**Internship Week – 4, 5, 6**

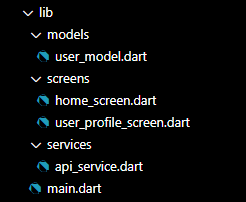
**Sajeela Ilyas**

**DHC – 679**

**GitHub:** [**https://github.com/sajeelailyas/**](https://github.com/sajeelailyas/)

2025

# **Week 4**



import 'package:flutter/material.dart';

import 'screens/home\_screen.dart';

void main() {

  runApp(MyApp());

}

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      title: 'API Integration Demo',

      theme: ThemeData(primarySwatch: Colors.blue),

      home: HomeScreen(),

    );

  }

}

import 'package:flutter/material.dart';

import '../models/user\_model.dart';

import '../services/api\_service.dart';

import 'user\_profile\_screen.dart';

class HomeScreen extends StatefulWidget {

  @override

  \_HomeScreenState createState() => \_HomeScreenState();

}

class \_HomeScreenState extends State<HomeScreen> {

  late Future<List<User>> \_users;

  @override

  void initState() {

    super.initState();

    \_users = ApiService.fetchUsers();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: Text("Users")),

      body: FutureBuilder<List<User>>(

        future: \_users,

        builder: (context, snapshot) {

          if (snapshot.connectionState == ConnectionState.waiting) {

            return Center(child: CircularProgressIndicator());

          } else if (snapshot.hasError) {

            return Center(child: Text("Error: ${snapshot.error}"));

          } else if (!snapshot.hasData || snapshot.data!.isEmpty) {

            return Center(child: Text("No users found"));

          }

          return ListView.builder(

            itemCount: snapshot.data!.length,

            itemBuilder: (context, index) {

              final user = snapshot.dataindex];

              return ListTile(

                leading: CircleAvatar(backgroundImage: NetworkImage(user.avatar)),

                title: Text(user.name),

                subtitle: Text(user.email),

                onTap: () => Navigator.push(

                  context,

                  MaterialPageRoute(builder: (\_) => UserProfileScreen(user: user)),

                ),

              );

            },

          );

        },

      ),

    );

  }

}

import 'package:flutter/material.dart';

import '../models/user\_model.dart';

class UserProfileScreen extends StatelessWidget {

  final User user;

  UserProfileScreen({required this.user});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: Text(user.name)),

      body: Padding(

        padding: const EdgeInsets.all(16.0),

        child: Column(

          children: [

            CircleAvatar(radius: 50, backgroundImage: NetworkImage(user.avatar)),

            SizedBox(height: 16),

            Text(user.name, style: TextStyle(fontSize: 24)),

            SizedBox(height: 8),

            Text(user.email, style: TextStyle(color: Colors.grey[600])),

          ],

        ),

      ),

    );

  }

}

import 'package:flutter/material.dart';

import '../models/user\_model.dart';

class UserProfileScreen extends StatelessWidget {

  final User user;

  UserProfileScreen({required this.user});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: Text(user.name)),

      body: Padding(

        padding: const EdgeInsets.all(16.0),

        child: Column(

          children: [

            CircleAvatar(radius: 50, backgroundImage: NetworkImage(user.avatar)),

            SizedBox(height: 16),

            Text(user.name, style: TextStyle(fontSize: 24)),

            SizedBox(height: 8),

            Text(user.email, style: TextStyle(color: Colors.grey[600])),

          ],

        ),

      ),

    );

  }

}

import 'dart:convert';

import 'package:http/http.dart' as http;

import '../models/user\_model.dart';

class ApiService {

  static const String url = 'https://jsonplaceholder.typicode.com/users';

  static Future<List<User>> fetchUsers() async {

    final response = await http.get(Uri.parse(url));

    if (response.statusCode == 200) {

      List jsonData = json.decode(response.body);

      return jsonData.map((user) => User.fromJson(user)).toList();

    } else {

      throw Exception('Failed to load users');

    }

  }

}

name: week5

description: "A new Flutter project."

# The following line prevents the package from being accidentally published to

# pub.dev using `flutter pub publish`. This is preferred for private packages.

publish\_to: 'none' # Remove this line if you wish to publish to pub.dev

# The following defines the version and build number for your application.

# A version number is three numbers separated by dots, like 1.2.43

# followed by an optional build number separated by a +.

# Both the version and the builder number may be overridden in flutter

# build by specifying --build-name and --build-number, respectively.

# In Android, build-name is used as versionName while build-number used as versionCode.

# Read more about Android versioning at https://developer.android.com/studio/publish/versioning

# In iOS, build-name is used as CFBundleShortVersionString while build-number is used as CFBundleVersion.

# Read more about iOS versioning at

# https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.html

# In Windows, build-name is used as the major, minor, and patch parts

# of the product and file versions while build-number is used as the build suffix.

version: 1.0.0+1

environment:

  sdk: ^3.7.2

# Dependencies specify other packages that your package needs in order to work.

# To automatically upgrade your package dependencies to the latest versions

# consider running `flutter pub upgrade --major-versions`. Alternatively,

# dependencies can be manually updated by changing the version numbers below to

# the latest version available on pub.dev. To see which dependencies have newer

# versions available, run `flutter pub outdated`.

dependencies:

  flutter:

    sdk: flutter

  # The following adds the Cupertino Icons font to your application.

  # Use with the CupertinoIcons class for iOS style icons.

  cupertino\_icons: ^1.0.8

  firebase\_core: ^3.15.1

  firebase\_auth: ^5.6.2

  cloud\_firestore: ^5.6.11

  firebase\_app\_check: ^0.3.2+9

  firebase\_crashlytics: ^4.3.9

dev\_dependencies:

  flutter\_test:

    sdk: flutter

  # The "flutter\_lints" package below contains a set of recommended lints to

  # encourage good coding practices. The lint set provided by the package is

  # activated in the `analysis\_options.yaml` file located at the root of your

  # package. See that file for information about deactivating specific lint

  # rules and activating additional ones.

  flutter\_lints: ^5.0.0

# For information on the generic Dart part of this file, see the

# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter packages.

flutter:

