

EXPERIMENT NO:- 6

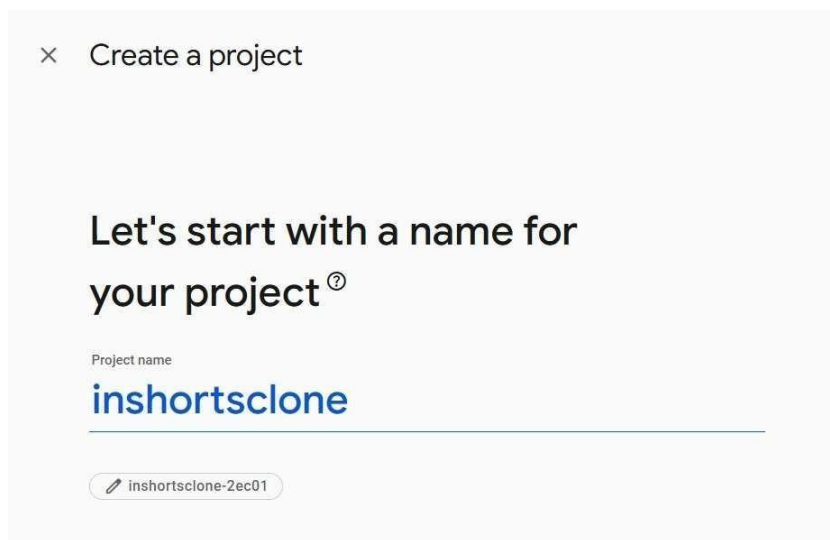
Name:-Soham Satpute

Class: D15A

Roll-no:-51

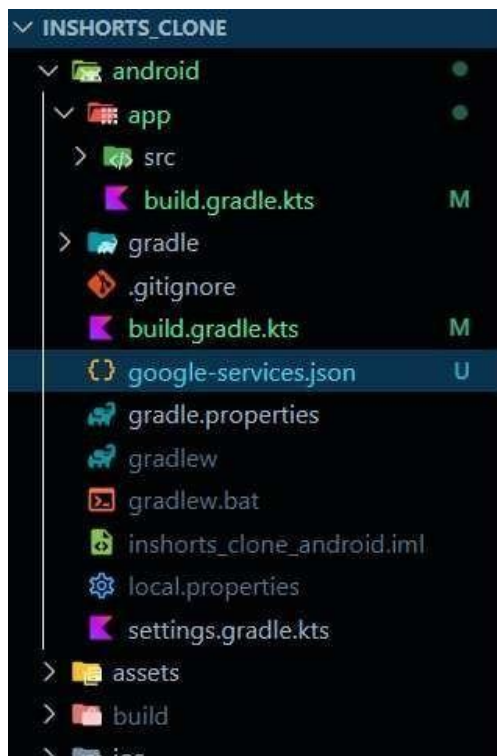
Aim:- To Connect flutter UI with firebase database

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:



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.

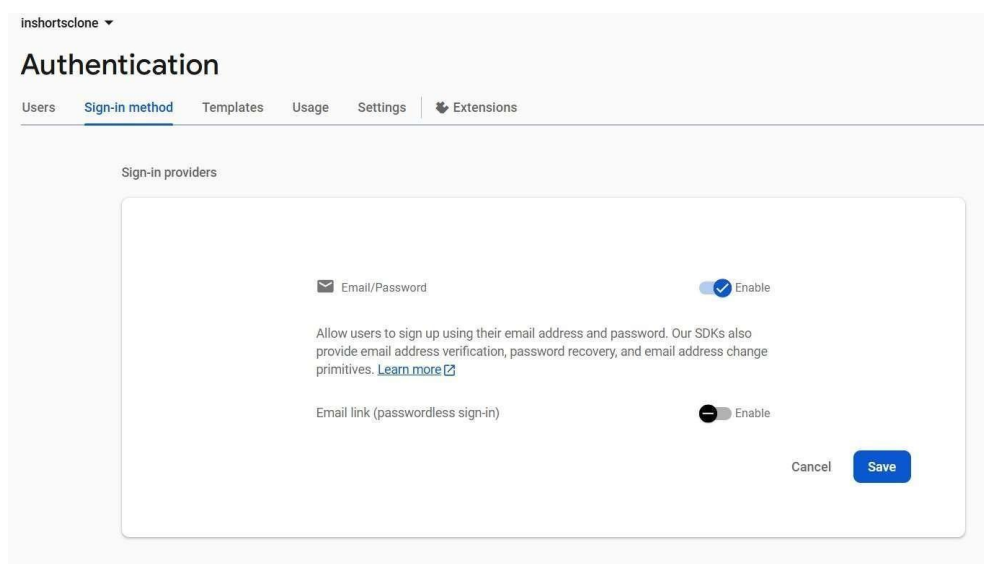
[Previous](#)

