

## EXPERIMENT NO:- 6

**Name:- Riya Varyani**

## D15A

**Roll-no:-61**

**Aim:- To Connect flutter UI with firebase database**

## Creating a New Firebase Project


×

Create a project

Let's start with a name for your project <sup>?</sup>

Project name

flutter-eats

 flutter-eats-ac4ee

Already have a Google Cloud project?  
[Add Firebase to Google Cloud project](#)

Continue

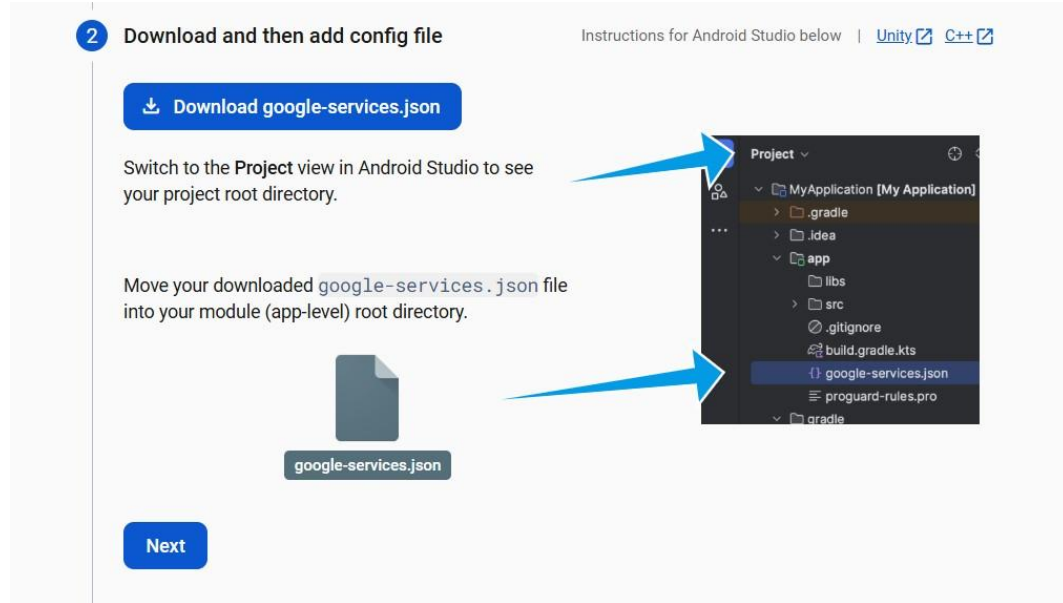
First, log in with your Google account to manage your Firebase projects. From within the Firebase dashboard, select the Create new project button and give it a name

In order to add Android support to our Flutter application, select the Android logo from the dashboard. This brings us to the following screen:

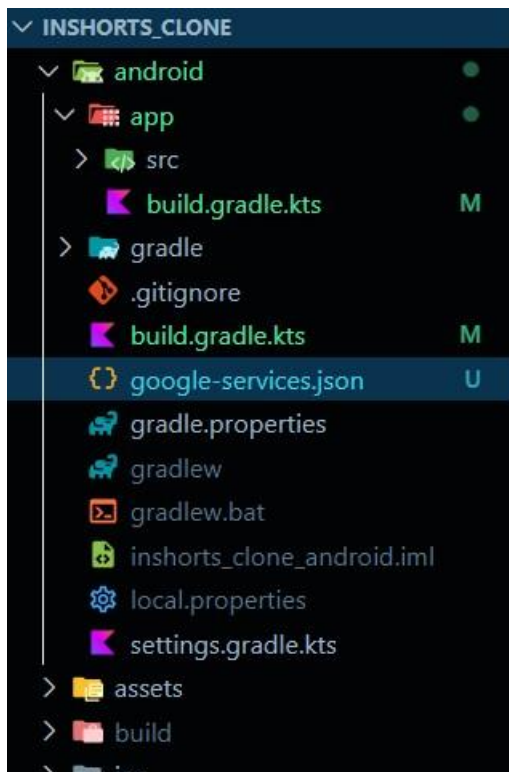
[illegible]

The most important thing here is to match up the Android package name that you choose here with the one inside of our application.