  # The following line ensures that the Material Icons font is

  # included with your application, so that you can use the icons in

  # the material Icons class.

  uses-material-design: true

  # To add assets to your application, add an assets section, like this:

  # assets:

  #   - images/a\_dot\_burr.jpeg

  #   - images/a\_dot\_ham.jpeg

  # An image asset can refer to one or more resolution-specific "variants", see

  # https://flutter.dev/to/resolution-aware-images

  # For details regarding adding assets from package dependencies, see

  # https://flutter.dev/to/asset-from-package

  # To add custom fonts to your application, add a fonts section here,

  # in this "flutter" section. Each entry in this list should have a

  # "family" key with the font family name, and a "fonts" key with a

  # list giving the asset and other descriptors for the font. For

  # example:

  # fonts:

  #   - family: Schyler

  #     fonts:

  #       - asset: fonts/Schyler-Regular.ttf

  #       - asset: fonts/Schyler-Italic.ttf

  #         style: italic

  #   - family: Trajan Pro

  #     fonts:

  #       - asset: fonts/TrajanPro.ttf

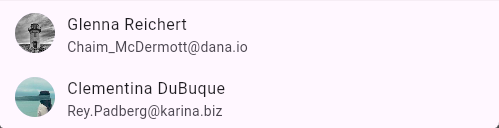
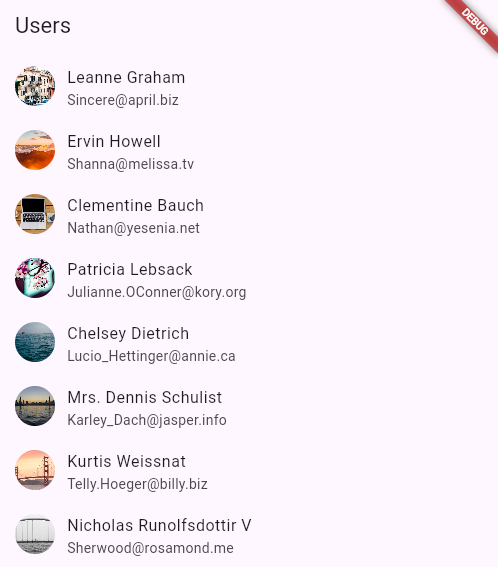
  #       - asset: fonts/TrajanPro\_Bold.ttf

  #         weight: 700

  #

  # For details regarding fonts from package dependencies,

  # see https://flutter.dev/to/font-from-package



![A screenshot of a phone

AI-generated content may be incorrect.

# **Week 5**



import 'package:firebase\_core/firebase\_core.dart';

import 'package:flutter/material.dart';

import 'firebase\_options.dart';

import 'screens/login.dart';

void main() async {

  WidgetsFlutterBinding.ensureInitialized();

  await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform);

  runApp(MyApp());

}

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      title: 'Week 5 - Firebase Auth',

      theme: ThemeData(primarySwatch: Colors.deepPurple),

      home: LoginScreen(),

      debugShowCheckedModeBanner: false,

    );

  }

}

import 'package:flutter/material.dart';

import 'package:week5/widgets/custom\_textfield.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

class SignupScreen extends StatefulWidget {

  @override

  \_SignupScreenState createState() => \_SignupScreenState();

}

class \_SignupScreenState extends State<SignupScreen> {

  final emailController = TextEditingController();

  final passwordController = TextEditingController();

  final nameController = TextEditingController();

  bool isLoading = false;

  void signupUser() async {

    setState(() {

      isLoading = true;

    });

    try {

      if (emailController.text.isEmpty ||

          passwordController.text.isEmpty ||

          nameController.text.isEmpty) {

        ScaffoldMessenger.of(context).showSnackBar(

          SnackBar(content: Text("Please fill all fields")),

        );

        return;

      }

      // Firebase Auth - Signup

      UserCredential userCredential = await FirebaseAuth.instance

          .createUserWithEmailAndPassword(

        email: emailController.text.trim(),

        password: passwordController.text.trim(),

      );

      // Firestore - Save user info

      await FirebaseFirestore.instance

          .collection('users')

          .doc(userCredential.user!.uid)

          .set({

        'name': nameController.text.trim(),

        'email': emailController.text.trim(),

        'uid': userCredential.user!.uid,

        'createdAt': Timestamp.now(),

      });

      ScaffoldMessenger.of(context).showSnackBar(

        SnackBar(content: Text("Signup successful!")),

      );

      Navigator.pop(context); // Go back to login screen

    } on FirebaseAuthException catch (e) {

      String message = "Signup failed.";

      if (e.code == 'email-already-in-use') {

        message = "Email already in use.";

      } else if (e.code == 'weak-password') {

        message = "Password should be at least 6 characters.";

      }

      ScaffoldMessenger.of(context).showSnackBar(

        SnackBar(content: Text(message)),

      );

    } catch (e) {

      print("Signup Error: $e");

      ScaffoldMessenger.of(context).showSnackBar(

        SnackBar(content: Text("An unexpected error occurred.")),

      );

    } finally {

      setState(() {

        isLoading = false;

      });

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: Text("Sign Up")),

      body: Padding(

        padding: const EdgeInsets.all(20.0),

        child: SingleChildScrollView(

          child: Column(

            children: [

              CustomTextField(

                hintText: 'Name',

                controller: nameController,

              ),

              CustomTextField(

                hintText: 'Email',

                controller: emailController,

              ),

              CustomTextField(

                hintText: 'Password',

                controller: passwordController,

                isPassword: true,

              ),

              SizedBox(height: 20),

              isLoading

                  ? CircularProgressIndicator()

                  : ElevatedButton(

                      onPressed: signupUser,

                      child: Text("Sign Up"),

                    ),

            ],

          ),

        ),

      ),

    );

  }

}

import 'package:flutter/material.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:week5/screens/user\_profile.dart';

import 'package:week5/screens/signup.dart';

import 'package:week5/widgets/custom\_textfield.dart';

class LoginScreen extends StatefulWidget {

  @override

  \_LoginScreenState createState() => \_LoginScreenState();

