Experiment No.6
MAD & PWA LAB

I. Aim: To Connect Flutter UI with firebase database.

II. Theory:

Firebase is a comprehensive application development platform backed by Google, supporting the creation of iOS, Android, and web apps. It offers various services, including analytics, authentication, cloud messaging, realtime database, crash reporting, performance monitoring, and testing. Firebase Analytics provides detailed reporting on user behavior, helping developers make informed decisions to improve app performance and marketing strategies. Firebase Authentication simplifies the implementation of secure authentication systems, supporting various login methods like email, phone, Google, Facebook, and more.

Firebase Cloud Messaging enables reliable message delivery across iOS, Android, and web platforms. The Realtime Database is a cloud-hosted NoSQL database that syncs data in real time between users and works seamlessly offline. Firebase Crashlytics provides real-time crash reporting, helping developers identify and fix stability issues quickly.

Firebase offers a number of services, including:

1. **Analytics** – Google Analytics for Firebase offers free, unlimited reporting on as many as 500 separate events. Analytics presents data about user behavior in iOS and Android apps, enabling better decision-making about improving performance and app marketing.

2. **Authentication** – Firebase Authentication makes it easy for developers to build secure authentication systems and enhances the sign-in and onboarding experience for users. This feature offers a complete identity solution, supporting email and password accounts, phone auth, as well as Google, Facebook, GitHub, Twitter login and more.

3. **Cloud messaging** – Firebase Cloud Messaging (FCM) is a cross-platform messaging tool that lets companies reliably receive and deliver messages on iOS, Android and the web at no cost.

4. **Realtime database** – the Firebase Realtime Database is a cloud-hosted NoSQL database that enables data to be stored and synced between users in real time. The data is synced across all clients in real time and is still available when an app goes offline.

5. **Crashlytics** – Firebase Crashlytics is a real-time crash reporter that helps developers track, prioritize and fix stability issues that reduce the quality of their apps. With crashlytics, developers spend less time organizing and troubleshooting crashes and more time building features for their apps.

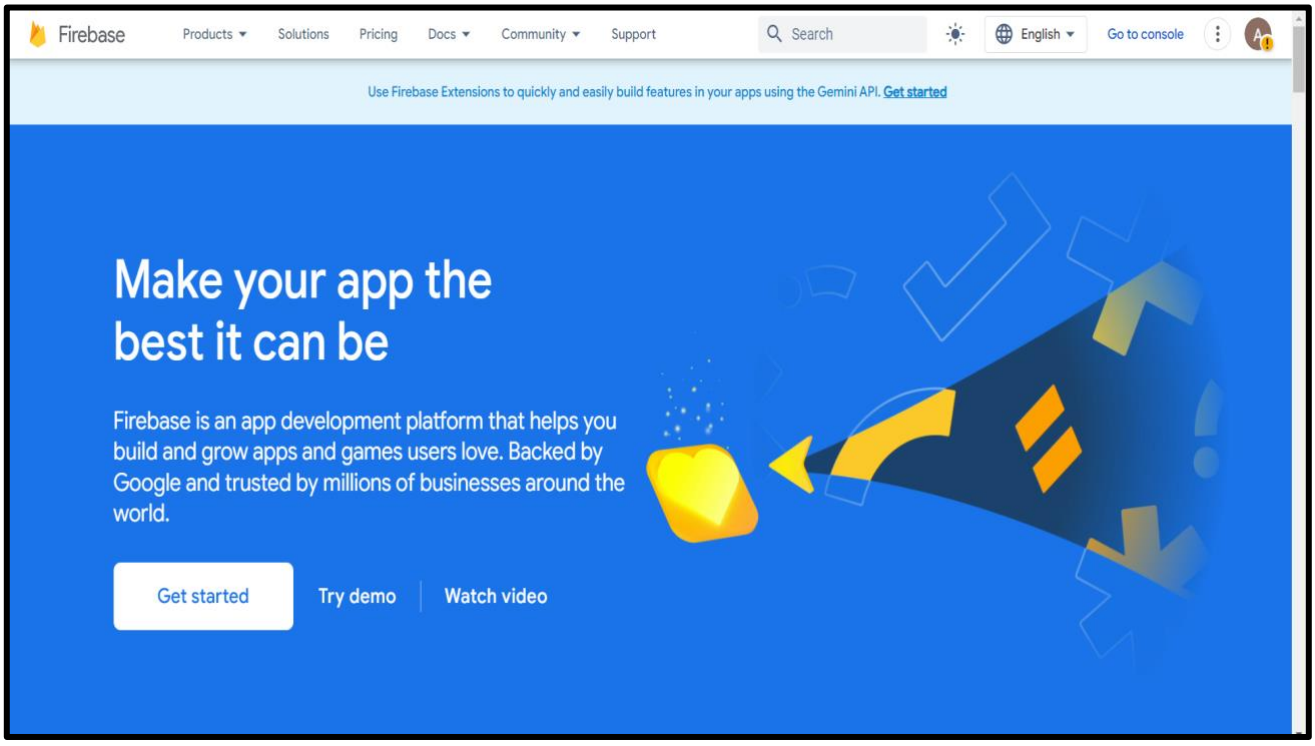
6. **Performance** – Firebase Performance Monitoring service gives developers insight into the performance characteristics of their iOS and Android apps to help them determine where and when the performance of their apps can be improved.

