Experiment no. 6
Name: Aryan Nandanwar
Div: D15B
Roll no: 55

Aim: To connect Flutter UI with Firebase.

Theory:
Firebase is a comprehensive platform provided by Google for developing mobile and web applications. It offers a wide range of services to help developers build high-quality apps more efficiently, including authentication, cloud storage, real-time databases, analytics, hosting, and more.

Here's a brief overview of some key Firebase services and how they are commonly used in Flutter projects:

1. Firebase Authentication: Firebase Authentication provides secure user authentication services, supporting various authentication methods such as email/password, phone number, Google Sign-In, Facebook Login, etc. It allows you to easily manage user authentication in your Flutter app, handling user sign-up, sign-in, and managing user sessions.

2. Cloud Firestore / Realtime Database: Firebase offers two database options: Cloud Firestore and the Realtime Database. Both are NoSQL databases that allow you to store and sync data in real-time between your Flutter app and the cloud. Cloud Firestore is a more scalable and powerful option, offering more advanced querying capabilities and a more flexible data model. The Realtime Database, on the other hand, provides a simpler JSON-based database structure and is
well-suited for real-time data synchronization.

3. Firebase Storage: Firebase Storage is a cloud-based storage solution that allows you to store and serve user-generated content such as images, videos, and other files. It provides secure and scalable storage, making it easy to integrate file storage into your Flutter app.

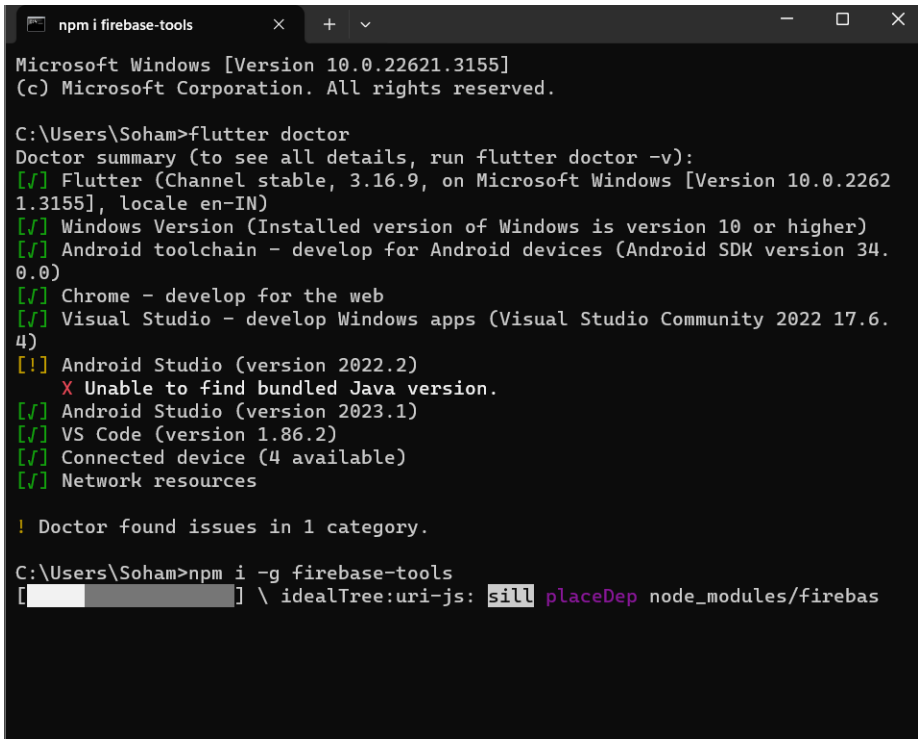
4. Firebase Cloud Messaging (FCM): Firebase Cloud Messaging is a cross-platform messaging solution that allows you to send notifications and messages to users across various platforms, including Android, iOS, and the web. It enables you to engage and retain users by sending targeted and personalized messages to your Flutter app users.

5. Firebase Analytics: Firebase Analytics provides powerful analytics tools to help you understand user behavior and app performance. It allows you to track user engagement, measure app performance metrics, and gain insights into how users interact with your Flutter app. This data can be used to optimize user experiences, improve app performance, and make data-driven decisions.