}

class \_LoginScreenState extends State<LoginScreen> {

  final emailController = TextEditingController();

  final passwordController = TextEditingController();

  void loginUser() async {

    try {

      await FirebaseAuth.instance.signInWithEmailAndPassword(

        email: emailController.text.trim(),

        password: passwordController.text.trim(),

      );

      Navigator.pushReplacement(

        context,

        MaterialPageRoute(builder: (context) => ProfileScreen()),

      );

    } catch (e) {

      print("Login Error: $e");

      ScaffoldMessenger.of(context).showSnackBar(

        SnackBar(content: Text("Login failed.")),

      );

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: Text("Login")),

      body: Padding(

        padding: const EdgeInsets.all(20.0),

        child: Column(

          children: [

            CustomTextField(

              hintText: 'Email',

              controller: emailController,

            ),

            CustomTextField(

              hintText: 'Password',

              controller: passwordController,

              isPassword: true,

            ),

            SizedBox(height: 20),

            ElevatedButton(

              onPressed: loginUser,

              child: Text("Login"),

            ),

            SizedBox(height: 10),

            TextButton(

              onPressed: () {

                Navigator.push(

                  context,

                  MaterialPageRoute(builder: (\_) => SignupScreen()),

                );

              },

              child: Text("Don't have an account? Sign Up"),

            ),

          ],

        ),

      ),

    );

  }

}

import 'package:flutter/material.dart';

import '../services/auth\_service.dart';

import 'login.dart';

class ProfileScreen extends StatefulWidget {

  @override

  State<ProfileScreen> createState() => \_ProfileScreenState();

}

class \_ProfileScreenState extends State<ProfileScreen> {

  final auth = AuthService();

  String? name;

  String? email;

  @override

  void initState() {

    super.initState();

    loadUser();

  }

  void loadUser() async {

    final data = await auth.getUserData();

    setState(() {

      name = data?['name'];

      email = data?['email'];

    });

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Profile'),

        actions: [

          IconButton(

            icon: Icon(Icons.logout),

            onPressed: () async {

              await auth.logout();

              Navigator.pushReplacement(context, MaterialPageRoute(builder: (\_) => LoginScreen()));

            },

          )

        ],

      ),

      body: name == null

          ? Center(child: CircularProgressIndicator())

          : Padding(

              padding: const EdgeInsets.all(20),

              child: Column(

                children: [

                  CircleAvatar(radius: 40, child: Icon(Icons.person, size: 50)),

                  SizedBox(height: 20),

                  Text("Name: $name", style: TextStyle(fontSize: 20)),

                  SizedBox(height: 10),

                  Text("Email: $email", style: TextStyle(fontSize: 16)),

                ],

              ),

            ),

    );

  }

}

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

class AuthService {

  final FirebaseAuth \_auth = FirebaseAuth.instance;

  final FirebaseFirestore \_db = FirebaseFirestore.instance;

  Future<String?> signUp(String name, String email, String password) async {

    try {

      UserCredential cred = await \_auth.createUserWithEmailAndPassword(

        email: email,

        password: password,

      );

      await \_db.collection('users').doc(cred.user!.uid).set({

        'name': name,

        'email': email,

      });

      return null;

    } catch (e) {

      return e.toString();

    }

  }

  Future<String?> login(String email, String password) async {

    try {

      await \_auth.signInWithEmailAndPassword(email: email, password: password);

      return null;

    } catch (e) {

      return e.toString();

    }

  }

  Future<void> logout() async {

    await \_auth.signOut();

  }

  Future<Map<String, dynamic>?> getUserData() async {

    final uid = \_auth.currentUser?.uid;

    if (uid != null) {

      DocumentSnapshot doc = await \_db.collection('users').doc(uid).get();

      return doc.data() as Map<String, dynamic>?;

    }

    return null;

  }

}

import 'package:flutter/material.dart';

class CustomTextField extends StatelessWidget {

  final String hintText;

  final TextEditingController controller;

  final bool isPassword;

  const CustomTextField({

    Key? key,

    required this.hintText,

    required this.controller,

    this.isPassword = false,

  }) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return Padding(

      padding: const EdgeInsets.symmetric(vertical: 8.0),

      child: TextField(

        controller: controller,

        obscureText: isPassword,

        decoration: InputDecoration(

          hintText: hintText,

          border: OutlineInputBorder(

            borderRadius: BorderRadius.circular(12),

          ),

          filled: true,

          fillColor: Colors.grey[200],

          contentPadding: const EdgeInsets.symmetric(horizontal: 16, vertical: 12),

        ),

      ),

    );

  }

}

// File generated by FlutterFire CLI.

// ignore\_for\_file: type=lint

import 'package:firebase\_core/firebase\_core.dart' show FirebaseOptions;

import 'package:flutter/foundation.dart'

    show defaultTargetPlatform, kIsWeb, TargetPlatform;

/// Default [FirebaseOptions] for use with your Firebase apps.

///

/// Example:

/// ```dart

/// import 'firebase\_options.dart';

/// // ...

/// await Firebase.initializeApp(

///   options: DefaultFirebaseOptions.currentPlatform,

/// );

/// ```

class DefaultFirebaseOptions {

  static FirebaseOptions get currentPlatform {

    if (kIsWeb) {

      return web;

    }

    switch (defaultTargetPlatform) {

      case TargetPlatform.android:

        return android;

      case TargetPlatform.iOS:

        return ios;

      case TargetPlatform.macOS:

        return macos;

      case TargetPlatform.windows:

        return windows;

      case TargetPlatform.linux:

        throw UnsupportedError(

          'DefaultFirebaseOptions have not been configured for linux - '

          'you can reconfigure this by running the FlutterFire CLI again.',

        );

      default:

        throw UnsupportedError(

          'DefaultFirebaseOptions are not supported for this platform.',

        );

