

## EXPERIMENT NO:- 6

**Name:-Bhagyesh Patil**

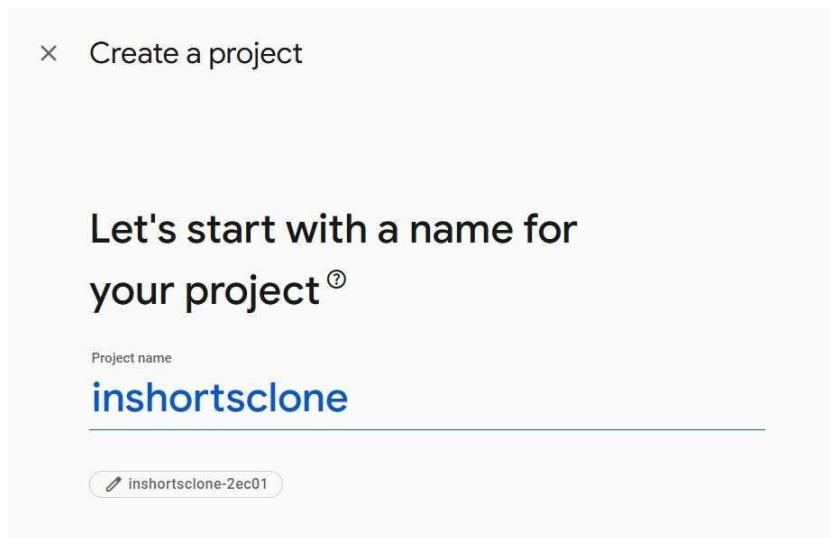
**D15A**

**Roll-no:-35**

Aim:- To Connect flutter UI with firebase database

---

### **Creating a New Firebase Project**



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.

**2 Download and then add config file** Instructions for Android Studio below | [Unity](#) [C++](#)

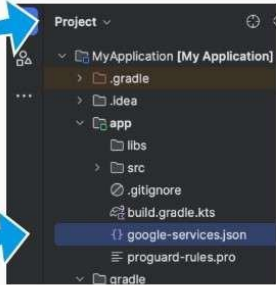
[Download google-services.json](#)

Switch to the **Project** view in Android Studio to see your project root directory.

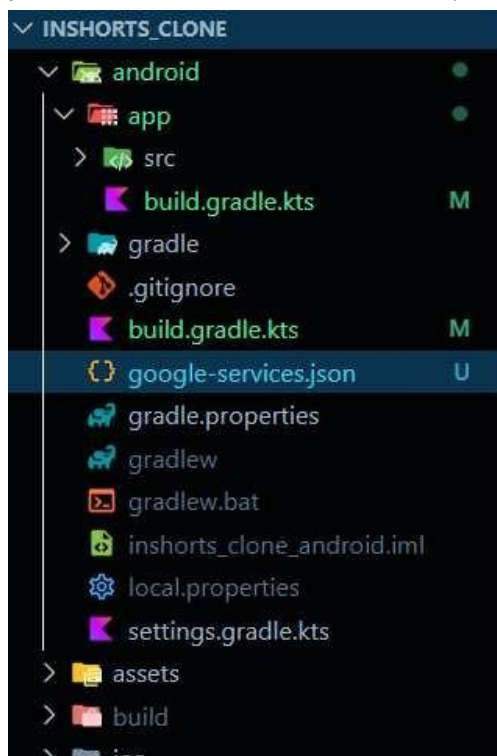
Move your downloaded `google-services.json` file into your module (app-level) root directory.

`google-services.json`

[Next](#)



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.

inshortsclone ▾

## Authentication


Users **Sign-in method** Templates Usage Settings Extensions

Sign-in providers

 Email/Password
 

☒ Enable

Allow users to sign up using their email address and password. Our SDKs also provide email address verification, password recovery, and email address change primitives. [Learn more](#)

 Email link (passwordless sign-in)
 