6. Firebase Hosting: Firebase Hosting is a static web hosting service that allows you to deploy and serve your Flutter web app quickly and securely. It provides a fast and reliable hosting solution with built-in SSL encryption, global CDN, and automatic scaling, making it easy to deploy and manage your Flutter web app.

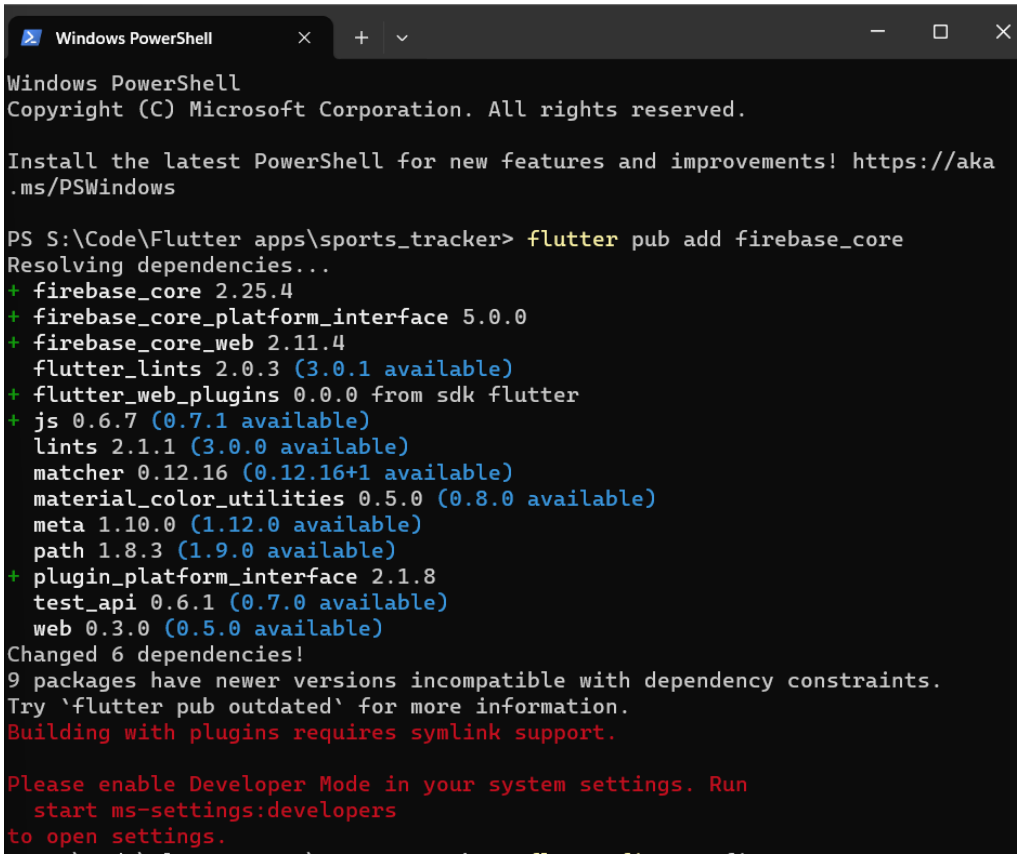
These are just some of the key Firebase services commonly used in Flutter projects. Firebase offers many more services and features that can help you build powerful and scalable apps with Flutter. It provides easy-to-use SDKs and comprehensive documentation, making it a popular choice for developers building mobile and web apps with Flutter.