    }

  }

  static const FirebaseOptions web = FirebaseOptions(

    apiKey: 'AIzaSyDLYuidKb74NS2e8ZRhBzi6eFQODhoUJyc',

    appId: '1:598435078298:web:0c1cb9676c5e128b06bf23',

    messagingSenderId: '598435078298',

    projectId: 'fir-auth-d52a7',

    authDomain: 'fir-auth-d52a7.firebaseapp.com',

    storageBucket: 'fir-auth-d52a7.firebasestorage.app',

    measurementId: 'G-R5N6P4KD6M',

  );

  static const FirebaseOptions android = FirebaseOptions(

    apiKey: 'AIzaSyCCAiJJerq5BZhz1uUNztlydRsZgqRxuho',

    appId: '1:598435078298:android:d13be18dd1859ee806bf23',

    messagingSenderId: '598435078298',

    projectId: 'fir-auth-d52a7',

    storageBucket: 'fir-auth-d52a7.firebasestorage.app',

  );

  static const FirebaseOptions ios = FirebaseOptions(

    apiKey: 'AIzaSyCVEtsLaQuyl1XT5A6Cnq8WY\_HPGEp1hts',

    appId: '1:598435078298:ios:f6f552261ae8c98a06bf23',

    messagingSenderId: '598435078298',

    projectId: 'fir-auth-d52a7',

    storageBucket: 'fir-auth-d52a7.firebasestorage.app',

    iosBundleId: 'com.example.week5',

  );

  static const FirebaseOptions macos = FirebaseOptions(

    apiKey: 'AIzaSyCVEtsLaQuyl1XT5A6Cnq8WY\_HPGEp1hts',

    appId: '1:598435078298:ios:f6f552261ae8c98a06bf23',

    messagingSenderId: '598435078298',

    projectId: 'fir-auth-d52a7',

    storageBucket: 'fir-auth-d52a7.firebasestorage.app',

    iosBundleId: 'com.example.week5',

  );

  static const FirebaseOptions windows = FirebaseOptions(

    apiKey: 'AIzaSyDLYuidKb74NS2e8ZRhBzi6eFQODhoUJyc',

    appId: '1:598435078298:web:be4b7d675f32fbeb06bf23',

    messagingSenderId: '598435078298',

    projectId: 'fir-auth-d52a7',

    authDomain: 'fir-auth-d52a7.firebaseapp.com',

    storageBucket: 'fir-auth-d52a7.firebasestorage.app',

    measurementId: 'G-LN11GQV67X',

  );

}

name: week5

description: "A new Flutter project."

# The following line prevents the package from being accidentally published to

# pub.dev using `flutter pub publish`. This is preferred for private packages.

publish\_to: 'none' # Remove this line if you wish to publish to pub.dev

# The following defines the version and build number for your application.

# A version number is three numbers separated by dots, like 1.2.43

# followed by an optional build number separated by a +.

# Both the version and the builder number may be overridden in flutter

# build by specifying --build-name and --build-number, respectively.

# In Android, build-name is used as versionName while build-number used as versionCode.

# Read more about Android versioning at https://developer.android.com/studio/publish/versioning

# In iOS, build-name is used as CFBundleShortVersionString while build-number is used as CFBundleVersion.

# Read more about iOS versioning at

# https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.html

# In Windows, build-name is used as the major, minor, and patch parts

# of the product and file versions while build-number is used as the build suffix.

version: 1.0.0+1

environment:

  sdk: ^3.7.2

# Dependencies specify other packages that your package needs in order to work.

# To automatically upgrade your package dependencies to the latest versions

# consider running `flutter pub upgrade --major-versions`. Alternatively,

# dependencies can be manually updated by changing the version numbers below to

# the latest version available on pub.dev. To see which dependencies have newer

# versions available, run `flutter pub outdated`.

dependencies:

  flutter:

    sdk: flutter

  # The following adds the Cupertino Icons font to your application.

  # Use with the CupertinoIcons class for iOS style icons.

  cupertino\_icons: ^1.0.8

  firebase\_core: ^3.15.1

  firebase\_auth: ^5.6.2

  cloud\_firestore: ^5.6.11

  firebase\_app\_check: ^0.3.2+9

  firebase\_crashlytics: ^4.3.9

dev\_dependencies:

  flutter\_test:

    sdk: flutter

  # The "flutter\_lints" package below contains a set of recommended lints to

  # encourage good coding practices. The lint set provided by the package is

  # activated in the `analysis\_options.yaml` file located at the root of your

  # package. See that file for information about deactivating specific lint

  # rules and activating additional ones.

  flutter\_lints: ^5.0.0

# For information on the generic Dart part of this file, see the

# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter packages.

flutter:

  # The following line ensures that the Material Icons font is

  # included with your application, so that you can use the icons in

  # the material Icons class.

  uses-material-design: true

  # To add assets to your application, add an assets section, like this:

  # assets:

  #   - images/a\_dot\_burr.jpeg

  #   - images/a\_dot\_ham.jpeg

  # An image asset can refer to one or more resolution-specific "variants", see

  # https://flutter.dev/to/resolution-aware-images

  # For details regarding adding assets from package dependencies, see

  # https://flutter.dev/to/asset-from-package

  # To add custom fonts to your application, add a fonts section here,

  # in this "flutter" section. Each entry in this list should have a

  # "family" key with the font family name, and a "fonts" key with a

  # list giving the asset and other descriptors for the font. For

  # example:

  # fonts:

  #   - family: Schyler

  #     fonts:

  #       - asset: fonts/Schyler-Regular.ttf

  #       - asset: fonts/Schyler-Italic.ttf

  #         style: italic

  #   - family: Trajan Pro

  #     fonts:

  #       - asset: fonts/TrajanPro.ttf

  #       - asset: fonts/TrajanPro\_Bold.ttf

  #         weight: 700

  #

  # For details regarding fonts from package dependencies,

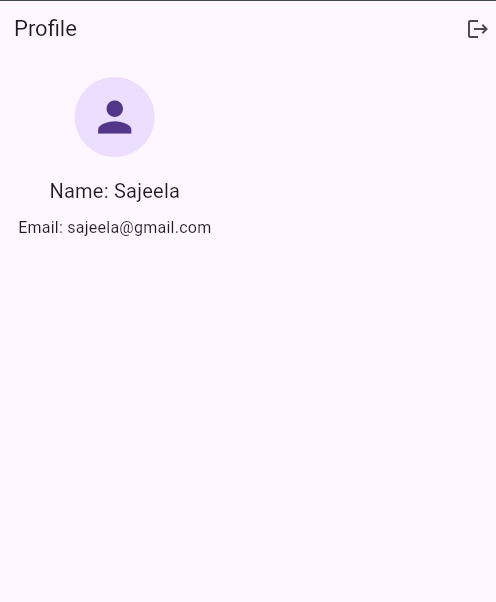
  # see https://flutter.dev/to/font-from-package

