alignment and children. The first child is a Text widget with the text 'Admin Login' and a TextStyle of 'fontStyle: FontStyle.italic, fontSize: 28'. The second child is a Text widget with the text 'Admin Login' and a TextStyle of 'fontStyle: FontStyle.italic, fontSize: 28'.

```
color: Color.fromRGBO(173, 216, 230, 0.7),
image: DecorationImage(
  image: AssetImage('assets/images/bg.jpg'),
  fit: BoxFit.cover,
), // DecorationImage
), // BoxDecoration
child: Padding(
  padding: const EdgeInsets.all(16.0),
)
child: Center(
  child: Card(
    elevation: 5,
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(15.0),
    ), // RoundedRectangleBorder
    child: Padding(
      padding: const EdgeInsets.all(16.0),
    )
    child: SingleChildScrollView(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Text(
            'Admin Login',
            style: TextStyle(
              fontStyle: FontStyle.italic,
              fontSize: 28,

```

project02 / lib / auth / adminlogin.dart

Resource Manager Project

adminlogin.dart

```
Column
  |
  |   fontWeight: FontWeight.bold,
  |   color: Colors.blue
  | ), // TextStyle
  | ), // Text
  | SizedBox(height: 16),
  | Form(
  |   key: _formKey,
  |   child: Column(
  |     mainAxisAlignment: MainAxisAlignment.center,
  |     children: [
  |       Image.asset(
  |         'assets/images/login.png',
  |         height: 100,
  |         width: 100,
  |       ), // Image.asset
  |       SizedBox(height: 16),
  |       TextFormField(
  |         controller: _emailController,
  |         decoration: InputDecoration(
  |           labelText: 'YOUR EMAIL',
  |           focusedBorder: OutlineInputBorder(
  |             borderRadius: BorderRadius.circular(16),
  |             borderSide: const BorderSide(color: Colors.black, width: 2.5),
  |           ), // OutlineInputBorder
  |         ),
  |       ), // TextFormField
  |     ],
  |   ),
  | 
```

Git TODO Profiler Problems Terminal Logcat Services App Quality Insights App Inspection Run Dart Analysis Layout Inspector

project02 / lib / auth / adminlogin.dart

Resource Manager Project

adminlogin.dart

```
Column
  |
  |   enabledBorder: OutlineInputBorder(
  |     borderRadius: BorderRadius.circular(16),
  |     borderSide: const BorderSide(color: Colors.black, width: 1.5),
  |   ), // OutlineInputBorder
  |   ), // InputDecoration
  |   validator: (value) {
  |     if (value!.isEmpty) {
  |       return 'Please enter your email';
  |     }
  |     return null;
  |   },
  |   ), // TextFormField
  |   SizedBox(height: 16),
  |   TextFormField(
  |     controller: _passwordController,
  |     decoration: InputDecoration(
  |       labelText: 'YOUR PASSWORD',
  |       focusedBorder: OutlineInputBorder(
  |         borderRadius: BorderRadius.circular(16),
  |         borderSide: const BorderSide(color: Colors.black, width: 2.5),
  |       ), // OutlineInputBorder
  |     ),
  |   ), // TextFormField
  | 
```

Git TODO Profiler Problems Terminal Logcat Services App Quality Insights App Inspection Run Dart Analysis Layout Inspector

The screenshot shows the Android Studio interface with the code editor open to the file `adminlogin.dart`. The code is a Dart script for an admin login screen. It includes imports for `OutlineInputBorder`, `InputDecoration`, `validator`, `TextFormField`, `SizedBox`, `Padding`, `ElevatedButton`, `EdgeInsets.all`, `Navigator.pushReplacement`, and `MaterialPageRoute`. The script handles user input validation and navigation to a home page if credentials are valid.

```
    ), // OutlineInputBorder
    ), // InputDecoration
    obscureText: true,
    validator: (value) {
        if (value!.isEmpty) {
            return 'Please enter your password';
        }
        return null;
    },
),
// TextFormField
SizedBox(height: 25),
Padding(
padding: const EdgeInsets.all(10.0),
child: ElevatedButton(
 onPressed: () async {
if (_formKey.currentState!.validate()) {
if (_emailController.text == adminEmail &&
_passwordController.text == adminPassword) {
// Admin login successful
Navigator.pushReplacement(
context,
MaterialPageRoute(builder: (context) => HomePage()), // Navigate to HomePage
);
} else {
// Invalid admin credentials
showDialog(
context: context,
builder: (BuildContext context) {
return AlertDialog(
title: Text('Error'),
content: Text('Invalid email or password.'),
actions: [
TextButton(
 onPressed: () => Navigator.of(context).pop(),
child: Text('OK'),
),
// TextButton
],
); // AlertDialog
});
}
}
},
style: ElevatedButton.styleFrom(
primary: Colors.blue,
```

This screenshot is identical to the one above, showing the same code editor for `adminlogin.dart`. The code remains the same, handling user input validation and navigation logic.

```
    ), // OutlineInputBorder
    ), // InputDecoration
    obscureText: true,
    validator: (value) {
        if (value!.isEmpty) {
            return 'Please enter your password';
        }
        return null;
    },
),
// TextFormField
SizedBox(height: 25),
Padding(
padding: const EdgeInsets.all(10.0),
child: ElevatedButton(
 onPressed: () async {
if (_formKey.currentState!.validate()) {
if (_emailController.text == adminEmail &&
_passwordController.text == adminPassword) {
// Admin login successful
Navigator.pushReplacement(
context,
MaterialPageRoute(builder: (context) => HomePage()), // Navigate to HomePage
);
} else {
// Invalid admin credentials
showDialog(
context: context,
builder: (BuildContext context) {
return AlertDialog(
title: Text('Error'),
content: Text('Invalid email or password.'),
actions: [
TextButton(
 onPressed: () => Navigator.of(context).pop(),
child: Text('OK'),
),
// TextButton
],
); // AlertDialog
});
}
}
},
style: ElevatedButton.styleFrom(
primary: Colors.blue,
```

project02 / lib / auth / adminlogin.dart

```
Project Manager Resource Manager Project Pull Requests Commit Bookmarks Variants
```

adminlogin.dart

```
Column
style: ElevatedButton.styleFrom(
    primary: Colors.blue,
    fixedSize: Size(280, 50),
    shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.all(Radius.circular(10)),
    ), // RoundedRectangleBorder
),
child: Text(
    'Login',
    style: TextStyle(color: Colors.white, fontWeight: FontWeight.bold, fontSize: 24),
), // Text
), // ElevatedButton
), // Padding
SizedBox(height: 20),
TextButton(
onPressed: () {
    Navigator.push(context, MaterialPageRoute(builder: (context) => RegisterForm()));
},
child: Text(
    'Want to register !!!',
    style: TextStyle(color: Colors.blue),
),
), // Text
), // TextButton
],
```