[Continue to console](#)

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

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  if (kIsWeb) { // Now kIsWeb is recognized
    await Firebase.initializeApp(
      options: FirebaseOptions(
        apiKey: "AIzaSyCJ5rQWWCA-ni6n1dN1bkmJunw0lgH-z_o",
        authDomain: "fir-flutter-20c74.firebaseio.com",
        projectId: "fir-flutter-20c74",
        storageBucket: "fir-flutter-20c74.appspot.com", // Fixed format
        messagingSenderId: "444971098036",
        appId: "1:444971098036:web:4c047110b5b7bfd34d1ce1",
        measurementId: "G-7R5LYFLVNV",
      ),
    );
  } else {
    await Firebase.initializeApp(); // Added missing semicolon
  }
}
```

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



Code:-

Auth_code:

```
import
'package:firebase_auth/
firebase_auth.dart';
import
'package:cloud_firestore
e/cloud_firestore.dart';

class AuthService {
  final FirebaseAuth
_auth =
FirebaseAuth.instance;
  final FirebaseFirestore
_firestore =
FirebaseFirestore.insta
nce;

  // Sign Up Method
  Future<String?>
signUp(String email,
String password) async
{
  try {
    UserCredential
userCredential = await
_auth.createUserWithE
mailAndPassword(
      email: email,
      password:
password,
    );

    // Save user details
to Firestore
```

```
      await
_firestore.collection('us
ers').doc(userCredential
.user!.uid).set({
      'email': email,
      'uid':
userCredential.user!.uid
,
      'createdAt':
FieldValue.serverTimes
tamp(),
    });

    return null; // No
error
  } on
FirebaseAuthException
catch (e) {
    return
_handleAuthError(e);
  } catch (e) {
    return 'An
unexpected error
occurred: $e';
  }
}

// Log In Method
  Future<String?>
signIn(String email,
String password) async
{
  try {
    await
_auth.signInWithEmailA
ndPassword(email:
email, password:
```

```

password);
    return null; // No
error
    } on
FirebaseAuthException
catch (e) {
    return
    _handleAuthError(e);
    }
    }

```

```

// Log Out Method
Future<void>
signOut() async {
    await
    _auth.signOut();
    }

```

```

// Get Current User
User?
getCurrentUser() {
    return
    _auth.currentUser;
    }

```

```

// Handle Firebase
Authentication Errors
String
_handleAuthError(Fireb
aseAuthException e) {
    switch (e.code) {
        case 'email-already-
in-use':
            return 'This email
is already registered!';
        case 'weak-
password':

```

```

        return 'Password
should be at least 6
characters!';
        case 'user-not-
found':
            return 'No user
found with this email!';
        case 'wrong-
password':
            return 'Incorrect
password!';
        case 'invalid-email':
            return 'Invalid
email format!';
        default:
            return 'An error
occurred:
${e.message}';
    }
    }
    }

```

Login screen :

```

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

class LoginPage extends StatefulWidget {
    @override
    _LoginPageState createState() =>
    _LoginPageState();
}

class _LoginPageState extends
State<LoginPage> {
    final _formKey =
GlobalKey<FormState>();
    final TextEditingController
_emailController = TextEditingController();
    final TextEditingController
_passwordController =

```

```

TextEditingController();
final AuthService _authService =
AuthService();

Future<void> _validateAndLogin() async
{
  if (_formKey.currentState!.validate()) {
    String? error = await
_authService.signIn(
      _emailController.text.trim(),
      _passwordController.text.trim(),
    );

    if (error == null) {

ScaffoldMessenger.of(context).showSnack
Bar(
  Snackbar(content: Text('Login
successful!')),
);