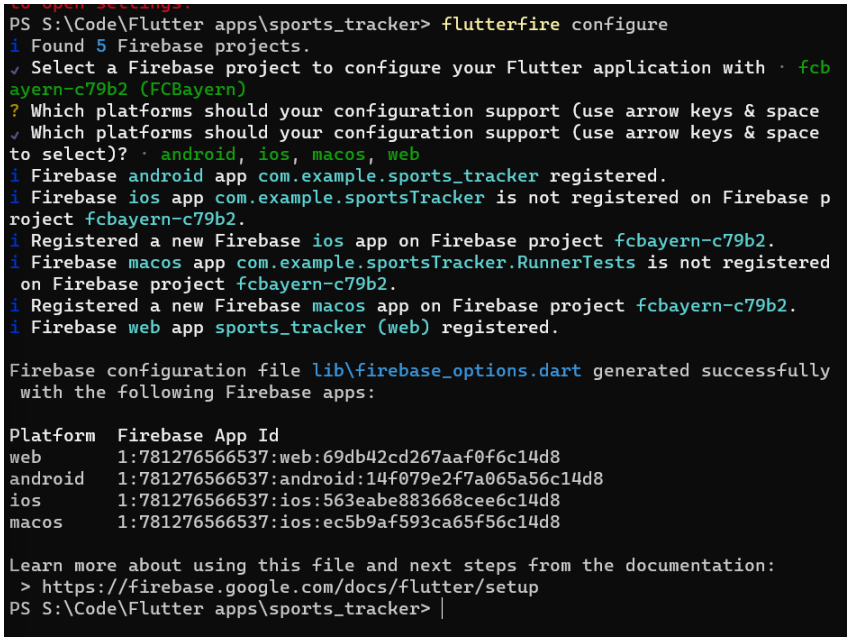
Implementing connection:



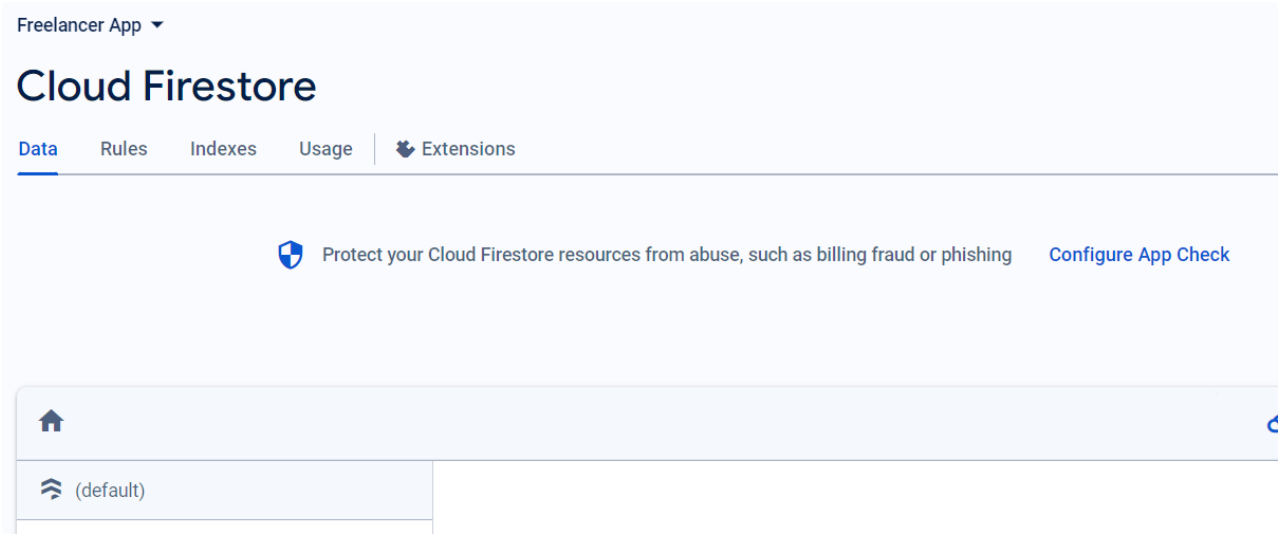
```
C:\Users\Soham>dart pub global activate flutterfire_cli
+ ansi_styles 0.3.2+1s... (2.1s)
+ args 2.4.2
+ async 2.11.0
+ boolean_selector 2.1.1
+ characters 1.3.0
+ ci 0.1.0
+ cli_util 0.3.5 (0.4.1 available)
+ clock 1.1.1
+ collection 1.18.0
+ dart_console 1.2.0
+ deep_pick 0.10.0 (1.0.0 available)
+ ffi 2.1.0 (2.1.2 available)
+ file 6.1.4 (7.0.0 available)
+ flutterfire_cli 0.2.7
+ http 0.13.6 (1.2.1 available)
+ http_parser 4.0.2
+ interact 2.2.0
+ intl 0.18.1 (0.19.0 available)
+ json_annotation 4.8.1
+ matcher 0.12.16+1
+ meta 1.12.0
+ path 1.9.0
+ petitparser 6.0.2
+ platform 3.1.4
+ process 4.2.4 (5.0.2 available)
+ pub_semver 2.1.4
+ pub_updater 0.2.4 (0.4.0 available)
+ pubspec 2.3.0
```

```
Building package executables... (4.5s)
Built flutterfire_cli:flutterfire.
Installed executable flutterfire.
Warning: Pub installs executables into C:\Users\Soham\AppData\Local\Pub\Cach
e\bin, which is not on your path.
You can fix that by adding that directory to your system's "Path" environmen
t variable.
A web search for "configure windows path" will show you how.
Activated flutterfire_cli 0.2.7.
```


Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS S:\Code\Flutter apps\sports_tracker> flutter pub add firebase_core
Resolving dependencies...
+ firebase_core 2.25.4
+ firebase_core_platform_interface 5.0.0
+ firebase_core_web 2.11.4
flutter_lints 2.0.3 (3.0.1 available)
+ flutter_web_plugins 0.0.0 from sdk flutter
+ js 0.6.7 (0.7.1 available)
lints 2.1.1 (3.0.0 available)
matcher 0.12.16 (0.12.16+1 available)
material_color_utilities 0.5.0 (0.8.0 available)
meta 1.10.0 (1.12.0 available)
path 1.8.3 (1.9.0 available)
+ plugin_platform_interface 2.1.8
test_api 0.6.1 (0.7.0 available)
web 0.3.0 (0.5.0 available)
Changed 6 dependencies!
9 packages have newer versions incompatible with dependency constraints.
Try `flutter pub outdated` for more information.
Building with plugins requires symlink support.
Please enable Developer Mode in your system settings. Run
start ms-settings:developers
to open settings.


PS S:\Code\Flutter apps\sports_tracker> flutterfire configure
i Found 5 Firebase projects.
✓ Select a Firebase project to configure your Flutter application with · fcbayern-c79b2 (FCBayern)
? Which platforms should your configuration support (use arrow keys & space
✓ Which platforms should your configuration support (use arrow keys & space to select)? · android, ios, macos, web
i Firebase android app com.example.sports_tracker registered.
i Firebase ios app com.example.sportsTracker is not registered on Firebase project fcbayern-c79b2.
i Registered a new Firebase ios app on Firebase project fcbayern-c79b2.
i Firebase macos app com.example.sportsTracker.RunnerTests is not registered on Firebase project fcbayern-c79b2.
i Registered a new Firebase macos app on Firebase project fcbayern-c79b2.
i Firebase web app sports_tracker (web) registered.
Firebase configuration file lib\firebase_options.dart generated successfully with the following Firebase apps:
Platform Firebase App Id
web 1:781276566537:web:69db42cd267aaf0f6c14d8
android 1:781276566537:android:14f079e2f7a065a56c14d8
ios 1:781276566537:ios:563eabe883668cee6c14d8
macos 1:781276566537:ios:ec5b9af593ca65f56c14d8
Learn more about using this file and next steps from the documentation:
> https://firebase.google.com/docs/flutter/setup
PS S:\Code\Flutter apps\sports_tracker>

```
Future <void> main() async {
  WidgetsFlutterBinding.ensureInitialized();

  await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform;
  runApp(MaterialApp(
    home: Home(),
    theme: ThemeData(
      fontFamily: 'Teko',
    ),  // ThemeData
  ));  // MaterialApp
}
```



Code:

```
import 'dart:io';
import 'dart:typed_data';


import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_database/firebase_database.dart';
import 'dart:convert';
import 'package:http/http.dart' as http;
import 'package:firebase_storage/firebase_storage.dart';
```