Git TODO Profiler Problems Terminal Logcat Services App Quality Insights App Inspection Run Dart Analysis Layout Inspector

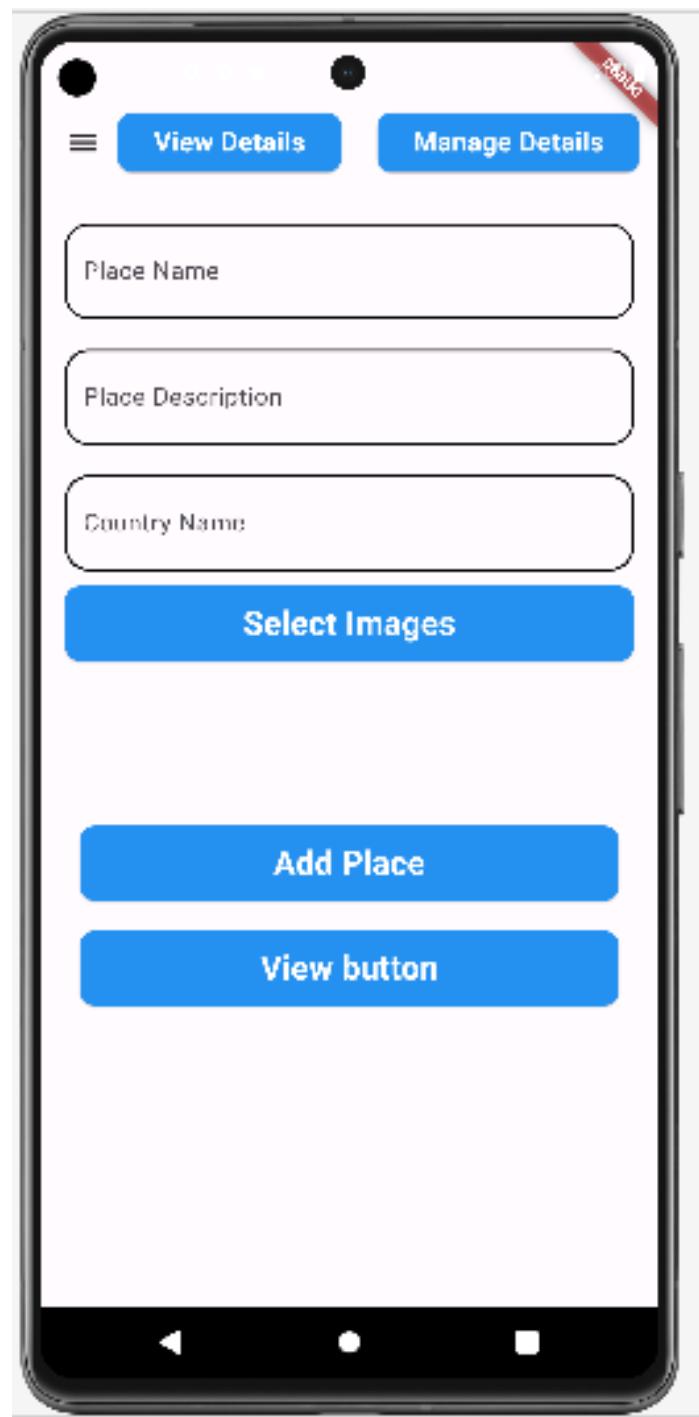
project02 / lib / auth / adminlogin.dart

```
Project Manager Resource Manager Project Pull Requests Commit Bookmarks Variants
```

adminlogin.dart

```
), // Padding
SizedBox(height: 20),
TextButton(
onPressed: () {
    Navigator.push(context, MaterialPageRoute(builder: (context) => RegisterForm()));
},
child: Text(
    'Want to register !!!',
    style: TextStyle(color: Colors.blue),
),
), // Text
), // TextButton
], // Column
), // Form
], // Column
), // SingleChildScrollView
), // Padding
), // Card
), // Center
), // Padding
), // Container
); // Scaffold
}
```

Git TODO Profiler Problems Terminal Logcat Services App Quality Insights App Inspection Run Dart Analysis Layout Inspector



The screenshot shows the Android Studio interface with the code editor open to the file `dashboard.dart`. The code implements a `StatefulWidget` named `AdminDashboardPage` which handles image picking and state management.

```
import 'package:flutter/material.dart';
import 'package:project02/dashboard/admin/firestore_service.dart';
import 'package:project02/screens/home_page.dart';

import 'dart:io';
import 'package:image_picker/image_picker.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:cloud_firestore/cloud_firestore.dart';

import 'package:flutter/material.dart';

class AdminDashboardPage extends StatefulWidget {
  @override
  _AdminDashboardPageState createState() => _AdminDashboardPageState();
}

class _AdminDashboardPageState extends State<AdminDashboardPage> {
  final TextEditingController nameController = TextEditingController();
  final TextEditingController descriptionController = TextEditingController();
  final TextEditingController countryController = TextEditingController();
  List<File> selectedImages = [];
  GlobalKey<ScaffoldState> _drawerKey = GlobalKey();

  Future<void> _pickImage() async {
    final ImagePicker _picker = ImagePicker();
    final XFile? image = await _picker.pickImage(source: ImageSource.gallery);

    if (image != null) {
      setState(() {
        selectedImages.add(File(image.path));
      });
    }
  }

  void _onDrawerItemTap(String itemName) {
    // Handle navigation based on drawer item selection
    switch (itemName) {
      case 'View Details':
        break;
      case 'Manage Details':
        break;
    }

    Navigator.pop(context);
  }
}
```