7. **Test lab** – Firebase Test Lab is a cloud-based app-testing infrastructure. With one operation, developers can test their iOS or Android apps across a variety of devices and device configurations. They can see the results, including videos, screenshots and logs, in the Firebase console.

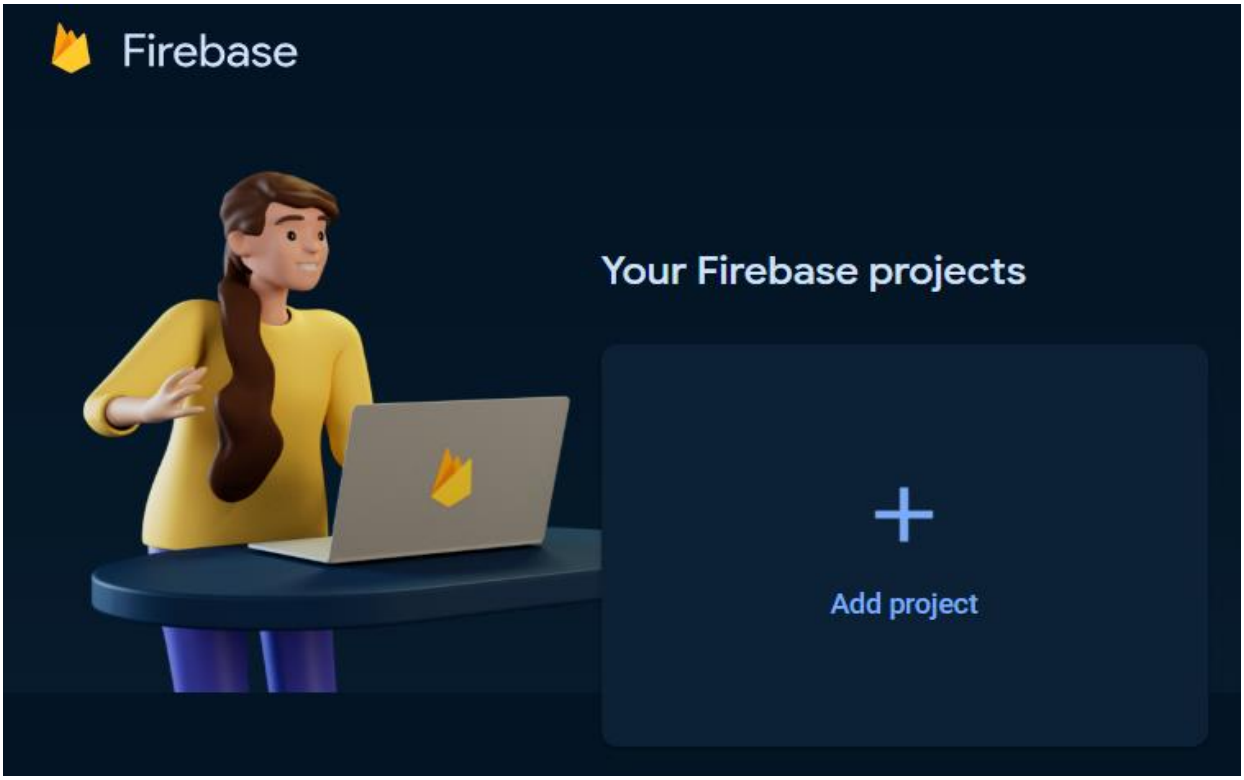
Creating a Firebase Project

Add Firebase to your existing Google Cloud project:

1. Log in to the Firebase console, then click Add project

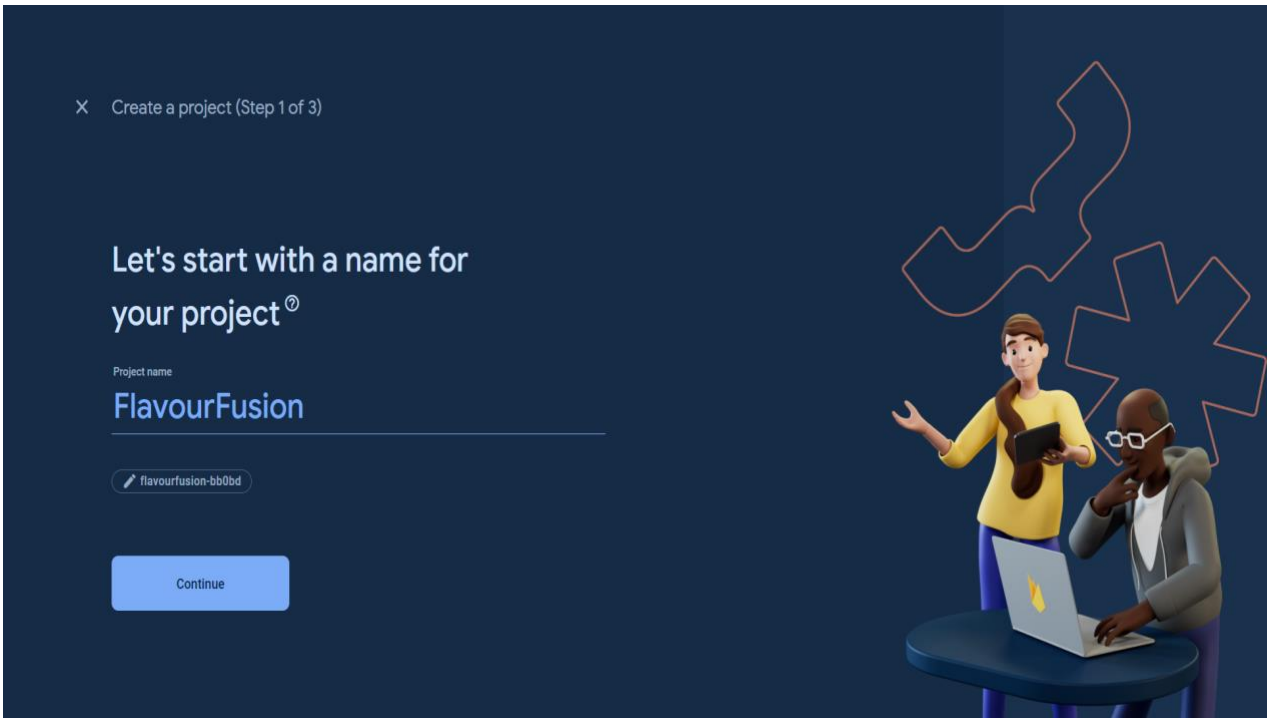


2. Select your existing Google Cloud project from the dropdown menu, then click Continue.
3. (Optional) Enable Google Analytics for your project, then follow the prompts to select or create a Google Analytics account.
4. Click Add Firebase.

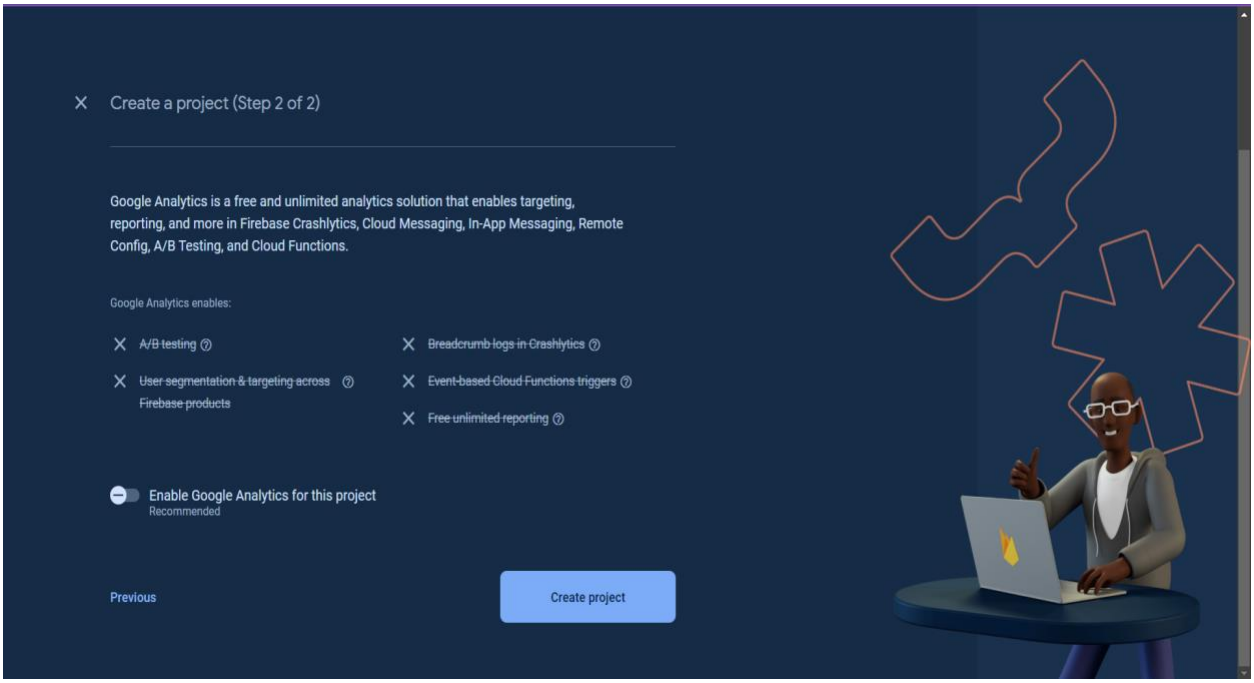


Enter the following information and click on Create Project:

1. Project name

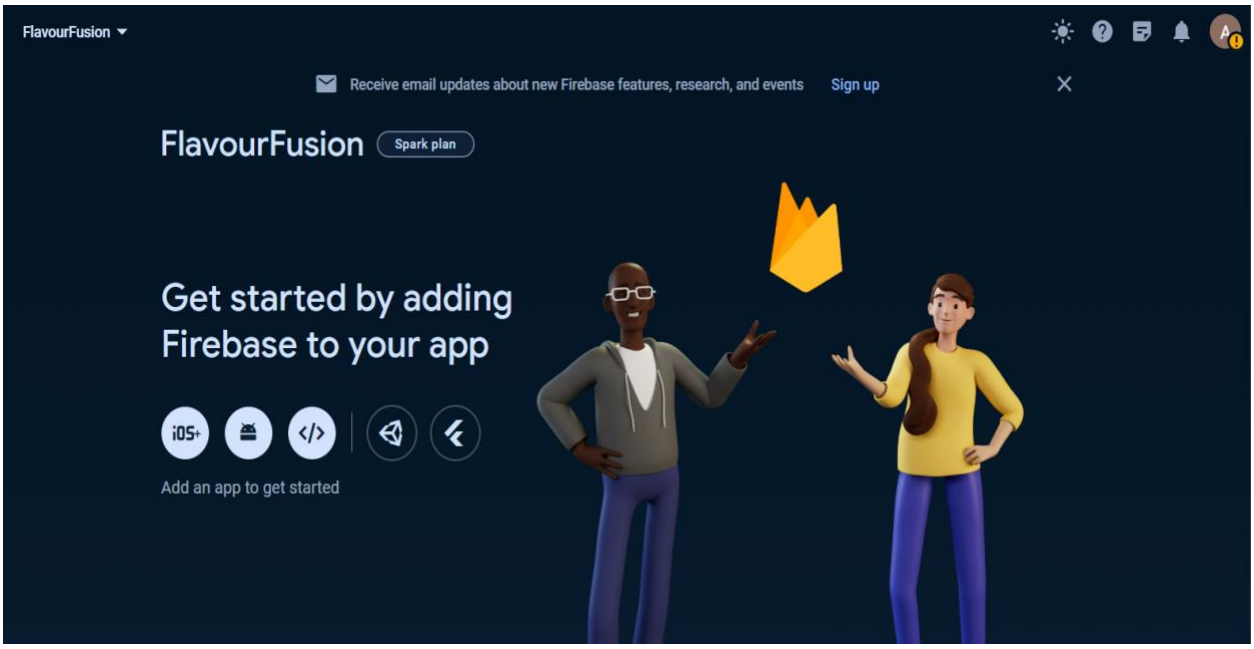


2. Disable the Google Analytics for the project, we do not need this now unless we deploy the app.

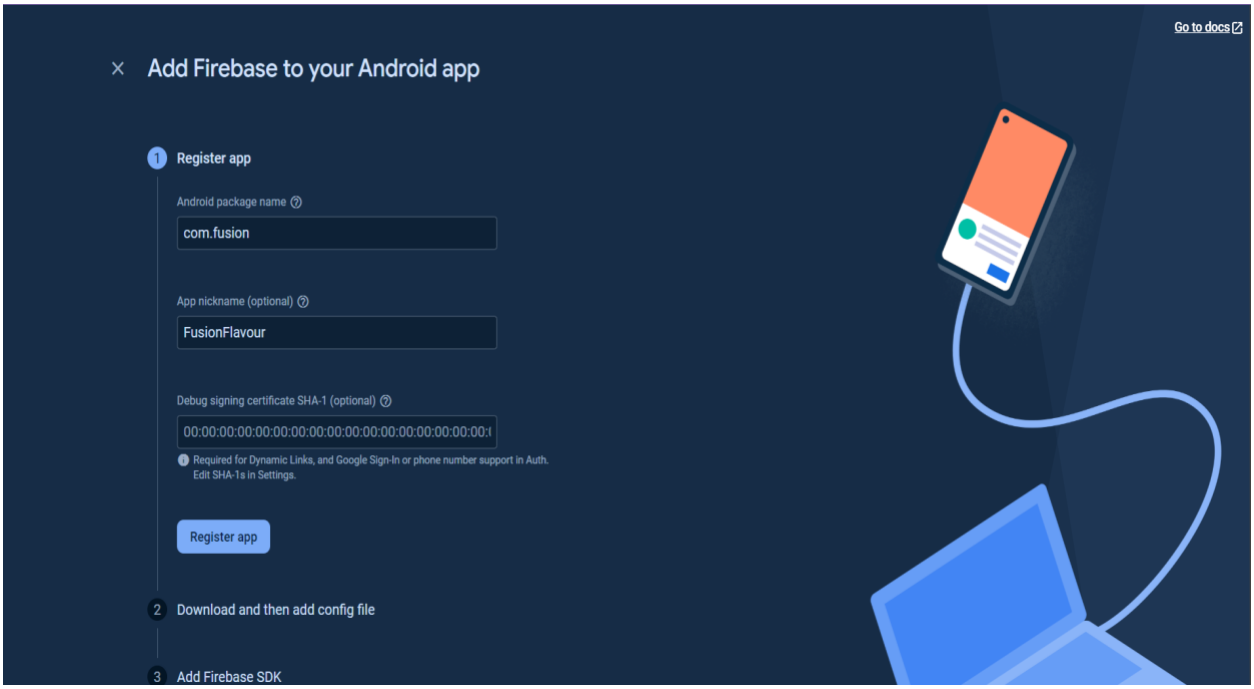