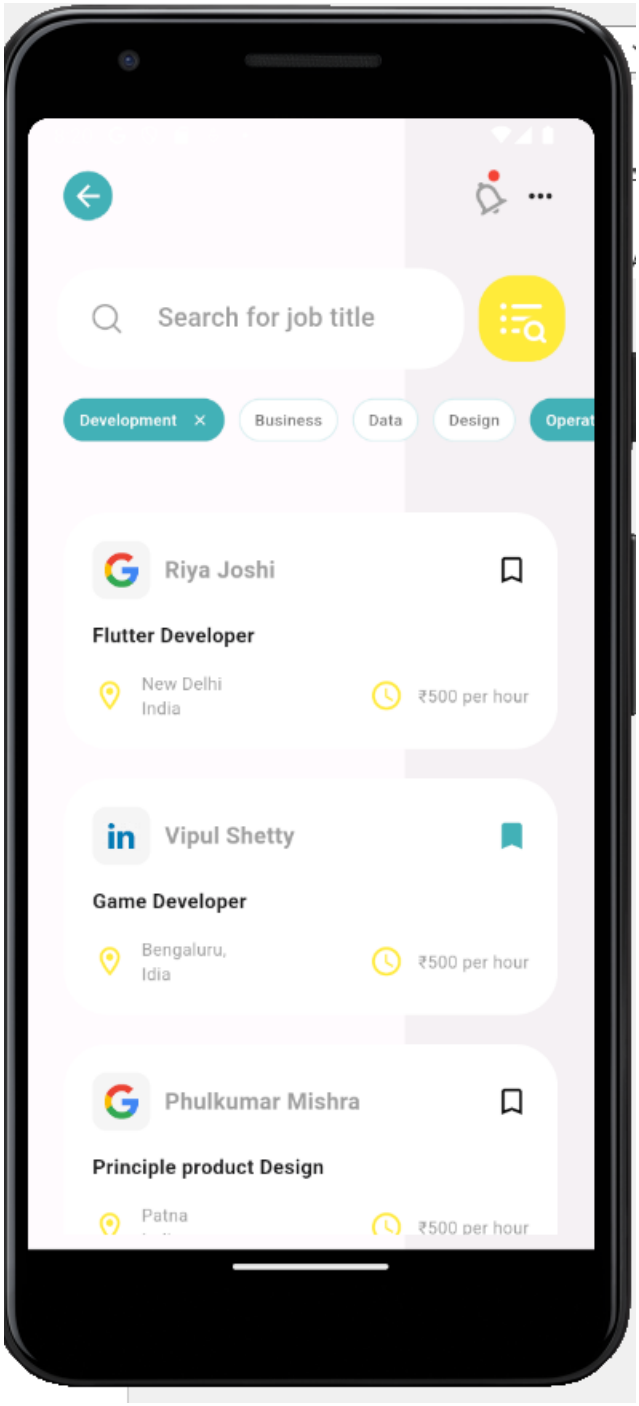
```
//List<dynamic> entire_squad = [];


void fetch_data() async {
 const API_KEY = "12457496b72249c29dd458fe11268b2a";

 var headers = {"X-Auth-Token": API_KEY};

 final Uri squad = Uri.parse("https://api.football-data.org/v4/teams/5");

 final Uri points =
Uri.parse("https://api.football-data.org/v4/competitions/2002/standings");

 final Uri matches = Uri.parse("https://api.football-data.org/v4/teams/5/matches/");

 final response_squad = await http.get(squad, headers :headers,);

 final response_points = await http.get(points, headers :headers,);

 final response_matches = await http.get(matches, headers :headers,);

 if (response_squad.statusCode == 200 && response_points.statusCode==200 &&
response_matches.statusCode==200) {
   // If the call to the server was successful, parse the JSON
   var data_squad = json.decode(response_squad.body);
   var data_points = json.decode(response_points.body);
   var data_matches = json.decode(response_matches.body);

   var bundesliga_link = data_points['competition']['emblem'];
   var bundesliga_name = data_points['competition']['name'];
   final Uri bundesliga_em = Uri.parse(bundesliga_link);
   final response_bundes_img= await http.get(bundesliga_em);

   if (response_bundes_img.statusCode==200){
     upload_firestore(response_bundes_img, bundesliga_name);
   }
   else{
     print("Image not fetched");
   }
```

```
  //print(response_bundes_img.body);




createRecord(data_squad['squad'],data_points['standings'][0]['table'],data_matches['mat
ches']);



  //createRecord(data_points['standings'], "Points_table/");
  // print(data_points['standings'][0]['table']);

 } else {
  // If that call was not successful, throw an error.
  throw Exception('Failed to load data');
 }
}


// Sign Up
Future<int> signUp(String email, String password) async {
 try {
  UserCredential userCredential = await
FirebaseAuth.instance.createUserWithEmailAndPassword(
    email: email,
    password: password,
  );
  User? user = userCredential.user;
  // Handle successful sign up
  return 0; // Return 0 for success
 } catch (e) {
  print('Sign up failed: $e');
  // Handle sign up failure
  return -1; // Return -1 for failure
 }
}

Future<int> signIn(String email, String password) async {
```

```
 try {
   final UserCredential credential = await
FirebaseAuth.instance.signInWithEmailAndPassword(
       email: email,
       password: password
   );
   User? user = credential.user;
   print(user);
   return 0;
 }