The screenshot shows the continuation of the `dashboard.dart` code from the previous screenshot. It includes the `_pickImage` method and the `_onDrawerItemTap` method, which handles navigation based on drawer item selection.

```
GlobalKey<ScaffoldState> _drawerKey = GlobalKey();

Future<void> _pickImage() async {
  final ImagePicker _picker = ImagePicker();
  final XFile? image = await _picker.pickImage(source: ImageSource.gallery);

  if (image != null) {
    setState(() {
      selectedImages.add(File(image.path));
    });
  }
}

void _onDrawerItemTap(String itemName) {
  // Handle navigation based on drawer item selection
  switch (itemName) {
    case 'View Details':
      break;
    case 'Manage Details':
      break;
  }

  Navigator.pop(context);
}
```

project02 / lib / dashboard / admin > dashboard.dart

```
Project Manager Resource Manager Project Commit Pull Requests Bookmarks Layout Inspector
project02 C:\Users\DELL\Andro
> dart_tool
> idea
> android [project02_android]
> assets
> build
> firebaseWork
> flt
> ios
lib
> auth
> dashboard
> admin
> screens
> widgets
> main.dart
> linux
> macos
> test

home_page.dart main.dart manageddetails.dart dashboard.dart

Navigator.pop(context);

Future<void> _uploadImagesAndAddPlace() async {
List<String> imageUrl = await uploadImagesAndGetUrls(selectedImages);

Map<String, dynamic> placeData = {
  'name': nameController.text,
  'description': descriptionController.text,
  'country': countryController.text,
  'images': imageUrl,
};

await FirestoreService().addPlace(placeData);

Navigator.push(context, MaterialPageRoute(builder: (context) => HomePage()));

Future<List<String>> uploadImagesAndGetUrls(List<File> images) async {
List<String> imageUrl = [];
for (File image in images) {
  String imageName = DateTime.now().millisecondsSinceEpoch.toString();
  Reference storageReference = FirebaseStorage.instance.ref().child('images/$imageName.jpg');

  UploadTask uploadTask = storageReference.putFile(image);
  TaskSnapshot taskSnapshot = await uploadTask.whenComplete(() => null);

  String downloadURL = await taskSnapshot.ref.getDownloadURL();
  imageUrl.add(downloadURL);
}

return imageUrl;
}

override
Widget build(BuildContext context) {
  return Scaffold(
    key: _drawerKey,
    appBar: AppBar(
      title: Text('Admin Dashboard'),
      actions: [
        Padding(
          padding: const EdgeInsets.symmetric(horizontal: 12.0),
          child: ElevatedButton(
            onPressed: () {
              Navigator.pop(context);
            },
            child: Text('Logout'),
          ),
        ),
      ],
    ),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [
          Text('Welcome to Admin Dashboard!'),
          Text('Total Places: ${placeData.length}'),
          Text('Total Images: ${imageUrl.length}'),
        ],
      ),
    ),
  );
}
```

Project Manager Resource Manager Project Commit Pull Requests Bookmarks Layout Inspector
project02 C:\Users\DELL\Andro
> dart\_tool
> idea
> android [project02\_android]
> assets
> build
> firebaseWork
> flt
> ios
lib
> auth
> dashboard
> admin
> screens
> widgets
> main.dart
> linux
> macos
> test

home\_page.dart main.dart manageddetails.dart dashboard.dart

for (File image in images) {
 String imageName = DateTime.now().millisecondsSinceEpoch.toString();
 Reference storageReference = FirebaseStorage.instance.ref().child('images/\$imageName.jpg');

 UploadTask uploadTask = storageReference.putFile(image);
 TaskSnapshot taskSnapshot = await uploadTask.whenComplete(() => null);

 String downloadURL = await taskSnapshot.ref.getDownloadURL();
 imageUrl.add(downloadURL);
}

return imageUrl;
}

override
Widget build(BuildContext context) {
 return Scaffold(
 key: \_drawerKey,
 appBar: AppBar(
 title: Text('Admin Dashboard'),
 actions: [
 Padding(
 padding: const EdgeInsets.symmetric(horizontal: 12.0),
 child: ElevatedButton(
 onPressed: () {
 Navigator.pop(context);
 },
 child: Text('Logout'),
 ),
 ),
 ],
 ),
 body: Center(
 child: Column(
 mainAxisAlignment: MainAxisAlignment.spaceEvenly,
 children: [
 Text('Welcome to Admin Dashboard!'),
 Text('Total Places: \${placeData.length}'),
 Text('Total Images: \${imageUrl.length}'),
 ],
 ),
 ),
 );
}

The screenshot displays two instances of the Android Studio code editor, showing the same Dart file (`dashboard.dart`) at different stages of development.

**Top Editor Content:**

```
override
Widget build(BuildContext context) {
  return Scaffold(
    key: _drawerKey,
    appBar: AppBar(
      title: Text('Admin Dashboard'),
      actions: [
        Padding(
          padding: const EdgeInsets.symmetric(horizontal: 12.0),
          child: ElevatedButton(
            onPressed: () {
            },
            style: ElevatedButton.styleFrom(
              primary: Colors.blue,
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(10),
              ),
            ),
            child: Text('View Details', style: TextStyle(color: Colors.white, fontWeight: FontWeight.bold, fontSize: 18)),
          ),
        ),
      ],
    ),
    padding: const EdgeInsets.symmetric(horizontal: 12.0),
    child: ElevatedButton(
      onPressed: () {
      },
      style: ElevatedButton.styleFrom(
        primary: Colors.blue,
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(10),
        ),
      ),
      child: Text('Manage Details', style: TextStyle(color: Colors.white, fontWeight: FontWeight.bold, fontSize: 18)),
    ),
  );
}
```

**Bottom Editor Content:**

```
Padding(
  padding: const EdgeInsets.symmetric(horizontal: 12.0),
  child: ElevatedButton(
    onPressed: () {
      Navigator.push(
        context, MaterialPageRoute(builder: (context) => AdminDashboardPage()));
    },
    style: ElevatedButton.styleFrom(
      primary: Colors.blue,
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(10),
      ),
    ),
    child: Text('Manage Details', style: TextStyle(color: Colors.white, fontWeight: FontWeight.bold, fontSize: 18)),
  ),
), // Padding
], // AppBar
drawer: Drawer(
  child: ListView(
    padding: EdgeInsets.zero,
    children: [
      DrawerHeader(
        decoration: BoxDecoration(

```

The bottom editor shows the addition of a `Navigator.push` call within the `onPressed` handler of the first Elevated Button, which navigates to the `AdminDashboardPage`.