Setting up the Android App/IOS App

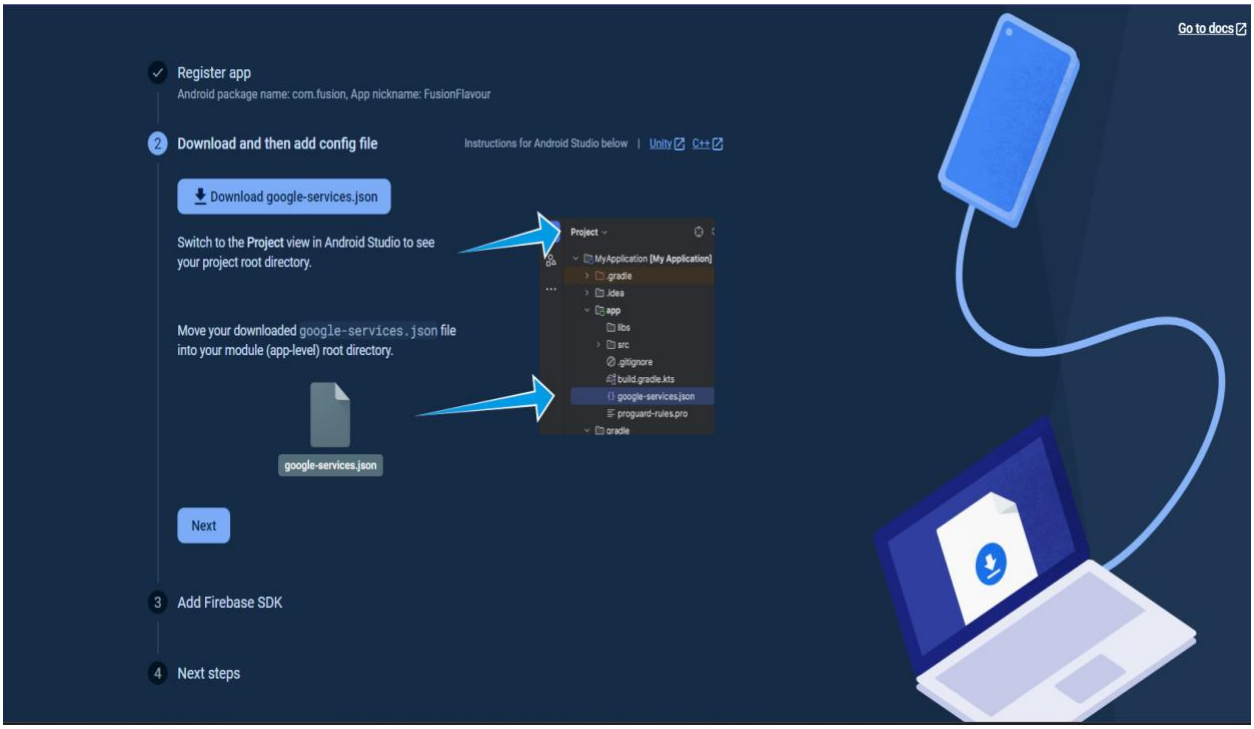
Now add an Android app if you are running your app on Android or else add an IOS app. I will be adding the Android app.