Then download the google-services.json file, that you will get.



put that file in the android folder (root level)



then select the build.gradle.kts (Kotlin DSL) part, and then follow the rest instructions

### 3 Add Firebase SDK

Instructions for Gradle | [Unity](#) [C++](#)

★ Are you still using the `buildscript` syntax to manage plugins? Learn how to [add Firebase plugins](#) using that syntax.

1. To make the `google-services.json` config values accessible to Firebase SDKs, you need the Google services Gradle plugin.

☒ Kotlin DSL (`build.gradle.kts`) ☐ Groovy (`build.gradle`)

Add the plugin as a dependency to your **project-level** `build.gradle.kts` file:

**Root-level (project-level) Gradle file** (`<project>/build.gradle.kts`):

```
plugins {  
    // ...  
  
    // Add the dependency for the Google services Gradle plugin  
    id("com.google.gms.google-services") version "4.4.2" apply false  
}
```

2. Then, in your **module (app-level)** `build.gradle.kts` file, add both the `google-services` plugin and any Firebase SDKs that you want to use in your app:

**Module (app-level) Gradle file** (`<project>/<app-module>/build.gradle.kts`):

```
plugins {  
    id("com.android.application")  
    // Add the Google services Gradle plugin  
    id("com.google.gms.google-services")  
    ...  
}  
  
dependencies {  
    // Import the Firebase BoM  
    implementation(platform("com.google.firebase:firebase-bom:33.9.0"))  
  
    // TODO: Add the dependencies for Firebase products you want to use  
    // When using the BoM, don't specify versions in Firebase dependencies  
    // https://firebase.google.com/docs/android/setup#available-libraries  
}
```

By using the Firebase Android BoM, your app will always use compatible Firebase library versions. [Learn more](#)

### 4 Next steps

You're all set!

Make sure to check out the [documentation](#) to learn how to get started with each Firebase product that you want to use in your app.

You can also explore [sample Firebase apps](#).

Or, continue to the console to explore Firebase.

[Previous](#)

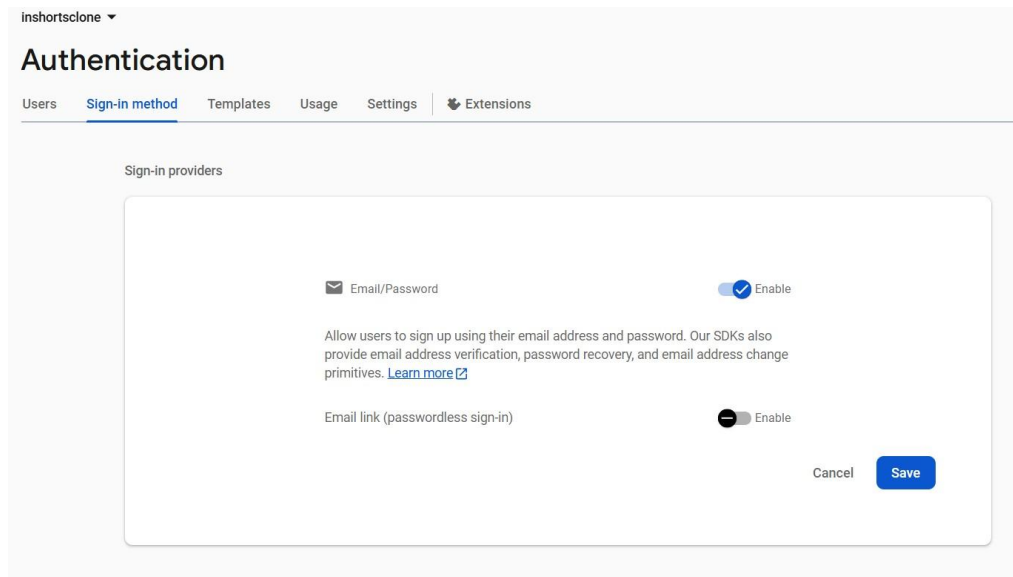
[Continue to console](#)

Generate the `firebase_options.dart` file, based on the `google-services.json` file

```
import 'package:firebase_core/firebase_core.dart';

class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    return const FirebaseOptions(
      apiKey: "AIzaSyA_VIR7fj4d-b6tNZhw9qJ6GRRx5EXKqs0",
      appId: "1:388272292768:android:33180b2382688b18781ac5",
      messagingSenderId: "388272292768",
      projectId: "inshortsclone-848a9",
      storageBucket: "inshortsclone-848a9.firebaseio.com",
      androidClientId: "1:388272292768:android:33180b2382688b18781ac5",
    );
  }
}
```

In your part select the sign-in method and enable it.