project02 / lib / dashboard / admin / dashboard.dart

```
Project Manager Resource Manager Project Commit Pull Requests Bookmarks App Quality Insights Logcat Services App Inspection Run Dart Analysis
```

```
padding: EdgeInsets.zero,
children: [
  DrawerHeader(
    decoration: BoxDecoration(
      color: Colors.blue,
    ), // BoxDecoration
    child: Text(
      'Drawer Header',
      style: TextStyle(
        color: Colors.white,
        fontSize: 24,
      ), // TextStyle
    ), // Text
  ), // DrawerHeader
  Padding(
    padding: const EdgeInsets.only(top: 10.0),
    child: ListTile(
      title: Text('Add Details'),
      onTap: () {
        Navigator.pop(context); // Close the drawer
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => AdminDashboardPage()),
        );
      },
    ),
  ),
]
```

project02 / lib / dashboard / admin / dashboard.dart

```
Project Manager Resource Manager Project Commit Pull Requests Bookmarks App Quality Insights Logcat Services App Inspection Run Dart Analysis
```

```
, // ListTile
), // Padding
Padding(
  padding: const EdgeInsets.only(top: 20.0),
  child: ListTile(
    title: Text('Manage Details'),
    onTap: () {
      Navigator.pop(context); // Close the drawer
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => AdminDashboardPage()),
      );
    },
  ),
), // ListTile
), // Padding
], // ListView
), // Drawer
body: Padding(
  padding: const EdgeInsets.all(16.0),
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.stretch,
    children: [
      Padding(
        padding: const EdgeInsets.only(bottom: 10.0),

```

project02 / lib / dashboard / admin / dashboard.dart

```
controller: nameController,
decoration: InputDecoration(
  labelText: 'Place Name',
  focusedBorder: OutlineInputBorder(
    borderRadius: BorderRadius.circular(16),
    borderSide: const BorderSide(color: Colors.black, width: 2.5),
  ), // OutlineInputBorder
  enabledBorder: OutlineInputBorder(
    borderRadius: BorderRadius.circular(16),
    borderSide: const BorderSide(color: Colors.black, width: 1.5),
  ), // OutlineInputBorder
), // InputDecoration
), // Padding
Padding(
  padding: const EdgeInsets.symmetric(vertical: 10.0),
  child: TextField(
    controller: descriptionController,
    decoration: InputDecoration(
      labelText: 'Place Description',
      focusedBorder: OutlineInputBorder(
        borderRadius: BorderRadius.circular(16),
        borderSide: const BorderSide(color: Colors.black, width: 2.5),
      ), // OutlineInputBorder
      enabledBorder: OutlineInputBorder(
        borderRadius: BorderRadius.circular(16),
        borderSide: const BorderSide(color: Colors.black, width: 1.5),
      ), // OutlineInputBorder
    ),
  ),
), // Padding
```

Project Manager    Resource Manager    Commit    Pull Requests    Bookmarks    Variants    Git    TODO    Profiler    Problems    Terminal    App Quality Insights    Logcat    Services    App Inspection    Run    Dart Analysis    Layout Inspector

project02 / lib / dashboard / admin / dashboard.dart

```
), // OutlineInputBorder
), // InputDecoration
), // Padding
Padding(
  padding: const EdgeInsets.symmetric(vertical: 10.0),
  child: TextField(
    controller: countryController,
    decoration: InputDecoration(
      labelText: 'Country Name',
      focusedBorder: OutlineInputBorder(
        borderRadius: BorderRadius.circular(16),
        borderSide: const BorderSide(color: Colors.black, width: 2.5),
      ), // OutlineInputBorder
      enabledBorder: OutlineInputBorder(
        borderRadius: BorderRadius.circular(16),
        borderSide: const BorderSide(color: Colors.black, width: 1.5),
      ), // OutlineInputBorder
    ),
  ),
), // Padding
ElevatedButton(
  onPressed: _pickImage,
  style: ElevatedButton.styleFrom(
    primary: Colors.blue,
  ),
), // ElevatedButton
```

Project Manager    Resource Manager    Commit    Pull Requests    Bookmarks    Variants    Git    TODO    Profiler    Problems    Terminal    App Quality Insights    Logcat    Services    App Inspection    Run    Dart Analysis    Layout Inspector

The screenshot shows the Android Studio interface with the following details:

- Project Bar:** Shows the project structure: `project02` / `lib` / `dashboard` / `admin` / `dashboard.dart`.
- Resource Manager:** Shows the file tree:
  - `project02`:
    - `.dart_tool`
    - `idea`
    - `android [project02_android]`
    - `assets`
    - `build`
    - `firebaseWork`
    - `fl`
    - `ios`
  - `lib`:
    - `auth`
    - `dashboard`:
      - `admin`:
        - `dashboard.dart`
        - `data.dart`
        - `firestore_service.dart`
        - `manageddetails.dart`
      - `screens`:
        - `home_page.dart`
        - `post_screen.dart`
        - `welcome_screen.dart`
    - `widgets`
    - `CMakeLists.txt`
    - `firebase_options.dart`
    - `main.dart`
  - `linux`
  - `macos`
  - `test`
- Code Editor:** The `dashboard.dart` file is open, showing Dart code for a UI component. The code includes a `ListView.builder` with `itemCount` and `itemBuilder` callbacks, and an `ElevatedButton` with various properties like `onPressed`, `style`, and `shape`.
- Toolbars:** Includes Git, TODO, Profiler, Problems, Terminal, App Quality Insights, Logcat, Services, App Inspection, Run, and Dart Analysis.
- Bottom Right:** Layout Inspector icon.

The screenshot shows the Android Studio interface with the following details:

- Project Bar:** Shows the project structure: `project02` / `lib` / `dashboard` / `admin` / `dashboard.dart`.
- Resource Manager:** Shows files in the `admin` directory: `dashboard.dart`, `data.dart`, `firebase_service.dart`, and `manageddetails.dart`.
- Code Editor:** The `dashboard.dart` file is open, showing Dart code for a UI component. The code includes imports for `Padding`, `ElevatedButton`, and `Navigator`. It defines a button with a rounded rectangle border, white text, and a bold font weight.
- Bottom Navigation Bar:** Includes icons for Git, TODO, Profiler, Problems, Terminal, App Quality Insights, Logcat, Services, App Inspection, Run, and Dart Analysis.



project02 lib dashboard admin manageddetails.dart

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:project02/dashboard/admin/dashboard.dart';
import 'package:project02/widgets/home_bottom_bar.dart';

class Managedetails extends StatefulWidget {
    final String? userProfileImageUrl;

    const Managedetails({Key? key, this.userProfileImageUrl}) : super(key: key);