 // on FirebaseAuthException catch (e) {
 //   if (e.code == 'user-not-found') {
 //     return 1;
 //   } else if (e.code == 'wrong-password') {
 //     return -1;
 //   }
 catch(e){
   print(e);
   return 1;
 }
}
//
// Future<Object?> getsquad(var squad) async {
//   FirebaseDatabase database = FirebaseDatabase.instance;
//   final ref = FirebaseDatabase.instance.ref();
//
//
//
//   try {
//   final snapshot = await ref.child('Squad').get();
//    var squad = snapshot.value;
//   }
//   catch (error) {
//   }
// }


// Object getData() {
//   final ref = FirebaseDatabase.instance.ref();
//   try {
```

```
//     // Perform the asynchronous operation without await
//     final snapshot = ref.child('Squad').get();
//
//     // Directly return the value without waiting for the future to complete
//     return snapshot.then((snapshot) => snapshot.value);
//   } catch (error) {
//     return [];
//   }
// }
 void createRecord(var data_squad,var data_points , var data_matches) async{
  DatabaseReference ref_squad = FirebaseDatabase.instance.ref("Squad/");
  DatabaseReference ref_points = FirebaseDatabase.instance.ref("Points_table/");
  DatabaseReference ref_matches = FirebaseDatabase.instance.ref("Matches/");

  await ref_squad.set(data_squad);
  await ref_points.set(data_points);
  await ref_matches.set(data_matches);
 }

 void upload_firestore(var img, var name) async{

  String imageName = 'image.png'; // Choose a name for your image
  // firebase_storage.Reference ref =
firebase_storage.FirebaseStorage.instance.ref().child('images/$imageName');
  //await ref.putData(bytes);


  try{
    final storageRef =
FirebaseStorage.instance.ref().child("Competitions/${name}/$imageName");
    Uint8List bytes = img.bodyBytes;


    await storageRef.putData(bytes);
    print('image added');

  }
  catch(e){
    print("Image not added");
  }
```

```
 //File file = File(img);
 //
 // try {
 //  final storageRef =
FirebaseStorage.instance.ref().child("Competitions/${name}/$imageName");
 //   Uint8List bytes = img.bodyBytes;
 //
 //
 //   await storageRef.putData(bytes);
 // } catch (e) {
 //   print("firestore");
 //   //print(e);
 // }


}
```



Conclusion : From this experiment, first of all we studied how to setup firebase. Then we created a firebase project, added multiple dependencies and packages to our flutter project so as to integrate our flutter project with firebase. Next step we authenticate the input fields from our app like email and password using Firebase and stored it.