### Code : auth\_gate.dart

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter_eats/pages/home_page.dart';
import 'package:flutter_eats/services/auth/login_or_register.dart';
```

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

```
@override
```

```
Widget build(BuildContext context) {
  return Scaffold(
    body: StreamBuilder(
```

```

        stream: FirebaseAuth.instance.authStateChanges(),
        builder: (context, snapshot) {
          // user is logged in
          if(snapshot.hasData) {
            return const HomePage();
          }

          // user is Not logged in
          else {
            return const LoginOrRegister();
          }
        },
      ),
    );
  }
}

```

#### **code : auth\_service.dart**

```

import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_auth/firebase_auth.dart';

class AuthService {
  // Get instance of Firebase Auth
  final FirebaseAuth _firebaseAuth = FirebaseAuth.instance;

  // Get current user
  User? getCurrentUser() {
    return _firebaseAuth.currentUser;
  }

  // Sign in
  Future<UserCredential?> signInWithEmailPassword(String email, String
password) async {
    // try sign user up
    try {
      // Sign in user
      UserCredential userCredential =
        await _firebaseAuth.signInWithEmailAndPassword(email: email,
password: password);
      return userCredential;
    }

    //catch any error

```

```

    on FirebaseAuthException catch (e) {
      throw Exception(e.code);
    }
  }

// Sign up (Placeholder for future functionality)
Future<UserCredential?> signUpWithEmailPassword(String email, String
password) async {
  try {
    // Sign in user
    UserCredential userCredential =
    await _firebaseAuth.createUserWithEmailAndPassword(
    email: email,
    password: password);
    return userCredential;
  }

  //catch any error
  on FirebaseAuthException catch (e) {
    throw Exception(e.code);
  }
}

// Sign out (Placeholder for future functionality)

Future<void> signOut() async {
  return await _firebaseAuth.signOut();
}
}

```

### **code : login\_or\_register.dart**

```

import 'package:flutter/material.dart';
import 'package:flutter_eats/pages/login_page.dart';
import 'package:flutter_eats/pages/register_page.dart';

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

  @override
  State<LoginOrRegister> createState() => _LoginOrRegisterSate();
}

class _LoginOrRegisterSate extends State<LoginOrRegister> {

```

```
//initially, shows login page
bool showLoginPage = true;

//toggle between login and register page

void togglePages() {
  setState(() {
    showLoginPage = !showLoginPage;
  });
}

@override
Widget build(BuildContext context) {
  if(showLoginPage) {
    return LoginPage(onTap: togglePages);
  } else {
    return RegisterPage(onTap: togglePages);
  }
}
```

Search by email address, phone number, or user UID					Add user	↺	⋮
Identifier	Providers	Created ↓	Signed In	User UID			
varyaniriya2@gmail.com	✉	Feb 25, 2025	Feb 25, 2025	ZPV1014ZqscQ4KDw8xVqeB...			
neha@gmail.com	✉	Feb 22, 2025	Feb 22, 2025	kFDGI4qZVbNLCRJCTh7VwA...	📄		
sonam22@gmail.com	✉	Feb 22, 2025	Feb 22, 2025	VrETqvQhBeMlvVcevfCGuLce...			
isha@gmail.com	✉	Feb 22, 2025	Feb 22, 2025	DzoptkzEX6gnsG44kLxpOXdP...			
riya@gmail.com	✉	Feb 22, 2025	Feb 22, 2025	mlz0cLipnoNaB7jamJ2o8GT9...			
Rows per page: 50					1 – 5 of 5	⏪	⏩

**Conclusion** : In this experiment, we successfully connected a Flutter UI with Firebase for authentication. We configured Firebase, integrated authentication services, and implemented user sign-in, sign-up, and sign-out functionalities. The AuthGate managed user state, while AuthService handled authentication operations. This project provides a foundation for secure user authentication in Flutter apps and can be extended with more Firebase features.