    @override
    ManagedetailsState createState() => _ManagedetailsState();
}

class _ManagedetailsState extends State<Managedetails> {
    final TextEditingController searchController = TextEditingController();
    final FirestoreService firestoreService = FirestoreService();
    late List<Map<String, dynamic>> _originalPlaces = [];
    late List<Map<String, dynamic>> _places = [];

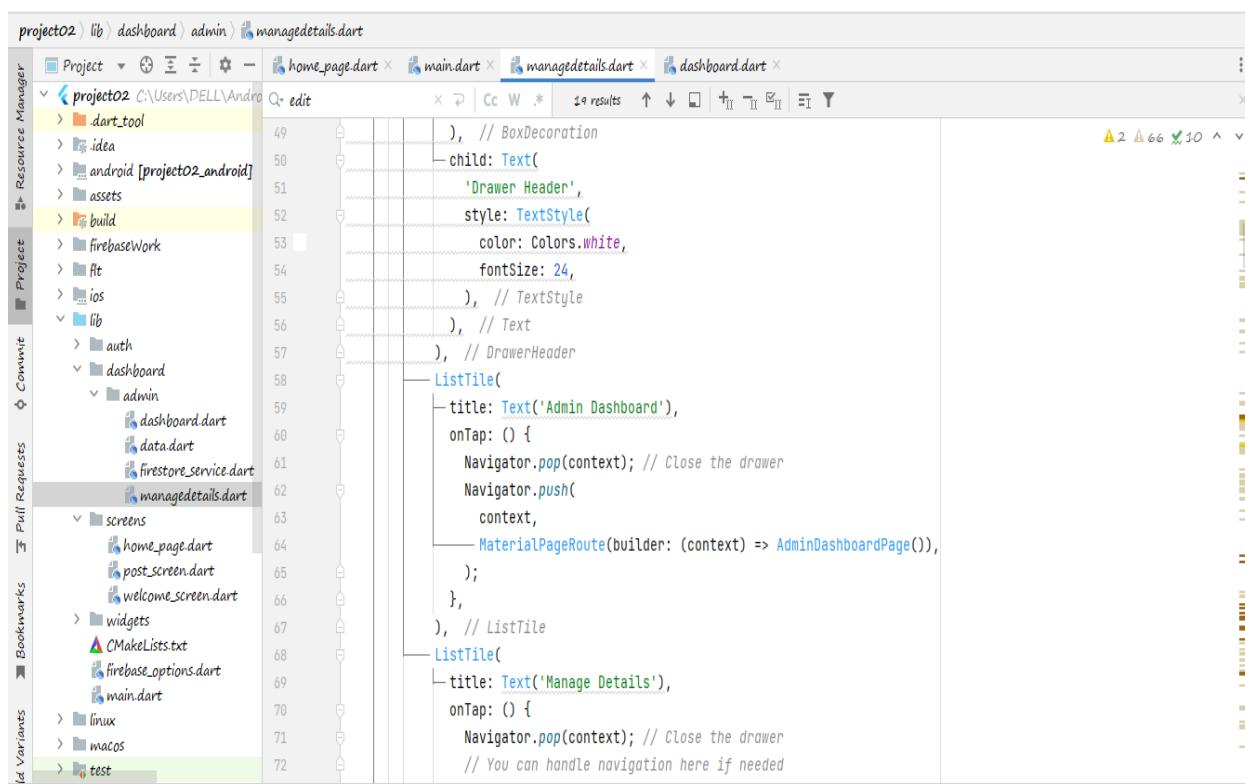
    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                title: Text('Home Page'),
                actions: [
                    if (widget.userProfileImageUrl != null)
                        CircleAvatar(
                            backgroundImage: NetworkImage(widget.userProfileImageUrl!),
                        ),
                    IconButton(
                        icon: Icon(Icons.manage_accounts, size: 40, color: Colors.blueAccent),
                        onPressed: () {
                            Navigator.push(
                                context,
                                MaterialPageRoute(builder: (context) => PostScreen()),
                            );
                        },
                    ),
                ],
            ),
            drawer: Drawer(
                child: ListView(
                    children: [

```

project02 lib dashboard admin manageddetails.dart

```
22 @override
23 Widget build(BuildContext context) {
24     return Scaffold(
25         appBar: AppBar(
26             title: Text('Home Page'),
27             actions: [
28                 if (widget.userProfileImageUrl != null)
29                     CircleAvatar(
30                         backgroundImage: NetworkImage(widget.userProfileImageUrl!),
31                     ),
32                     IconButton(
33                         icon: Icon(Icons.manage_accounts, size: 40, color: Colors.blueAccent),
34                         onPressed: () {
35                             Navigator.push(
36                                 context,
37                                 MaterialPageRoute(builder: (context) => PostScreen()),
38                             );
39                         },
40                     ),
41                 ],
42             ),
43         ),
44         drawer: Drawer(
45             child: ListView(

```



The screenshot shows the Android Studio interface with the project navigation bar at the top. The left sidebar displays the project structure under 'Project'. The main code editor area shows the 'manageddetails.dart' file, which is currently selected in the tabs at the top. The code itself is a Dart file containing a list of ListTile widgets. The bottom navigation bar includes icons for Git, TODO, Profiler, Problems, Terminal, App Quality Insights, Logcat, Services, App Inspection, Run, and Dart Analysis.

```
project02 / lib / dashboard / admin / manageddetails.dart
Project  home_page.dart  main.dart  manageddetails.dart  dashboard.dart
Resource Manager
  project02 C:\Users\DELLY\AndroidStudioProjects\project02
    > .dart_tool
    > idea
    > android [project02_android]
    > assets
    > build
    > FirebaseWorkshop
    > flt
    > ios
    > lib
      > auth
      > dashboard
        > admin
          > dashboard.dart
          > data.dart
          > firestore_service.dart
          > manageddetails.dart
        > Screens
          > home_page.dart
          > post_screen.dart
          > welcome_screen.dart
      > widgets
        > CMakeLists.txt
        > firebase_options.dart
        > main.dart
    > linux
    > macos
    > test

Project  home_page.dart  main.dart  manageddetails.dart  dashboard.dart
Resource Manager
  project02 C:\Users\DELLY\AndroidStudioProjects\project02
    > .dart_tool
    > idea
    > android [project02_android]
    > assets
    > build
    > FirebaseWorkshop
    > flt
    > ios
    > lib
      > auth
      > dashboard
        > admin
          > dashboard.dart
          > data.dart
          > firestore_service.dart
          > manageddetails.dart
        > Screens
          > home_page.dart
          > post_screen.dart
          > welcome_screen.dart
      > widgets
        > CMakeLists.txt
        > firebase_options.dart
        > main.dart
    > linux
    > macos
    > test
```

project02 / lib / dashboard / admin) manageddetails.dart