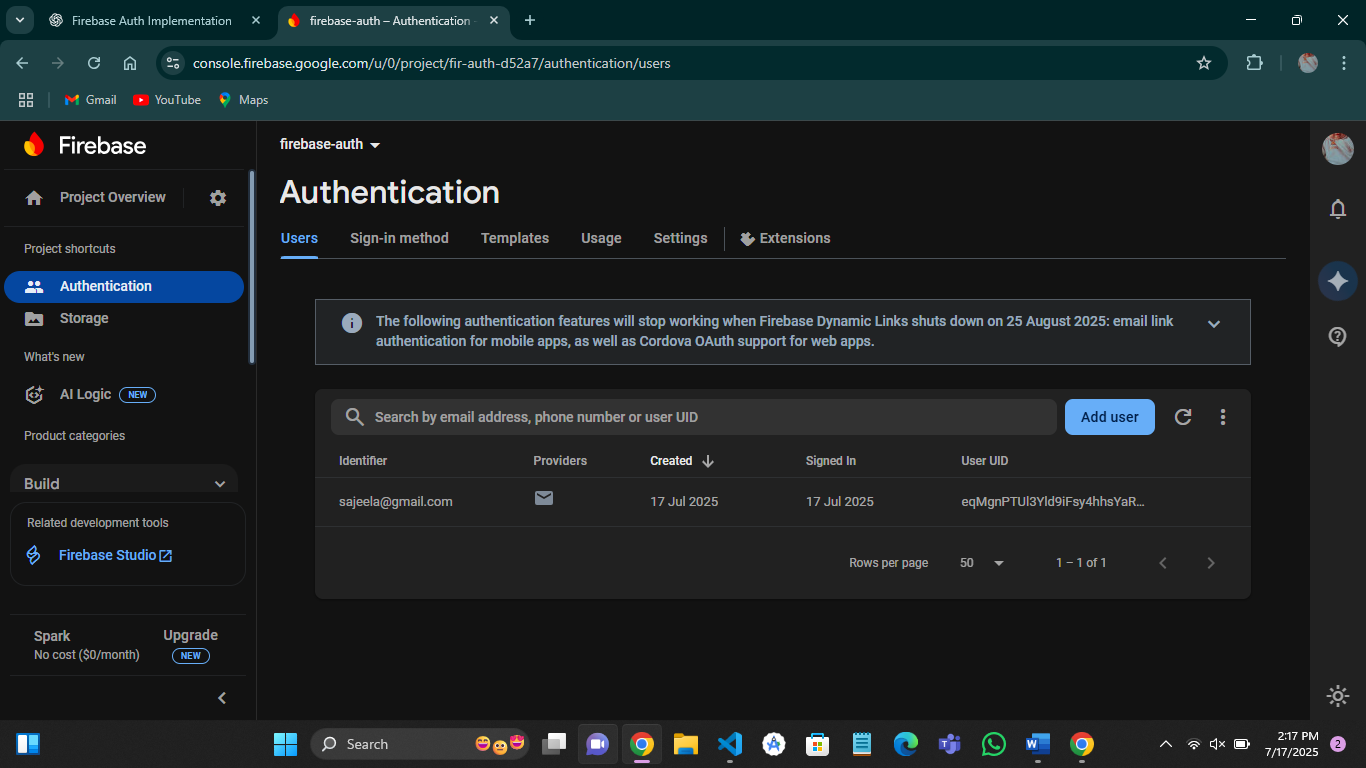
A screenshot of a login form

AI-generated content may be incorrect. A screenshot of a login form

AI-generated content may be incorrect.

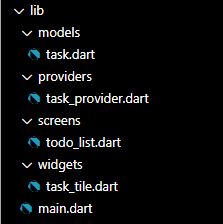
A screenshot of a login form

AI-generated content may be incorrect. 



# **Week 6**

# **To-Do List**



import 'package:flutter/material.dart';

import 'package:provider/provider.dart';

import 'providers/task\_provider.dart';

import 'screens/todo\_list.dart';

void main() {

  runApp(const TodoApp());

}

class TodoApp extends StatelessWidget {

  const TodoApp({super.key});

  @override

  Widget build(BuildContext context) {

    return ChangeNotifierProvider(

      create: (\_) => TaskProvider(),

      child: MaterialApp(

        title: 'To-Do List',

        debugShowCheckedModeBanner: false,

        theme: ThemeData(

          primaryColor: const Color(0xFF006400), // Dark Green

          scaffoldBackgroundColor: const Color(0xFFF5F5DC), // Beige

          appBarTheme: const AppBarTheme(

            backgroundColor: Color(0xFF006400), // Dark Green

            foregroundColor: Colors.white,

          ),

          elevatedButtonTheme: ElevatedButtonThemeData(

            style: ElevatedButton.styleFrom(

              backgroundColor: const Color(0xFF006400), // Dark Green

              foregroundColor: Colors.white,

              shape: RoundedRectangleBorder(

                borderRadius: BorderRadius.circular(10),

              ),

            ),

          ),

          inputDecorationTheme: InputDecorationTheme(

            filled: true,

            fillColor: Colors.white,

            labelStyle: const TextStyle(color: Colors.black87),

            enabledBorder: OutlineInputBorder(

              borderSide: const BorderSide(color: Colors.black26),

              borderRadius: BorderRadius.circular(12),

            ),

            focusedBorder: OutlineInputBorder(

              borderSide: const BorderSide(color: Colors.black),

              borderRadius: BorderRadius.circular(12),

            ),

          ),

          iconTheme: const IconThemeData(color: Colors.black),

        ),

        home: const TodoListPage(),

      ),

    );