Navigator.pushReplacementNamed(conte
xt, '/home');
} else {

ScaffoldMessenger.of(context).showSnack
Bar(
  Snackbar(content: Text(error)),
);
}
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor:
Colors.blue.shade700,
    body: SafeArea(
      child: Column(
        children: [
          Spacer(),
          Text(
            'Welcome Back',
            style: TextStyle(fontSize: 26,
color: Colors.white, fontWeight:
FontWeight.bold),
          ),
          Spacer(),
          Container(
            width: double.infinity,

```

```

padding:
EdgeInsets.symmetric(horizontal: 20,
vertical: 30),
    decoration: BoxDecoration(
      color: Colors.white,
      borderRadius:
BorderRadius.only(topLeft:
Radius.circular(30), topRight:
Radius.circular(30)),
      boxShadow: [BoxShadow(color:
Colors.black12, spreadRadius: 1,
blurRadius: 10)],
    ),
    child: Form(
      key: _formKey,
      child: Column(
        mainAxisAlignment:
MainAxisSize.min,
        children: [
          _buildTextField("Email",
Icons.email, _emailController),
          SizedBox(height: 10),
          _buildTextField("Password",
Icons.lock, _passwordController,
isPassword: true),
          SizedBox(height: 20),
          _buildButton("Login",
_validateAndLogin),
        ],
      ),
    ),
  ),
],
),
),
),
),
),
),
);
}

Widget _buildTextField(String label,
IconData icon, TextEditingController
controller, {bool isPassword = false}) {
  return TextFormField(
    controller: controller,
    decoration: InputDecoration(
      labelText: label,
      prefixIcon: Icon(icon, color:
Colors.blue),
      border:
OutlineInputBorder(borderRadius:
BorderRadius.circular(10)),
      filled: true,

```



```

        fillColor: Colors.grey.shade100,
      ),
      obscureText: isPassword,
      validator: (value) => value == null ||
value.isEmpty ? 'This field cannot be
empty' : null,
    );
  }

```

```

Widget _buildButton(String text,
VoidCallback onPressed) {
  return SizedBox(
    width: double.infinity,
    child: ElevatedButton(
      onPressed: onPressed,
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.blue,
        padding:
EdgeInsets.symmetric(vertical: 15),
        shape:
RoundedRectangleBorder(borderRadius:
BorderRadius.circular(10)),
      ),
      child: Text(text, style:
TextStyle(fontSize: 18, color:
Colors.white)),
    ),
  );
}

```

Sign_up Page:

```

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

class SignUpPage extends
StatefulWidget {
  @override
  _SignUpPageState createState() =>
_SignUpPageState();
}

class _SignUpPageState extends
State<SignUpPage> {
  final _formKey =
GlobalKey<FormState>();
  final TextEditingController
_nameController =
TextEditingController();

```

```

final TextEditingController
_ageController = TextEditingController();
final TextEditingController
_emailController =
TextEditingController();
final TextEditingController
_passwordController =
TextEditingController();
final AuthService _authService =
AuthService();

Future<void> _validateAndSignUp()
async {
  if (_formKey.currentState!.validate()) {
    String? error = await
_authService.signUp(
      _emailController.text.trim(),
      _passwordController.text.trim(),
    );
  }
}

```

```

    if (error == null) {

ScaffoldMessenger.of(context).showSnac
kBar(
    Snackbar(content: Text('Sign up
successful!')),
    );

Navigator.pushReplacementNamed(cont
ext, '/home');
    } else {

ScaffoldMessenger.of(context).showSnac
kBar(
    Snackbar(content: Text(error)),
    );
    }
}

@override
Widget build(BuildContext context) {
    return Scaffold(
        backgroundColor:
Colors.blue.shade700,
        body: SafeArea(
            child: Column(
                children: [
                    Spacer(),
                    Text(
                        'Create Account',
                        style: TextStyle(fontSize: 26,
color: Colors.white, fontWeight:
FontWeight.bold),
                    ),
                    Spacer(),
                    Container(
                        width: double.infinity,
                        padding:
EdgeInsets.symmetric(horizontal: 20,
vertical: 30),
                        decoration: BoxDecoration(
                            color: Colors.white,
                            borderRadius:
BorderRadius.only(topLeft:
Radius.circular(30), topRight:
Radius.circular(30)),
                            boxShadow:
[BoxShadow(color: Colors.black12,

```

