EXPERIMENT-6

Aim: To Connect Flutter UI with firebase database.

Lab Outcome: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS.

Theory:

1. ***What is Firebase?***

Firebase is a set of backend cloud computing services and application development platforms provided by Google. It hosts databases, services, authentication, and integration for a variety of applications, including Android, iOS, JavaScript, Node.js, Java, Unity, PHP, C++. Firebase provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiment. Firebase offers a number of services, including:

- Analytics – Google Analytics for Firebase offers free, unlimited reporting on as many as 500 separate events. Analytics presents data about user behavior in iOS and Android apps, enabling better decision-making about improving performance and app marketing.
- Authentication – Firebase Authentication makes it easy for developers to build secure authentication systems and enhances the sign-in and onboarding experience for users. This feature offers a complete identity solution, supporting email and password accounts, phone auth, as well as Google, Facebook, GitHub, Twitter login and more.
- Cloud messaging – Firebase Cloud Messaging is a cross-platform messaging tool that lets companies reliably receive & deliver messages on iOS, Android & the web at no cost.
- Realtime database – the Firebase Realtime Database is a cloud-hosted NoSQL database that enables data to be stored and synced between users in real time. The data is synced across all clients in real time and is still available when an app goes offline.
- Crashlytics – Firebase Crashlytics is a real-time crash reporter that helps developers track, prioritize and fix stability issues that reduce the quality of their apps. With crashlytics, developers spend less time organizing and troubleshooting crashes and more time building features for their apps.
- Performance – Firebase Performance Monitoring service gives developers insight into the performance characteristics of their iOS and Android apps to help them determine where and when the performance of their apps can be improved.
- Test lab – Firebase Test Lab is a cloud-based app-testing infrastructure. With one operation, developers can test their iOS or Android apps across a variety of devices and device configurations. They can see the results, including videos, screenshots and logs, in the Firebase console.

2. ***What is Cloud Firestore?***

Firestore is a NoSQL document database built for automatic scaling, high performance, and ease of application development. Unlike a SQL database, there are no tables or rows. Instead, you store data in *documents*, which are organized into *collections*.

Each *document* contains a set of key-value pairs. Firestore is optimized for storing large collections of small documents. All documents must be stored in collections. Documents can contain *subcollections* and nested objects, both of which can include primitive fields like strings or complex objects like lists.

Collections and documents are created implicitly in Firestore. Simply assign data to a document within a collection. If either the collection or document does not exist, Firestore creates it.

Features of Firebase:

- Firestore is designed for cloud and native apps.
- Offers ACID(atomicity, consistency, isolation, and durability) transaction for data consistency and integrity.
- Multi-region replication enhances platform security.
- Provides serverless development with client-SDK and backend security controls.
- Autoscaling allows for consistent performance even as the database grows.
- Supports offline mode for iOS, Android, and web apps.
- Powerful query engine for complex queries without degradation of performance.
- Top-notch security features including automatic data validation and disaster recovery.

Firestore offers several advantages:

- Offline synchronization for web, Android, and iOS apps.
- Fully integrated with Firebase and Google Cloud, providing access to both platforms' features.
- Serverless and scalable architecture simplifies development and allows for horizontal scaling.
- Improved querying with support for indexed queries and ACID transactions.
- "Pay as you go" pricing structure based on server resource usage.

3. ***Write down steps to integrate flutter application with firebase.***

Practical & Outcome:

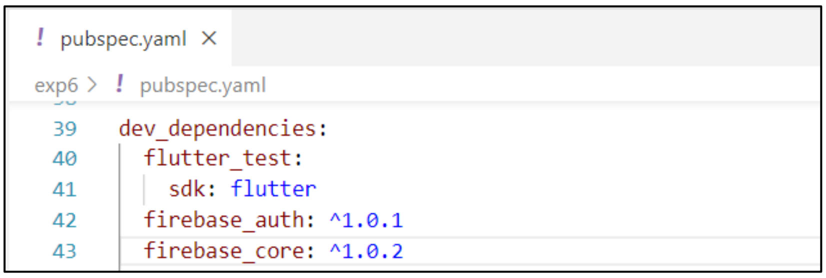
Step 1: Create a Flutter project:

Command: flutter create exp6

Step 2: Add the following dependencies in the pubspec.yaml file:

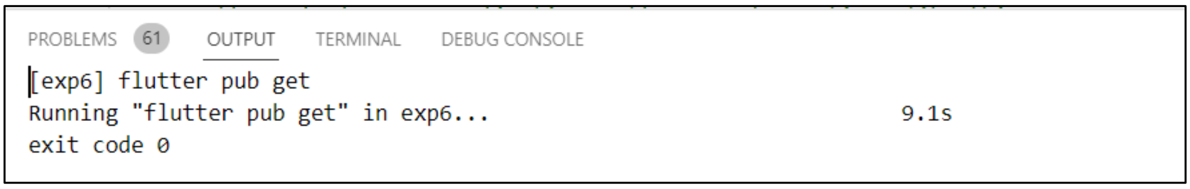
firebase_auth: ^1.0.1 # add this line

firebase_core: ^1.0.2 # add this line



```yaml
dev_dependencies:
  flutter_test:
    sdk: flutter
  firebase_auth: ^1.0.1
  firebase_core: ^1.0.2
```