  }

}

import 'package:flutter/material.dart';

import 'package:provider/provider.dart';

import '../providers/task\_provider.dart';

import '../widgets/task\_tile.dart';

class TodoListPage extends StatelessWidget {

  const TodoListPage({super.key});

  @override

  Widget build(BuildContext context) {

    final taskProvider = Provider.of<TaskProvider>(context);

    final taskController = TextEditingController();

    void \_addTask() {

      final task = taskController.text.trim();

      if (task.isNotEmpty) {

        taskProvider.addTask(task);

        taskController.clear();

      }

    }

    return Scaffold(

      appBar: AppBar(title: const Text('To-Do List')),

      body: Column(

        children: [

          Padding(

            padding: const EdgeInsets.all(16),

            child: Row(

              children: [

                Expanded(

                  child: TextField(

                    controller: taskController,

                    decoration: const InputDecoration(

                      labelText: 'Enter new task',

                    ),

                    onSubmitted: (\_) => \_addTask(),

                  ),

                ),

                const SizedBox(width: 10),

                ElevatedButton(

                  onPressed: \_addTask,

                  child: const Text('Add'),

                ),

              ],

            ),

          ),

          Expanded(

            child: Consumer<TaskProvider>(

              builder: (context, provider, child) {

                final tasks = provider.tasks;

                return tasks.isEmpty

                    ? const Center(

                        child: Text(

                          'No tasks added yet.',

                          style: TextStyle(fontSize: 18, color: Colors.black54),

                        ),

                      )

                    : ListView.separated(

                        itemCount: tasks.length,

                        separatorBuilder: (\_, \_\_) => const SizedBox(height: 4),

                        itemBuilder: (context, index) => Padding(

                          padding: const EdgeInsets.symmetric(

                              horizontal: 16, vertical: 4),

                          child: TaskTile(

                            task: tasks[index],

                            onDelete: () => provider.removeTask(index),

                          ),

                        ),

                      );

              },

            ),

          ),

        ],

      ),

    );

  }

}

import 'package:flutter/material.dart';

class TaskTile extends StatelessWidget {

  final String task;

  final VoidCallback onDelete;

  const TaskTile({super.key, required this.task, required this.onDelete});

  @override

  Widget build(BuildContext context) {

    return Card(

      color: Colors.white,

      elevation: 1,

      shape: RoundedRectangleBorder(

        borderRadius: BorderRadius.circular(10),

      ),

      child: ListTile(

        title: Text(task),

        trailing: IconButton(

          icon: const Icon(Icons.delete),

          onPressed: onDelete,

        ),

      ),

    );

  }

}

import 'package:flutter/foundation.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

class TaskProvider with ChangeNotifier {

  final List<String> \_tasks = [];

  List<String> get tasks => \_tasks;

  static const String tasksKey = 'tasks';

  TaskProvider() {

    \_loadTasks();

  }

  Future<void> \_loadTasks() async {

    final prefs = await SharedPreferences.getInstance();

    final savedTasks = prefs.getStringList(tasksKey);

    if (savedTasks != null) {

      \_tasks.addAll(savedTasks);

      notifyListeners();

    }

  }

  Future<void> \_saveTasks() async {

    final prefs = await SharedPreferences.getInstance();

    await prefs.setStringList(tasksKey, \_tasks);

  }

  void addTask(String task) {

    \_tasks.add(task);

    \_saveTasks();

    notifyListeners();

  }

  void removeTask(int index) {

    \_tasks.removeAt(index);

    \_saveTasks();

    notifyListeners();

  }

}

class TodoTask {

  String title;

  bool isDone;

  TodoTask({required this.title, this.isDone = false});

  // For saving to shared\_preferences

  Map<String, dynamic> toJson() => {

        'title': title,

        'isDone': isDone,

      };

  factory TodoTask.fromJson(Map<String, dynamic> json) => TodoTask(

        title: json['title'],

        isDone: json['isDone'],

      );

}

name: to\_do\_list\_app

description: "A new Flutter project."

# The following line prevents the package from being accidentally published to

# pub.dev using `flutter pub publish`. This is preferred for private packages.

publish\_to: 'none' # Remove this line if you wish to publish to pub.dev

# The following defines the version and build number for your application.

# A version number is three numbers separated by dots, like 1.2.43

# followed by an optional build number separated by a +.

# Both the version and the builder number may be overridden in flutter

# build by specifying --build-name and --build-number, respectively.

# In Android, build-name is used as versionName while build-number used as versionCode.

# Read more about Android versioning at https://developer.android.com/studio/publish/versioning

# In iOS, build-name is used as CFBundleShortVersionString while build-number is used as CFBundleVersion.

# Read more about iOS versioning at

# https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.html

# In Windows, build-name is used as the major, minor, and patch parts

# of the product and file versions while build-number is used as the build suffix.

version: 1.0.0+1

environment:

  sdk: ^3.7.2

# Dependencies specify other packages that your package needs in order to work.

# To automatically upgrade your package dependencies to the latest versions

# consider running `flutter pub upgrade --major-versions`. Alternatively,

# dependencies can be manually updated by changing the version numbers below to

# the latest version available on pub.dev. To see which dependencies have newer

# versions available, run `flutter pub outdated`.

dependencies:

  flutter:

    sdk: flutter

  provider: ^6.1.1

  shared\_preferences: ^2.2.2

  # The following adds the Cupertino Icons font to your application.

  # Use with the CupertinoIcons class for iOS style icons.

  cupertino\_icons: ^1.0.8

dev\_dependencies:

  flutter\_test:

    sdk: flutter

  # The "flutter\_lints" package below contains a set of recommended lints to

  # encourage good coding practices. The lint set provided by the package is

  # activated in the `analysis\_options.yaml` file located at the root of your

  # package. See that file for information about deactivating specific lint

  # rules and activating additional ones.

  flutter\_lints: ^5.0.0

# For information on the generic Dart part of this file, see the

# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter packages.

flutter:

  # The following line ensures that the Material Icons font is

  # included with your application, so that you can use the icons in

  # the material Icons class.

  uses-material-design: true

  # To add assets to your application, add an assets section, like this:

  # assets:

  #   - images/a\_dot\_burr.jpeg

  #   - images/a\_dot\_ham.jpeg

  # An image asset can refer to one or more resolution-specific "variants", see

  # https://flutter.dev/to/resolution-aware-images

  # For details regarding adding assets from package dependencies, see

  # https://flutter.dev/to/asset-from-package

  # To add custom fonts to your application, add a fonts section here,

  # in this "flutter" section. Each entry in this list should have a

  # "family" key with the font family name, and a "fonts" key with a

  # list giving the asset and other descriptors for the font. For

  # example:

  # fonts:

  #   - family: Schyler

  #     fonts:

  #       - asset: fonts/Schyler-Regular.ttf

  #       - asset: fonts/Schyler-Italic.ttf

  #         style: italic

  #   - family: Trajan Pro

  #     fonts:

  #       - asset: fonts/TrajanPro.ttf

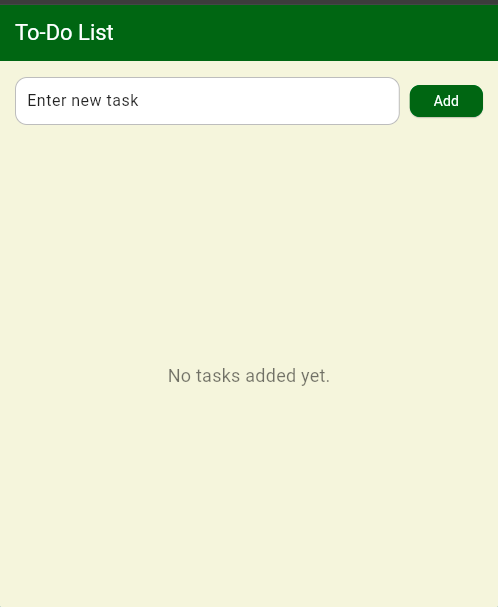
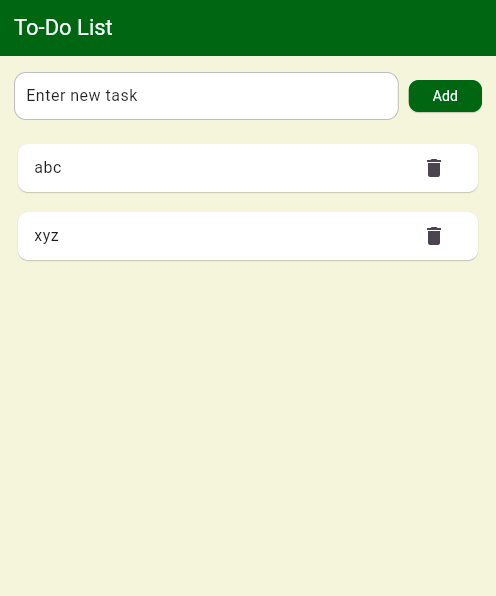
  #       - asset: fonts/TrajanPro\_Bold.ttf

  #         weight: 700

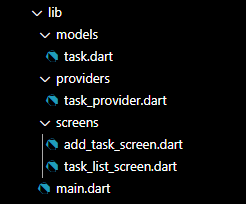
  #

  # For details regarding fonts from package dependencies,

  # see https://flutter.dev/to/font-from-package

# **Task Management**



import 'package:flutter/material.dart';

import 'package:provider/provider.dart';

import 'providers/task\_provider.dart';

import 'screens/task\_list\_screen.dart';

void main() {

  runApp(TaskApp());

}

class TaskApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return ChangeNotifierProvider(

      create: (\_) => TaskProvider(),

      child: MaterialApp(

        debugShowCheckedModeBanner: false,

        home: TaskListScreen(),

      ),

    );

  }

}

import 'package:flutter/material.dart';

import 'package:provider/provider.dart';

import '../providers/task\_provider.dart';

import 'add\_task\_screen.dart';

class TaskListScreen extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    final taskProvider = Provider.of<TaskProvider>(context);

    return Scaffold(

      appBar: AppBar(title: const Text("Task Manager")),

      body: ListView.builder(

        itemCount: taskProvider.tasks.length,

        itemBuilder: (ctx, index) {

          final task = taskProvider.tasks[index];

          return ListTile(

            title: Text(

              task.title,

              style: TextStyle(

                decoration: task.isCompleted

                    ? TextDecoration.lineThrough

                    : TextDecoration.none,

              ),

            ),

            trailing: Row(mainAxisSize: MainAxisSize.min, children: [

              IconButton(

                icon: Icon(task.isCompleted ? Icons.check\_box : Icons.check\_box\_outline\_blank),

                onPressed: () => taskProvider.toggleCompletion(task.id),

              ),

              IconButton(

                icon: const Icon(Icons.delete, color: Colors.red),

                onPressed: () => taskProvider.deleteTask(task.id),

              ),

            ]),

          );

        },

      ),

      floatingActionButton: FloatingActionButton(

        onPressed: () => Navigator.push(context,

            MaterialPageRoute(builder: (\_) => AddTaskScreen())),

        child: const Icon(Icons.add),

      ),

    );

  }

}

import 'package:flutter/material.dart';

import 'package:provider/provider.dart';

import '../providers/task\_provider.dart';

class AddTaskScreen extends StatelessWidget {

  final \_controller = TextEditingController();

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: const Text("Add New Task")),

      body: Padding(

        padding: const EdgeInsets.all(16.0),

        child: Column(children: [

          TextField(

            controller: \_controller,

            decoration: const InputDecoration(labelText: "Task Title"),

          ),

          const SizedBox(height: 20),

          ElevatedButton(

            onPressed: () {

              if (\_controller.text.trim().isNotEmpty) {

                Provider.of<TaskProvider>(context, listen: false)

                    .addTask(\_controller.text.trim());

                Navigator.pop(context);

              }

            },

            child: const Text("Add Task"),

          )

        ]),

      ),

    );

  }

}

import 'package:flutter/material.dart';

import '../models/task.dart';

class TaskProvider with ChangeNotifier {

  final List<Task> \_tasks = [];

  List<Task> get tasks => \_tasks;

  void addTask(String title) {

    \_tasks.add(Task(id: DateTime.now().toString(), title: title));

    notifyListeners();

  }

  void deleteTask(String id) {

    \_tasks.removeWhere((task) => task.id == id);

    notifyListeners();

  }

  void toggleCompletion(String id) {

    final task = \_tasks.firstWhere((task) => task.id == id);

    task.isCompleted = !task.isCompleted;

    notifyListeners();

  }

}

class Task {

  String id;

  String title;

  bool isCompleted;

  Task({required this.id, required this.title, this.isCompleted = false});

}

name: task\_mng\_app

description: "A new Flutter project."