```
    ), // Padding
    _buildPlacesList(),
  ],
),
// Column
),
// SingleChildScrollView
),
// Padding
),
// SafeArea
bottomNavigationBar: HomeBottomBar(),
); // Scaffold
}

Widget _buildPlacesList() {
  return FutureBuilder<List<Map<String, dynamic>>>(
    future: firestoreService.getPlaces(),
    builder: (context, snapshot) {
      if (snapshot.connectionState == ConnectionState.waiting) {
        return Center(
          child: CircularProgressIndicator(),
        ); // Center
      } else if (snapshot.hasError) {
        return Center(
          child: Text('Error: ${snapshot.error}'),
        ); // Center
      } else if (!snapshot.hasData || snapshot.data!.isEmpty) {
        return Center(
          child: Text('No places found'),
        );
      } else {
        List<Map<String, dynamic>> places = _places.isNotEmpty ? _places : snapshot.data!;
        _originalPlaces = List.from(places);
        return ListView.builder(
          shrinkWrap: true,
          physics: NeverScrollableScrollPhysics(),
          itemCount: places.length,
          itemBuilder: (BuildContext context, int index) {
            Map<String, dynamic> place = places[index];
            return Padding(
              padding: EdgeInsets.all(15),
              child: Column(
                children: [
                  InkWell(
                    onTap: () {
                      Navigator.push(
                        context,
                        MaterialPageRoute(
                          builder: (context) => PlaceDetailsPage(placeIndex: index),
                        ),
                      );
                    },
                  );
                ],
              ),
            );
          },
        );
      }
    }
  );
}
```

Project Manager    Resource Manager    Project    Commit    Full Requests    Bookmarks    TODO    Profiler    Problems    Terminal    App Quality Insights    Logcat    Services    App Inspection    Run    Dart Analysis    Layout Inspector

project02 / lib / dashboard / admin) manageddetails.dart

```
    ),
  );
}

Widget _buildPlacesList() {
  return FutureBuilder<List<Map<String, dynamic>>>(
    future: firestoreService.getPlaces(),
    builder: (context, snapshot) {
      if (snapshot.connectionState == ConnectionState.waiting) {
        return Center(
          child: CircularProgressIndicator(),
        ); // Center
      } else if (snapshot.hasError) {
        return Center(
          child: Text('Error: ${snapshot.error}'),
        ); // Center
      } else if (!snapshot.hasData || snapshot.data!.isEmpty) {
        return Center(
          child: Text('No places found'),
        );
      } else {
        List<Map<String, dynamic>> places = _places.isNotEmpty ? _places : snapshot.data!;
        _originalPlaces = List.from(places);
        return ListView.builder(
          shrinkWrap: true,
          physics: NeverScrollableScrollPhysics(),
          itemCount: places.length,
          itemBuilder: (BuildContext context, int index) {
            Map<String, dynamic> place = places[index];
            return Padding(
              padding: EdgeInsets.all(15),
              child: Column(
                children: [
                  InkWell(
                    onTap: () {
                      Navigator.push(
                        context,
                        MaterialPageRoute(
                          builder: (context) => PlaceDetailsPage(placeIndex: index),
                        ),
                      );
                    },
                  );
                ],
              ),
            );
          },
        );
      }
    }
  );
}
```

Project Manager    Resource Manager    Project    Commit    Full Requests    Bookmarks    TODO    Profiler    Problems    Terminal    App Quality Insights    Logcat    Services    App Inspection    Run    Dart Analysis    Layout Inspector

The screenshot shows the Android Studio interface with the following details:

- Project Tree:** The left sidebar shows the project structure under "project02". Key components include "lib" (containing "admin" and "dashboard" folders), "main.dart", "home\_page.dart", and "manageddetails.dart".
- Code Editor:** The main window displays the content of "manageddetails.dart". The code uses Dart's null-safety features and includes imports like "Container", "BoxDecoration", "Image", "ColorFilter", "BlendMode", "InkWell", "Padding", and "Row".

```
child: Container(
    height: 200,
    decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(15),
        image: DecorationImage(
            image: NetworkImage(place['images'][0]),
            fit: BoxFit.cover,
            colorFilter: ColorFilter.mode(
                Colors.black.withOpacity(0.8),
                BlendMode.srcOver,
            ), // ColorFilter.mode
        ), // DecorationImage
    ), // BoxDecoration
), // Container
), // InkWell
Padding(
    padding: EdgeInsets.only(top: 10),
    child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: [
            Text(
                place['name'] ?? '',
                style: TextStyle(
                    fontSize: 20,
```
- Search Bar:** At the top, there is a search bar with the text "manageddetails.dart" and a count of "19 results".
- Bottom Navigation:** The bottom navigation bar includes icons for Git, TODO, Profiler, Problems, Terminal, App Quality Insights, Logcat, Services, App Inspection, Run, and Dart Analysis.

The screenshot shows the Android Studio interface with the following details:

- Project Bar:** Shows the project structure: `project02` > `lib` > `dashboard` > `admin` > `manageddetails.dart`.
- Resource Manager:** A tree view of files and assets under `project02`, including `build`, `assets`, `build`, `firebaseWork`, `fl`, `ios`, and `lib` (containing `auth`, `dashboard`, `admin`, `widgets`, `linux`, `macos`, and `test`).
- Code Editor:** The `manageddetails.dart` file is open. The code defines a `Row` widget with a child `ElevatedButton` that has a `fontSize: 16` and a `color: Colors.white`. It also includes `borderRadius: BorderRadius.only` and `shape: RoundedRectangleBorder` properties.
- Bottom Navigation:** Includes tabs for `Git`, `TODO`, `Profiler`, `Problems`, `Terminal`, `App Quality Insights`, `Logcat`, `Services`, `App Inspection`, `Run`, `Dart Analysis`, and `Layout Inspector`.

The screenshot shows the Android Studio interface with the following details:

- Project Tree:** The project structure is visible on the left, showing the file `manageddetails.dart` under the `admin` folder in the `dashboard` package.
- Code Editor:** The main window displays the `manageddetails.dart` file. The code uses the `ElevatedButton` widget from the `Material` library to create a button with a blue background, rounded corners, and a large font size. It also includes a `Row` and `Icon` widget to display a star icon with an amber color and a size of 20.
- Toolbars and Status Bar:** Standard Android Studio toolbars and status bar elements are present at the bottom.

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The left sidebar shows the project structure under "project02". The "lib/dashboard/admin" directory contains the "manageddetails.dart" file, which is currently selected.
- Code Editor:** The main window displays the Dart code for "manageddetails.dart". The code includes imports for "Icon" and "Text", and defines a row of icons followed by a row of text and descriptions.
- Toolbars and Status Bar:** The top bar includes tabs for "home\_page.dart", "main.dart", "manageddetails.dart" (which is the active tab), and "dashboard.dart". The bottom bar includes icons for Git, TODO, Profiler, Problems, Terminal, App Quality Insights, Logcat, Services, App Inspection, Run, and Dart Analysis.

project02 / lib / dashboard / admin) manageddetails.dart