```

spreadRadius: 1, blurRadius: 10)],
                ),
                child: Form(
                    key: _formKey,
                    child: Column(
                        mainAxisAlignment:
MainAxisSize.min,
                        children: [
                            _buildTextField("Full Name",
Icons.person, _nameController),
                            SizedBox(height: 10),
                            _buildTextField("Age",
Icons.calendar_today, _ageController,
isNumber: true),
                            SizedBox(height: 10),
                            _buildTextField("Email",
Icons.email, _emailController),
                            SizedBox(height: 10),
                            _buildTextField("Password",
Icons.lock, _passwordController,
isPassword: true),
                            SizedBox(height: 20),
                            _buildButton("Sign Up",
_validateAndSignUp),
                        ],
                    ),
                ),
            ],
        ),
    );
}

Widget _buildTextField(String label,
IconData icon, TextEditingController
controller, {bool isPassword = false, bool
isNumber = false}) {
    return TextFormField(
        controller: controller,
        decoration: InputDecoration(
            labelText: label,
            prefixIcon: Icon(icon, color:
Colors.blue),
            border:
OutlineInputBorder(borderRadius:
BorderRadius.circular(10)),
            filled: true,
            fillColor: Colors.grey.shade100,

```

```

    ),
    keyboardType: isNumber ?
TextInputType.number :
TextInputType.text,
    obscureText: isPassword,
    validator: (value) => value == null ||
value.isEmpty ? 'This field cannot be
empty' : null,
  );
}

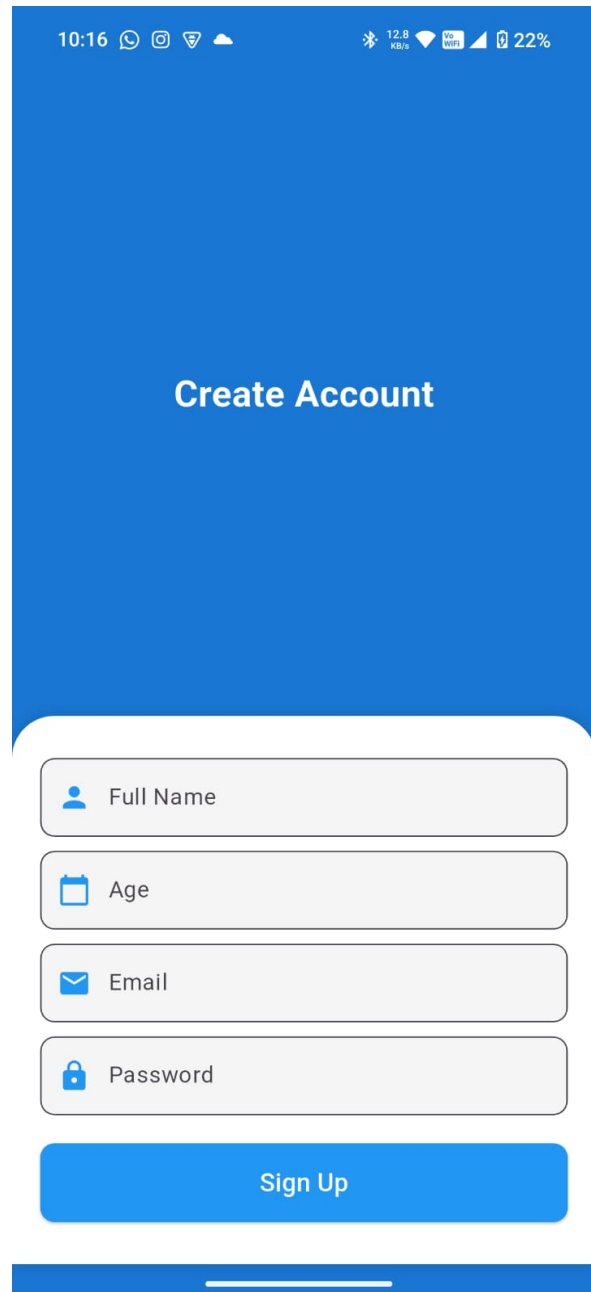
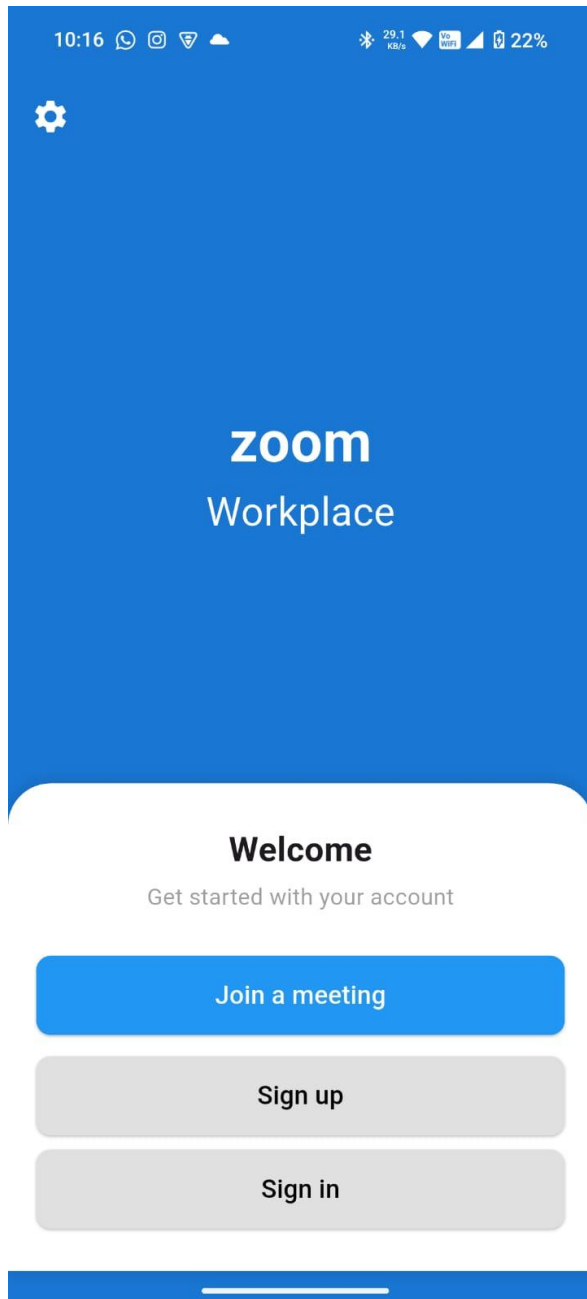
```