# The following line prevents the package from being accidentally published to

# pub.dev using `flutter pub publish`. This is preferred for private packages.

publish\_to: 'none' # Remove this line if you wish to publish to pub.dev

# The following defines the version and build number for your application.

# A version number is three numbers separated by dots, like 1.2.43

# followed by an optional build number separated by a +.

# Both the version and the builder number may be overridden in flutter

# build by specifying --build-name and --build-number, respectively.

# In Android, build-name is used as versionName while build-number used as versionCode.

# Read more about Android versioning at https://developer.android.com/studio/publish/versioning

# In iOS, build-name is used as CFBundleShortVersionString while build-number is used as CFBundleVersion.

# Read more about iOS versioning at

# https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.html

# In Windows, build-name is used as the major, minor, and patch parts

# of the product and file versions while build-number is used as the build suffix.

version: 1.0.0+1

environment:

  sdk: ^3.7.2

# Dependencies specify other packages that your package needs in order to work.

# To automatically upgrade your package dependencies to the latest versions

# consider running `flutter pub upgrade --major-versions`. Alternatively,

# dependencies can be manually updated by changing the version numbers below to

# the latest version available on pub.dev. To see which dependencies have newer

# versions available, run `flutter pub outdated`.

dependencies:

  flutter:

    sdk: flutter

  shared\_preferences: ^2.1.1

  # The following adds the Cupertino Icons font to your application.

  # Use with the CupertinoIcons class for iOS style icons.

  cupertino\_icons: ^1.0.8

dev\_dependencies:

  flutter\_test:

    sdk: flutter

  provider: ^6.1.1

  shared\_preferences: ^2.2.2

  # The "flutter\_lints" package below contains a set of recommended lints to

  # encourage good coding practices. The lint set provided by the package is

  # activated in the `analysis\_options.yaml` file located at the root of your

  # package. See that file for information about deactivating specific lint

  # rules and activating additional ones.

  flutter\_lints: ^5.0.0

# For information on the generic Dart part of this file, see the

# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter packages.

flutter:

  # The following line ensures that the Material Icons font is

  # included with your application, so that you can use the icons in

  # the material Icons class.

  uses-material-design: true

  # To add assets to your application, add an assets section, like this:

  # assets:

  #   - images/a\_dot\_burr.jpeg

  #   - images/a\_dot\_ham.jpeg

  # An image asset can refer to one or more resolution-specific "variants", see

  # https://flutter.dev/to/resolution-aware-images

  # For details regarding adding assets from package dependencies, see

  # https://flutter.dev/to/asset-from-package

  # To add custom fonts to your application, add a fonts section here,

  # in this "flutter" section. Each entry in this list should have a

  # "family" key with the font family name, and a "fonts" key with a

  # list giving the asset and other descriptors for the font. For

  # example:

  # fonts:

  #   - family: Schyler

  #     fonts:

  #       - asset: fonts/Schyler-Regular.ttf

  #       - asset: fonts/Schyler-Italic.ttf

  #         style: italic

  #   - family: Trajan Pro

  #     fonts:

  #       - asset: fonts/TrajanPro.ttf

  #       - asset: fonts/TrajanPro\_Bold.ttf

  #         weight: 700

  #

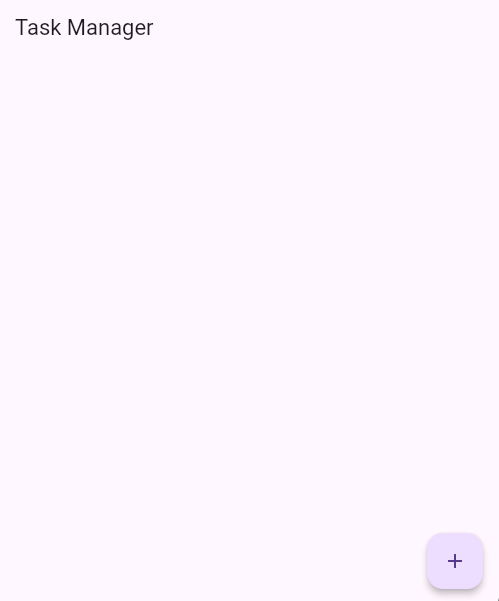
  # For details regarding fonts from package dependencies,

  # see https://flutter.dev/to/font-from-package

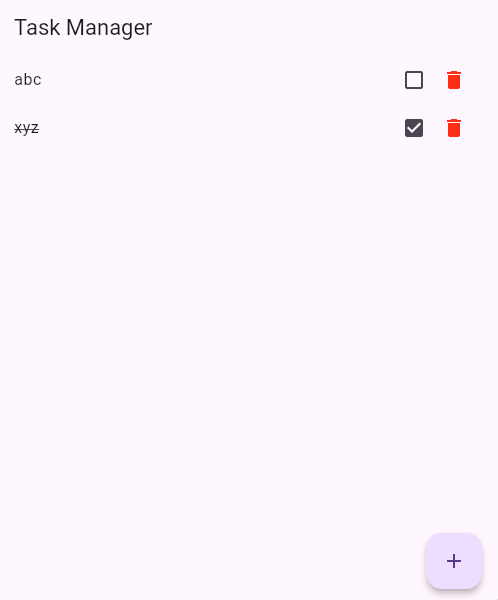
flutter\_native\_splash:

  color: "#ffffff"

  image: assets/splash.png

 A screenshot of a task

AI-generated content may be incorrect.

 A screenshot of a cell phone

AI-generated content may be incorrect.