☐ Enable

## Code:-

### Authenction\_logic

```
import
'package:firebase_auth/firebase_auth.dart'
;
import
'package:google_sign_in/google_sign_in.d
art';
```

```
class AuthService {
  final FirebaseAuth _auth =
  FirebaseAuth.instance;

  // Sign in with Email & Password
  Future<User?> signInWithEmail(String
  email, String password) async {
    try {
      UserCredential userCredential = await
      _auth.signInWithEmailAndPassword(
        email: email,
        password: password,
      );
      return userCredential.user;
    } catch (e) {
      print("Error: $e");
      return null;
    }
  }
}
```

```
// Register with Email & Password
Future<User?> registerWithEmail(String
email, String password) async {
  try {
    UserCredential userCredential = await
    _auth.createUserWithEmailAndPassword(
      email: email,
      password: password,
    );
    return userCredential.user;
  } catch (e) {
```

```
    return null;
  }
}
```

```
// Google Sign-In
Future<User?> signInWithGoogle()
async {
  try {
    final GoogleSignInAccount?
    googleUser = await
    GoogleSignIn().signIn();
    if (googleUser == null) return null;
```

```
    final GoogleSignInAuthentication
    googleAuth = await
    googleUser.authentication;
    final AuthCredential credential =
    GoogleAuthProvider.credential(
      accessToken:
    googleAuth.accessToken,
      idToken: googleAuth.idToken,
    );
```

```
    UserCredential userCredential = await
    _auth.signInWithCredential(credential);
    return userCredential.user;
  } catch (e) {
    print("Google Sign-In Error: $e");
    return null;
  }
}
```

```
// Sign Out
Future<void> signOut() async {
  await _auth.signOut();
  await GoogleSignIn().signOut();
}
```

// Get current user

```
User? getCurrentUser() {  
    return _auth.currentUser;  
}
```

**Login screen**

```

import 'package:flutter/material.dart';
import '../services/auth_service.dart';
import 'home_screen.dart';

class LoginScreen extends StatefulWidget {
  @override
  _LoginScreenState createState() =>
  _LoginScreenState();
}

class _LoginScreenState extends
State<LoginScreen> {
  final TextEditingController emailController =
TextEditingController();
  final TextEditingController passwordController
= TextEditingController();
  final AuthService _authService =
AuthService();

  void _login() async {
    String email = emailController.text.trim();
    String password =
passwordController.text.trim();
    var user = await
_authService.signInWithEmail(email,
password);
    if (user != null) {
      Navigator.pushReplacement(context,
MaterialPageRoute(builder: (context) =>
HomeScreen()));
    } else {

ScaffoldMessenger.of(context).showSnackBar(
SnackBar(content: Text("Login failed!")));
    }

  void _googleSignIn() async {
    var user = await
_authService.signInWithGoogle();
    if (user != null) {
      Navigator.pushReplacement(context,
MaterialPageRoute(builder: (context) =>
HomeScreen()));
    } else {

ScaffoldMessenger.of(context).showSnackBar(
SnackBar(content: Text("Google Sign-In
failed!")));
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Rapido Login')),
      body: Padding(
        padding: EdgeInsets.all(16.0),

```

```

child: Column(

      mainAxisAlignment:
MainAxisAlignment.center,
      children: [
        TextField(controller: emailController,
decoration: InputDecoration(labelText:
'Email')),
        TextField(controller:
passwordController, decoration:
InputDecoration(labelText: 'Password'),
obscureText: true),
        SizedBox(height: 20),
        ElevatedButton(onPressed: _login,
child: Text('Login')),
        SizedBox(height: 10),
        ElevatedButton(onPressed:
_googleSignIn, child: Text('Sign in with
Google')),
      ],
    ),
  ),
);
}
}

```

## Home\_Screen

```

import 'package:flutter/material.dart';
import '../services/auth_service.dart';
import 'login_screen.dart';

class HomeScreen extends StatelessWidget {
  final AuthService _authService =
AuthService();

  void _logout(BuildContext context) async {
    await _authService.signOut();
    Navigator.pushReplacement(context,
MaterialPageRoute(builder: (context) =>
LoginScreen()));
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Rapido

```



```
Home')),  
  body: Center(  
    child: Column(  
      mainAxisAlignment:  
MainAxisAlignment.center,  
      children: [  
        Text('Welcome to Rapido!'),  
        SizedBox(height: 20),  
        ElevatedButton(onPressed: () =>  
_logout(context), child: Text('Logout')),  
      ],  
    ),  
  ),
```

OUTPUT :