1. Register your Android App using the package name in the app-level build.gradle file and a random app nickname.



2. Download the google-services.json file and store it in the android/app directory.

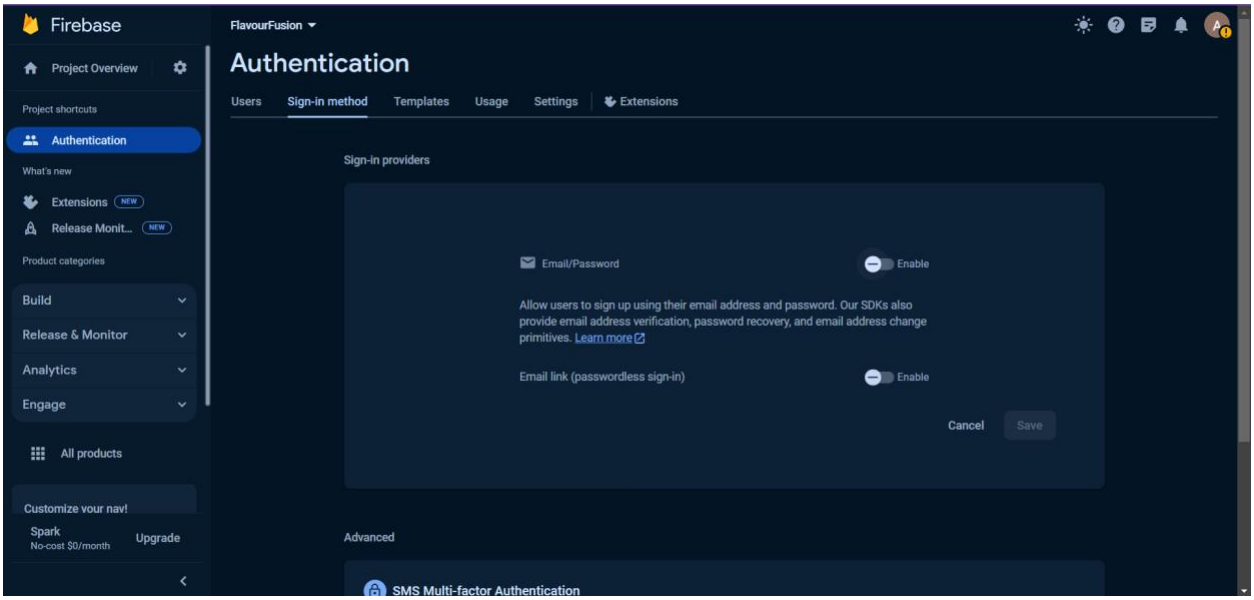


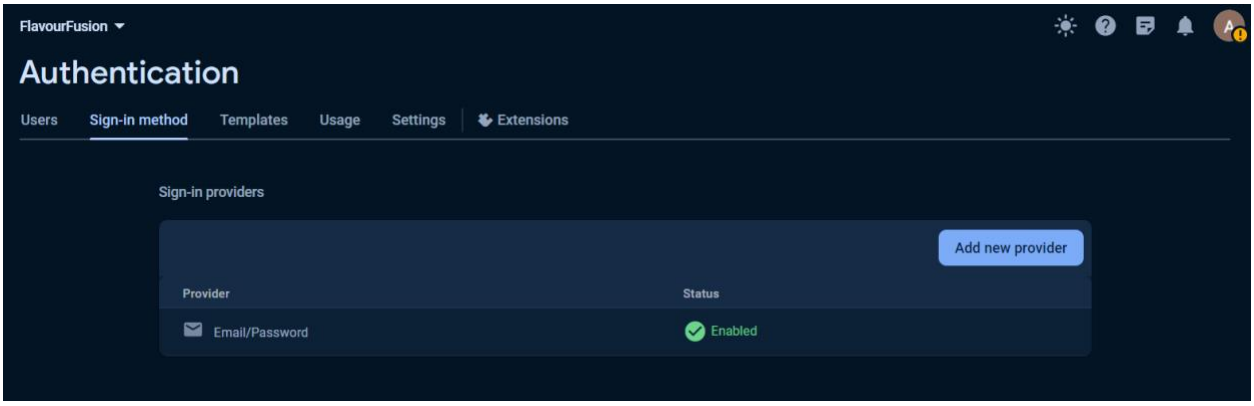
3. Add Firebase SDK to the project

Installing the FlutterFire libraries Add the following packages from pub.dev to your pubspec.yaml file.

```yaml
dependencies:
  flutter:
    sdk: flutter
  firebase_auth: ^4.3.0
  cupertino_icons: ^1.0.2
  flutter_svg: ^2.0.6
  form_field_validator: ^1.1.0
  flutter_tawk: ^0.1.0
```

Making a simple Realtime Database call For Login and Sign Up, I have used Email/Password provider in Authentication of the Firebase project.





