**MAD and PWA Lab**

**Experiment - 6**

**Name: Abhishek Fatate Class: D15A Roll no:15**

**Aim: To Connect Flutter UI with FireBase database**

**Theory:**

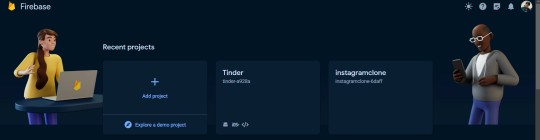
**Prerequisites**

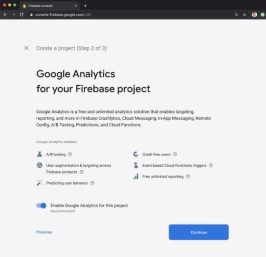
To complete this tutorial, you will need:

* A Google account to use Firebase.
* Developing for iOS will require XCode.
* To download and install Flutter.
* To download and install Android Studio and Visual Studio Code.
* It is recommended to install plugins for your code editor:
  + Flutter and Dart plugins installed for Android Studio.
  + Flutter extension installed for Visual Studio Code.

# Create a 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:





# Go to the Firebase Console and create a new project. Add your Flutter app to the Firebase project:

Register your app in the Firebase project, and follow the instructions to download the configuration files (google-services.json for Android, GoogleService-Info.plist for iOS).

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

The structure consists of at least two segments. A common pattern is to use a domain name, a company name, and the application name:

com.example.flutterfirebaseexample

Once you’ve decided on a name, open android/app/build.gradle in your code editor and update the applicationId to match the Android package name:

android/app/build.gradle

...

defaultConfig {

// TODO: Specify your own unique Application ID (https://developer.android.com/studio/build/application-id.html). applicationId 'com.example.flutterfirebaseexample'

...

}

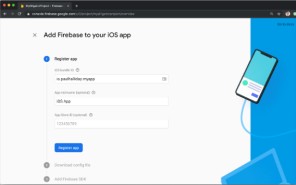
...

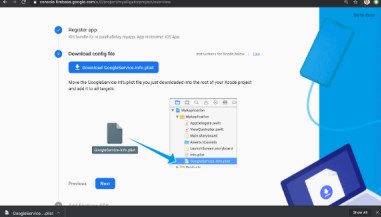
**Downloading the Conﬁg File**

The next step is to add the Firebase conﬁguration ﬁle into our Flutter project. This is important as it contains the API keys and other critical information for Firebase to use.

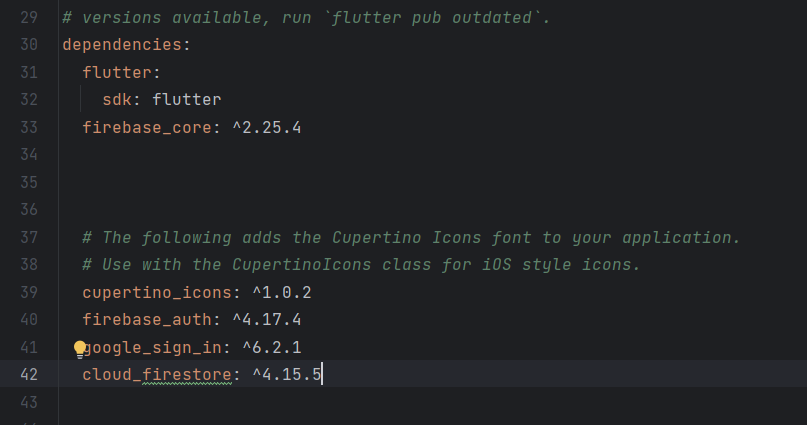
Select Download google-services.json from this page:







# 2. Add Firebase to your Flutter project: Add Dependencies:

Open your pubspec.yaml file and add the necessary dependencies: yaml

# Code:

Signup.dart

import 'package:flutter/material.dart';

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

import 'package:google\_sign\_in/google\_sign\_in.dart';

import 'homepage.dart'; // Import the HomePage widget

class LoginScreen extends StatelessWidget {

final FirebaseAuth \_auth = FirebaseAuth.instance;

final GoogleSignIn googleSignIn = GoogleSignIn();

Future<void> \_signInWithGoogle(BuildContext context) async {

try {

final GoogleSignInAccount? googleUser = await googleSignIn.signIn();

if (googleUser != null) {

final GoogleSignInAuthentication googleAuth = await googleUser.authentication;

final AuthCredential credential = GoogleAuthProvider.credential(

accessToken: googleAuth.accessToken,

idToken: googleAuth.idToken,

);

final UserCredential userCredential = await \_auth.signInWithCredential(credential);

// Navigate to the HomePage after successful sign-in

Navigator.pushReplacement(

context,

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

);

}

} catch (e) {

print('Error signing in with Google: $e');

// Handle error

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: const Text('Sign In'),

centerTitle: true,

),

body: Padding(

padding: const EdgeInsets.all(20.0),

child: Column(

crossAxisAlignment: CrossAxisAlignment.center,

children: [

// Your login form fields here

ElevatedButton(

onPressed: () => \_signInWithGoogle(context),

child: Text('Sign In with Google'),

),

],

),

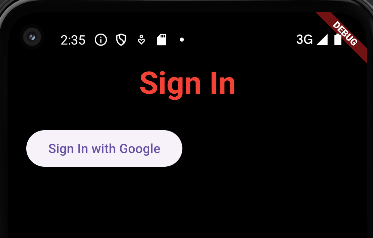
),

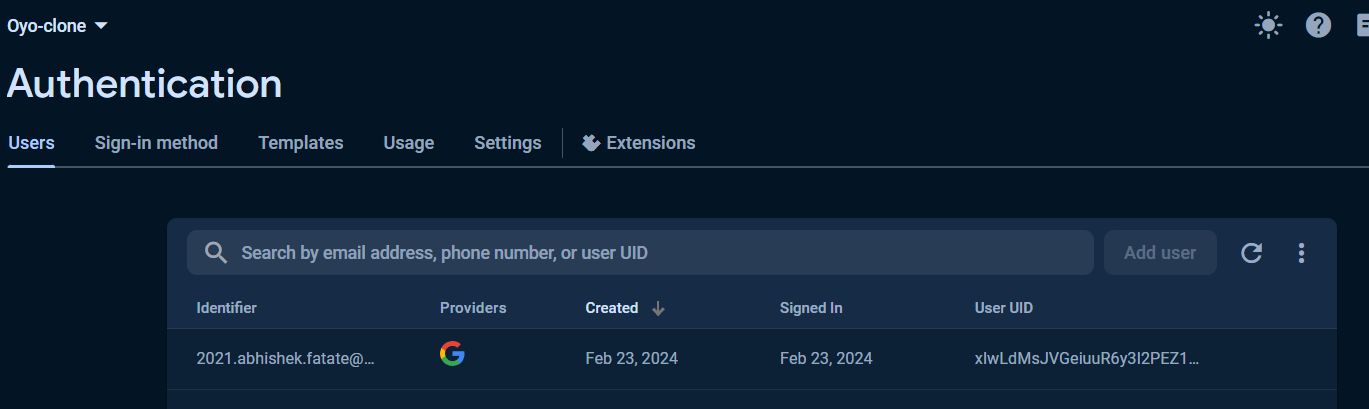
);

}

}

# Output:





**Conclusion**:

In this experiment, we have successfully connected firebase database and authenticated using google signin.