```
247 ), // Text
248 Text(
249   'Country: ${place['country']} ?? ''},
250   style: TextStyle(
251     fontSize: 16,
252   ), // TextStyle
253   ), // Text
254   ],
255   ), // Column
256   ); // Padding
257 },
258 ); // ListView.builder
259 }
260 },
261 ); // FutureBuilder
262 }
263
264 void _filterPlaces(String query) {
265 setState(() {
266   // Filter places based on the search query
267   _places = _originalPlaces.where((place) =>
268     (place['name']?.toLowerCase() ?? '').contains(query.toLowerCase()) ||
269     (place['cityName']?.toLowerCase() ?? '').contains(query.toLowerCase()) ||
270     (place['country']?.toLowerCase() ?? '').contains(query.toLowerCase())
271 });
272 }
```

Project Manager

- project02 C:\Users\DELL\AndroidStudioProjects\project02
- > dart\_tool
- > idea
- > android [project02\_android]
- > assets
- > build
- > firebaseWork
- > flt
- > ios
- < lib
- > auth
- < dashboard
- < admin
- < screens
- < widgets
- CMakeLists.txt
- firebase\_options.dart
- main.dart
- > linux
- > macos
- > test

Id Variants Bookmarks Pull Requests Commit

Layout Inspector

project02 / lib / dashboard / admin) manageddetails.dart

```
274
275 void _editPlace(Map<String, dynamic> place) {
276   Navigator.push(
277     context,
278     MaterialPageRoute(
279       builder: (context) => EditPlaceScreen(place: place),
280     ), // MaterialPageRoute
281   );
282 }
283
284 void _deletePlace(Map<String, dynamic> place) {
285   firestoreService.deletePlace(place['id']); // Call FirestoreService to delete place
286 }
287
288 class FirestoreService {
289   final FirebaseFirestore _firestore = FirebaseFirestore.instance;
290
291   Future<List<Map<String, dynamic>>> getPlaces() async {
292     QuerySnapshot querySnapshot = await _firestore.collection('places').get();
293     return querySnapshot.docs.map((doc) => doc.data() as Map<String, dynamic>).toList();
294   }
295
296   Future<void> deletePlace(String placeId) async {
297 }
```

Project Manager

- project02 C:\Users\DELL\AndroidStudioProjects\project02
- > dart\_tool
- > idea
- > android [project02\_android]
- > assets
- > build
- > firebaseWork
- > flt
- > ios
- < lib
- > auth
- < dashboard
- < admin
- < screens
- < widgets
- CMakeLists.txt
- firebase\_options.dart
- main.dart
- > linux
- > macos
- > test

Id Variants Bookmarks Pull Requests Commit

Layout Inspector

project02 lib dashboard admin manageddetails.dart

```
try {
    await _firebase.collection('places').doc(placeId).delete();
    print('Place deleted successfully');
} catch (error) {
    print('Error deleting place: $error');
    throw error;
}

class EditPlaceScreen extends StatefulWidget {
    final Map<String, dynamic> place;

    const EditPlaceScreen({Key? key, required this.place}) : super(key: key);

    @override
    _EditPlaceScreenState createState() => _EditPlaceScreenState();
}

class _EditPlaceScreenState extends State<EditPlaceScreen> {
    late TextEditingController _nameController;
    late TextEditingController _cityController;

    @override
```

Resource Manager

- project02 C:\Users\DELL\Android
- > dart\_tool
- > idea
- > android [project02\_android]
- > assets
- > build
- > firebaseWork
- > flt
- > ios
- lib
  - auth
  - dashboard
    - admin
      - dashboard.dart
      - data.dart
      - firestore\_service.dart
      - manageddetails.dart
    - screens
      - home\_page.dart
      - post\_screen.dart
      - welcome\_screen.dart
    - widgets
      - CMakeLists.txt
      - firebase\_options.dart
      - main.dart
  - linux
  - macos
  - test

Bookmarks

Layout Inspector

project02 lib dashboard admin manageddetails.dart

```
void initState() {
    super.initState();
    _nameController = TextEditingController(text: widget.place['name']);
    _cityController = TextEditingController(text: widget.place['cityName']);
}

@Override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            title: Text('Edit Place'),
        ),
        body: Padding(
            padding: EdgeInsets.all(16),
            child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                    Text('Name:'),
                    TextFormField(
                        controller: _nameController,
                        decoration: InputDecoration(
                            hintText: 'Enter name',
                        ),
                    ),
                    TextFormField(
                        controller: _cityController,
                        decoration: InputDecoration(
                            hintText: 'Enter city',
                        ),
                    ),
                ],
            ),
        ),
    );
}
```

Resource Manager

- project02 C:\Users\DELL\Android
- > dart\_tool
- > idea
- > android [project02\_android]
- > assets
- > build
- > firebaseWork
- > flt
- > ios
- lib
  - auth
  - dashboard
    - admin
      - dashboard.dart
      - data.dart
      - firestore\_service.dart
      - manageddetails.dart
    - screens
      - home\_page.dart
      - post\_screen.dart
      - welcome\_screen.dart
    - widgets
      - CMakeLists.txt
      - firebase\_options.dart
      - main.dart
  - linux
  - macos
  - test

Bookmarks

Layout Inspector

project02 / lib / dashboard / admin / manageddetails.dart