For user authentication, I have made a separate auth_methods.dart file where I have added the functions to get user details, login and register.

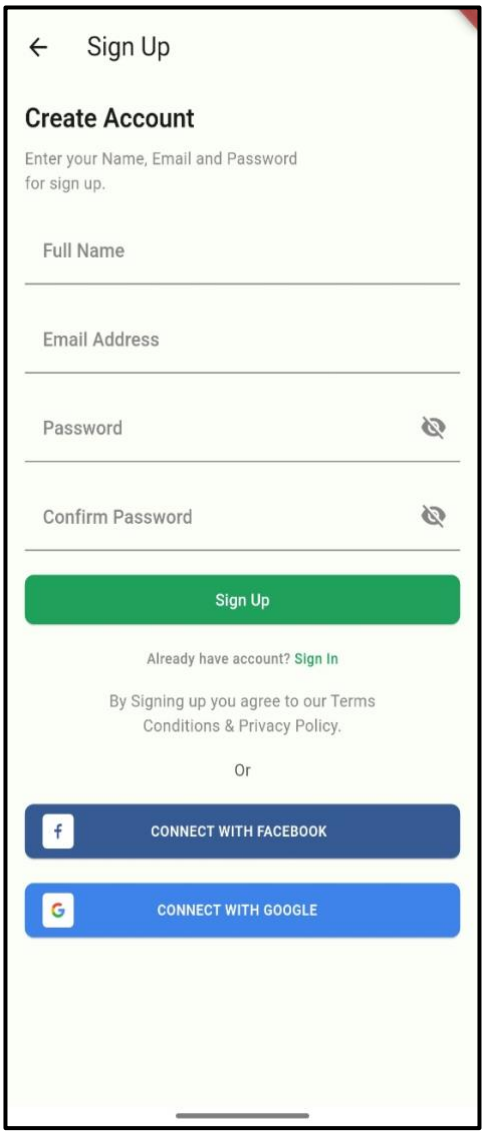
```dart
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:foodly_ui/screens/findRestaurants/find_restaurants_screen.dart';
import 'package:foodly_ui/screens/phoneLogin/phone_login_screen.dart';
import '../../../constants.dart';

class SignUpForm extends StatefulWidget {
  const SignUpForm({Key? key});
```

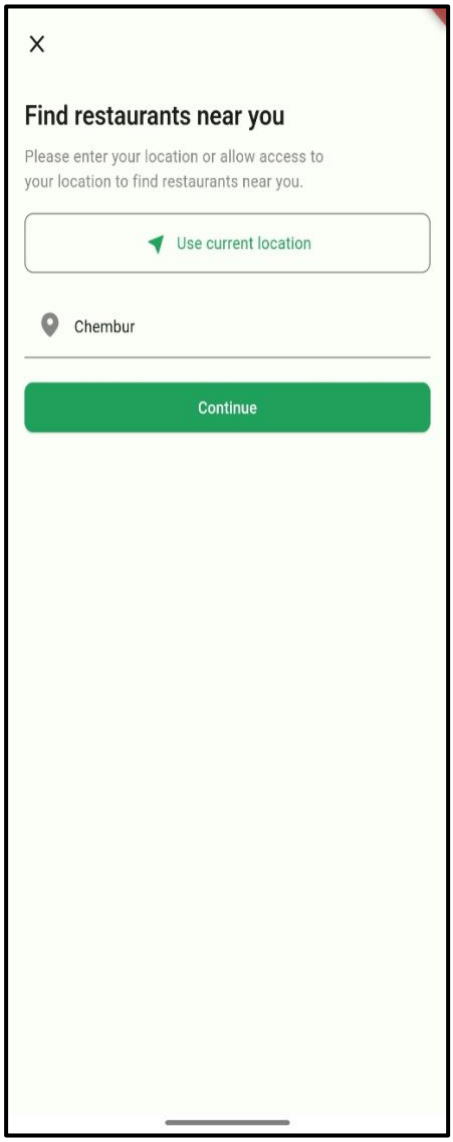
```
  @override
  _SignUpFormState createState() => _SignUpFormState();
}

class _SignUpFormState extends State<SignUpForm> {
  final _formKey = GlobalKey<FormState>();
  final FirebaseAuth _auth = FirebaseAuth.instance;

  TextEditingController _emailController = TextEditingController();
  TextEditingController _passwordController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Form(
      key: _formKey,
      child: Column(
        children: [
          TextFormField(
            controller: _emailController,
            validator: (value) {
              if (value!.isEmpty) {
                return 'Please enter your email';
              }
              return null;
            },
            decoration: const InputDecoration(hintText: "Email Address"),
          ),
          const SizedBox(height: defaultPadding),
          TextFormField(
            controller: _passwordController,
            obscureText: true,
            validator: (value) {
              if (value!.isEmpty) {
                return 'Please enter your password';
              }
              return null;
            },
            decoration: const InputDecoration(hintText: "Password"),
          ),
          const SizedBox(height: defaultPadding),
          ElevatedButton(
            onPressed: () async {
              if (_formKey.currentState!.validate()) {
                try {
                  await _auth.createUserWithEmailAndPassword(
                    email: _emailController.text.trim(),
```