```

Widget _buildButton(String text,
VoidCallback onPressed) {
  return SizedBox(
    width: double.infinity,
    child: ElevatedButton(
      onPressed: onPressed,
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.blue,
        padding:
EdgeInsets.symmetric(vertical: 15),
        shape:
RoundedRectangleBorder(borderRadius:
BorderRadius.circular(10)),
      ),
      child: Text(text, style:
TextStyle(fontSize: 18, color:
Colors.white)),
    ),
  );
}
}

```

OUTPUT :



:

10:16



0.62 KB/s



22%

Welcome Back



Email




Password


Login

firebase-flutter ▾

Authentication


[Users](#) [Sign-in method](#) [Templates](#) [Usage](#) [Settings](#) | [Extensions](#)

 The following Authentication features will stop working when Firebase Dynamic Links shuts down on August 25, 2025: email link authentication for mobile apps, as well as Cordova OAuth support for web apps. ▾



 Search by email address, phone number, or user UID

Add user



| Identifier | Providers | Created ▾ | Signed In | User UID |
|------------------------|---|-------------|-------------|----------------------------|
| satputes203@gmail.com |  | Mar 7, 2025 | Mar 7, 2025 | HZRCub01F1WEP77sHnvOo8... |
| sohamsat619@gmail.c... |  | Mar 7, 2025 | Mar 7, 2025 | H6Dib4NP1CdGb6K6YLGaViW... |


Rows per page: 50 ▾ 1 - 2 of 2 < >

 > users > qrmTb7WkKoe2. 

 More in Google Cloud ▾

 (default)

 users

 qrmTb7WkKoe2ER0D3Pfq

+ Start collection

+ Add document

+ Start collection

users >

qrmTb7WkKoe2ER0D3Pfq >

+ Add field

createdAT: March 7, 2025 at 12:00:00 AM UTC+5:30

email: "satputes203@gmail.com"

uid: "satpute@1133"