```
346     SizedBox(height: 10),
347     Text('City:'),
348     TextFormField(
349       controller: _cityController,
350       decoration: InputDecoration(
351         hintText: 'Enter city',
352       ), // InputDecoration
353     ), // TextFormField
354     SizedBox(height: 10),
355     // Add more fields for editing other details
356     // You can use text fields, dropdowns, etc. to allow users to edit the details
357   ],
358   ), // Column
359   ), // Padding
360   floatingActionButton: FloatingActionButton(
361     onPressed: () {
362       _saveChanges();
363     },
364     child: Icon(Icons.save),
365   ), // FloatingActionButton
366 ); // Scaffold
367 }
368
369 void _saveChanges() async {
```

Git TODO Profiler Problems Terminal App Quality Insights Logcat Services App Inspection Run Dart Analysis Layout Inspector

project02 / lib / dashboard / admin / manageddetails.dart

```
370 // Get updated values from text controllers
371 String newName = _nameController.text;
372 String newCity = _cityController.text;

373 // Get the document ID of the place
374 String placeId = widget.place['id'];

375 // Print the details to verify
376 print('Place ID: $placeId');
377 print('New Name: $newName');
378 print('New City: $newCity');

379 // Update place details in Firestore
380 try {
381   await FirebaseFirestore.instance.collection('places').doc(placeId).update(
382     'name': newName,
383     'cityName': newCity,
384     // Add more fields to update as needed
385   );
386   print('Place details updated successfully');
387   // Close the screen and return to the previous screen
388   Navigator.pop(context);
389 } catch (error) {
390   print('Error updating place details: $error');
391 }
```

Git TODO Profiler Problems Terminal App Quality Insights Logcat Services App Inspection Run Dart Analysis Layout Inspector

project02\lib\dashboard\admin\manageddetails.dart

```
397
398
399 override
400 void dispose() {
401     _nameController.dispose();
402     _cityController.dispose();
403     super.dispose();
404 }
405
406 class PostScreen extends StatelessWidget {
407     @override
408     Widget build(BuildContext context) {
409         // Implement your post screen widget here
410         return Container();
411     }
412 }
413
414 class PlaceDetailsPage extends StatelessWidget {
415     final int placeIndex;
416
417     const PlaceDetailsPage({Key? key, required this.placeIndex}) : super(key: key);
418
419     @override
420     Widget build(BuildContext context) {
```

Project Manager | Resource Manager | Variants | Bookmarks | Full Requests | Commit | Project | Git | TODO | Profiler | Problems | Terminal | App Quality Insights | Logcat | Services | App Inspection | Run | Dart Analysis | Layout Inspector

project02\lib\dashboard\admin\manageddetails.dart

```
403     }
404 }
405
406 class PostScreen extends StatelessWidget {
407     @override
408     Widget build(BuildContext context) {
409         return Container();
410     }
411 }
412
413 class PlaceDetailsPage extends StatelessWidget {
414     final int placeIndex;
415
416     const PlaceDetailsPage({Key? key, required this.placeIndex}) : super(key: key);
417
418     @override
419     Widget build(BuildContext context) {
420         return Container();
421     }
422 }
423
```

Project Manager | Resource Manager | Variants | Bookmarks | Full Requests | Commit | Project | Git | TODO | Profiler | Problems | Terminal | App Quality Insights | Logcat | Services | App Inspection | Run | Dart Analysis | Layout Inspector

# FIREBASE INTERFACE OF DATA STORING :

The screenshot shows the Firebase Authentication interface for project01. On the left, the navigation bar includes Project Overview, Authentication (which is selected), App Check, Storage, Firestore Database, and Realtime Database. Below these are sections for What's new, Extensions (with a NEW badge), and Release Monitor (with a NEW badge). The main content area displays a table of users with columns: Identifier, Providers, Created, Signed In, and User UID. A search bar at the top allows filtering by email address, phone number, or user UID. A blue 'Add user' button is located at the top right of the table.

Identifier	Providers	Created	Signed In	User UID
myme@gmail.com	✉️	Mar 1, 2024	Mar 2, 2024	Vn5ISHGoMySsJLqsPC43Jl6...
ss@gmail.com	✉️	Feb 25, 2024	Feb 25, 2024	Kt7qbZZrjEQv9qRqqgsRQ8qY...
hh@gmail.com	✉️	Feb 24, 2024	Feb 26, 2024	Nb1jZUndVuXFY6yjcBlo5rEIM...
gag@gmail.com	✉️	Feb 18, 2024	Feb 18, 2024	xkua8xizhPhsZnPPutfuNEs3Z...
iam@gmail.com	✉️	Feb 14, 2024	Feb 14, 2024	j4d8apzxbzRi3fMZDBEDfyM2...
gg@gmail.com	✉️	Jan 11, 2024	Jan 12, 2024	ctLzmBg7ifMtUu8BJ7fuEG7...
aa@gmail.com	✉️	Jan 11, 2024	Jan 11, 2024	kBYQ54p2BcdnfJxFH3uusU08...
ghi@gmail.com	✉️	Jan 7, 2024	Jan 7, 2024	aWt2w6fosFXTpW2tgsUpU9c...

The screenshot shows the Firebase Storage interface for project01. The navigation bar includes Project Overview, Storage (selected), App Check, Authentication, Firestore Database, and Realtime Database. Below these are sections for What's new, Extensions (with a NEW badge), and Release Monitor (with a NEW badge). The main content area displays a table of files under gs://project01-146f2.appspot.com. The columns are Name, Size, Type, and Last modified. A blue 'Upload file' button is located at the top right of the table.

Name	Size	Type	Last modified
images/	-	Folder	-
profile_pictures/	-	Folder	-

More in Google Cloud ▾

places <span style="float: right;">⋮</span>	0lBG89Mhz9nPyALvmWv <span style="float: right;">⋮</span>
<b>+ Add document</b>	<b>+ Start collection</b>
<a href="#">0lBG89Mhz9nPyALvmWv</a> >	<b>+ Add field</b>
5h2IsAgEjqB7CDSeY9h4	country: "Colombia"
Gam65PLCyV50V2IAMm5D	description: "Colombia has major attractions for a tourist destination, such as Cartagena and its historic surroundings,"
Ojzrf0I01nJD4SRFNQjz	<b>+ images</b>
dWwRKyD95xL0H6cbH31B	0 "https://firebasestorage.googleapis.com/v0/b/project01- 146f2.appspot.com/o/images%2F1709396695412.jpg? alt=media&token=1e104b1b-20ed-4935-8fab-8a055238a2a6"
fIwBdGuDoXK1JrZC1Wlo	name : "Cartagena"
m8mCavYYgSUM5TjsNmXN	
obd5JgP1uIm0mE4RZrQZ	

**THANK  
YOU**