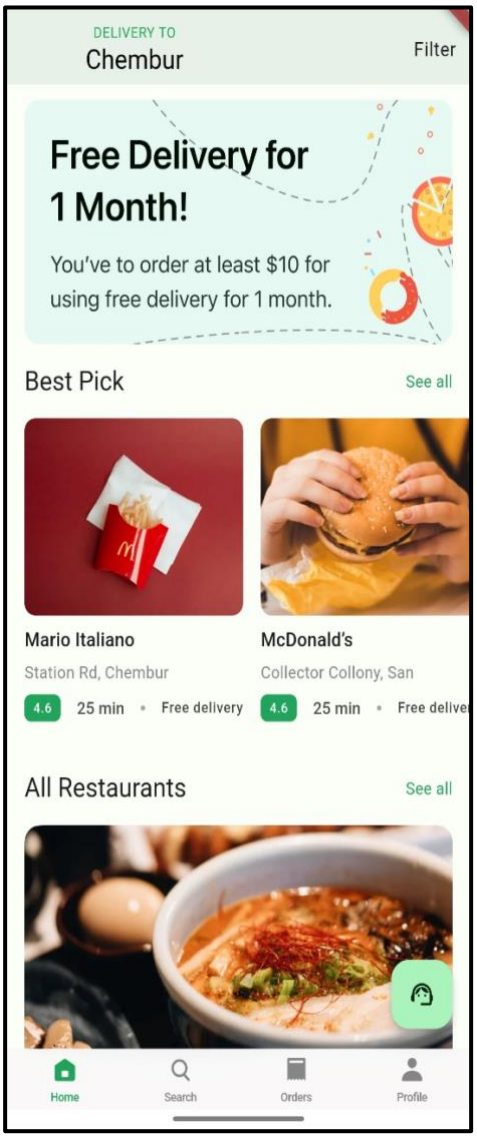
```
                  password: _passwordController.text.trim(),
                );
                Navigator.pushReplacement(
                  context,
                  MaterialPageRoute(builder: (_) => const PhoneLoginScreen()),
                );
              } catch (e) {
                print('Error during sign-up: $e');
                // Handle sign-up errors here
              }
            }
          },
          child: const Text("Sign Up"),
        ),

class SignInForm extends StatefulWidget {
  const SignInForm({Key? key});

  @override
  State<SignInForm> createState() => _SignInFormState();
}

class _SignInFormState extends State<SignInForm> {
  final _formKey = GlobalKey<FormState>();
  final FirebaseAuth _auth = FirebaseAuth.instance;

  TextEditingController _emailController = TextEditingController();
  TextEditingController _passwordController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Form(
      key: _formKey,
      child: Column(
        children: [
          TextFormField(
            controller: _emailController,
            validator: (value) {
              if (value!.isEmpty) {
                return 'Please enter your email';
              }
              return null;
            },
            decoration: const InputDecoration(hintText: "Email Address"),
          ),
          const SizedBox(height: defaultPadding),
```

```
          TextFormField(
            controller: _passwordController,
            obscureText: true,
            validator: (value) {
              if (value!.isEmpty) {
                return 'Please enter your password';
              }
              return null;
            },
            decoration: const InputDecoration(hintText: "Password"),
          ),
          const SizedBox(height: defaultPadding),
          ElevatedButton(
            onPressed: () async {
              if (_formKey.currentState!.validate()) {
                try {
                  await _auth.signInWithEmailAndPassword(
                    email: _emailController.text.trim(),
                    password: _passwordController.text.trim(),
                  );
                  Navigator.pushReplacement(
                    context,
                    MaterialPageRoute(
                        builder: (_) => const FindRestaurantsScreen()),
                  );
                } catch (e) {
                  print('Error during sign-in: $e');
                  // Handle sign-in errors here
                }
              }
            },
            child: const Text("Sign in"),
          ),
        ],
      ),
    );
  }
}
```

Now when we login to the app, the screen goes to Home Screen from SignUp Screen.







III. Conclusion:

Hence, we understood how to connect Flutter UI with Firebase database. Integrating Flutter UI with Firebase database enables real-time data storage and retrieval, enhancing app functionality. Firebase authentication offers secure login options for iOS and Android, ensuring user data protection. Challenges such as **maintaining state consistency** across screens can be overcome with state management solutions like **Provider or Riverpod**. Overall, combining Flutter with Firebase provides a powerful platform for building secure and feature-rich cross-platform applications.