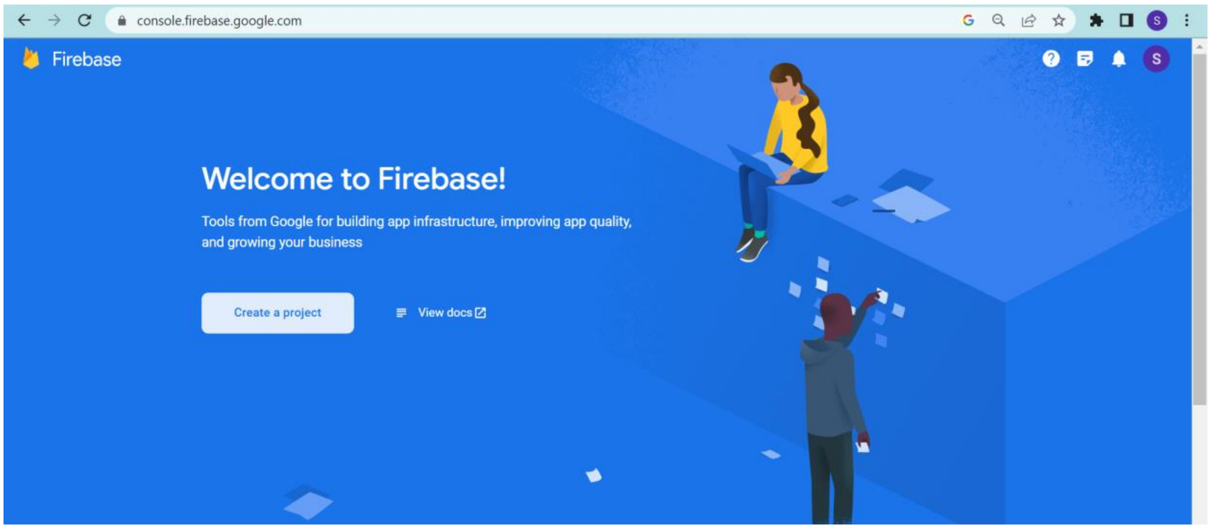
Step 3: Run the command: flutter pub get



```
[exp6] flutter pub get
Running "flutter pub get" in exp6...                              9.1s
exit code 0
```

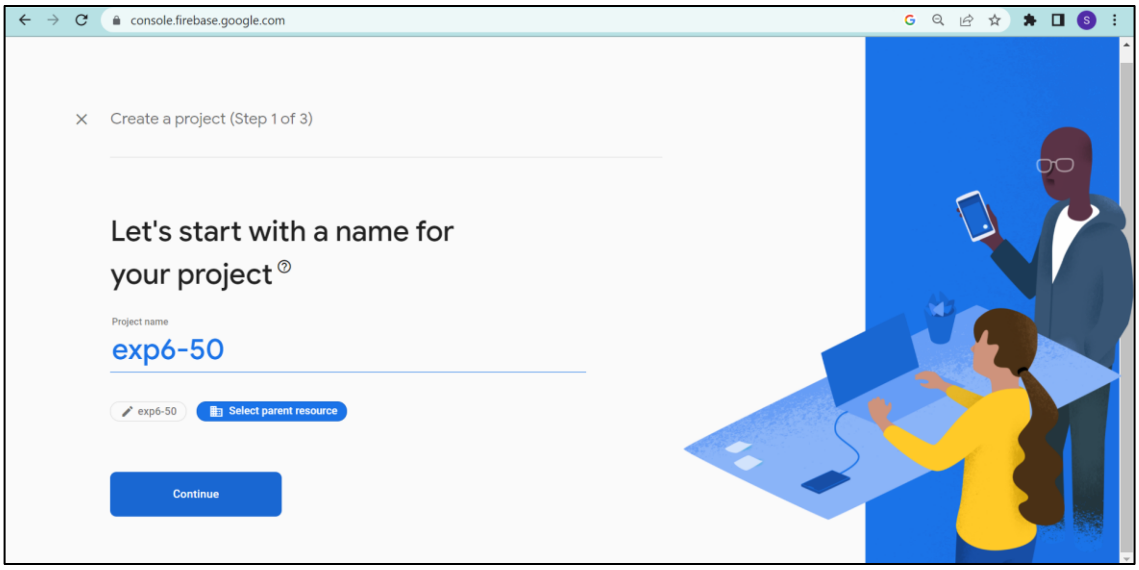
Step 4: Visit firebase.google.com and perform the following steps:

1. Visit the Firebase Console.

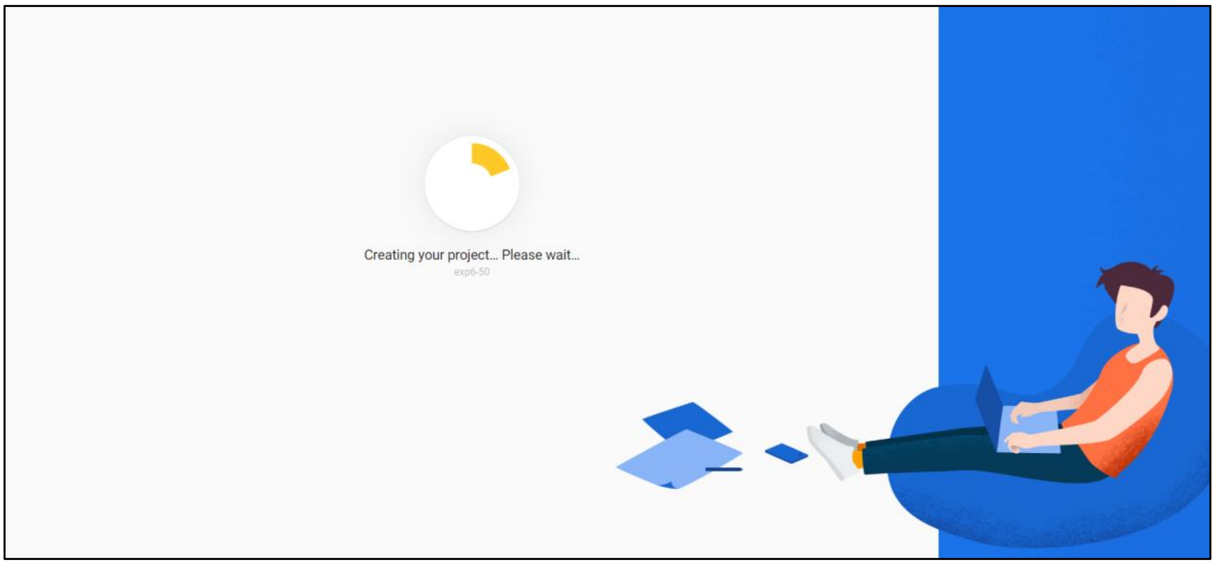


console.firebase.google.com
Firebase
Welcome to Firebase!
Tools from Google for building app infrastructure, improving app quality, and growing your business
Create a project
View docs

2. Start by adding a New Project.


console.firebase.google.com
Create a project (Step 1 of 3)
Let's start with a name for your project
Project name
exp6-50
exp6-50
Select parent resource
Continue

3. Give the project a name and agree to all they've asked.
4. Click 'Create Project.'

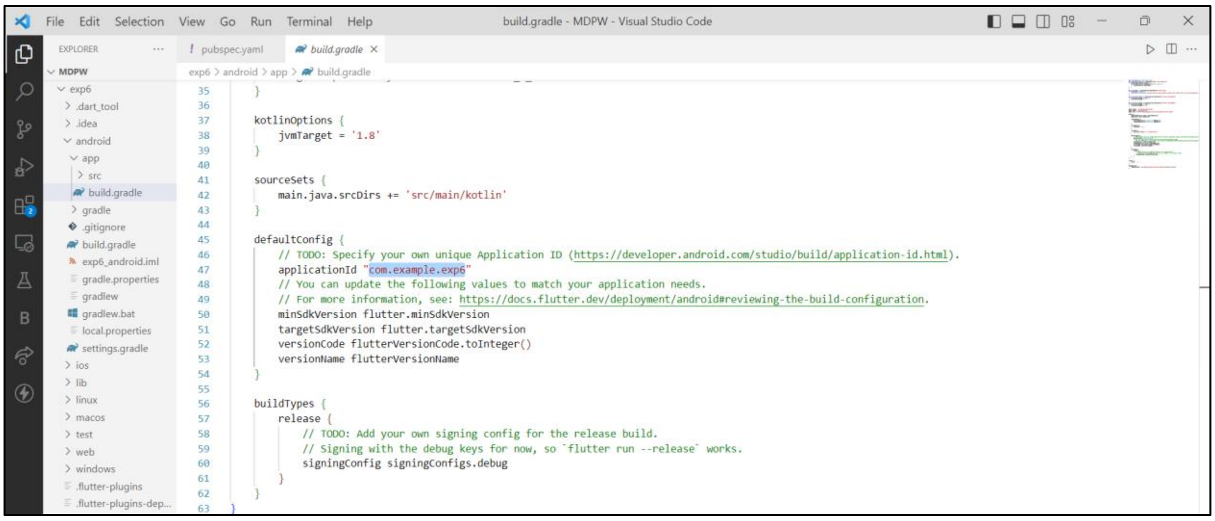


Once you're done creating the project, it's time to integrate it with Android and iOS applications.

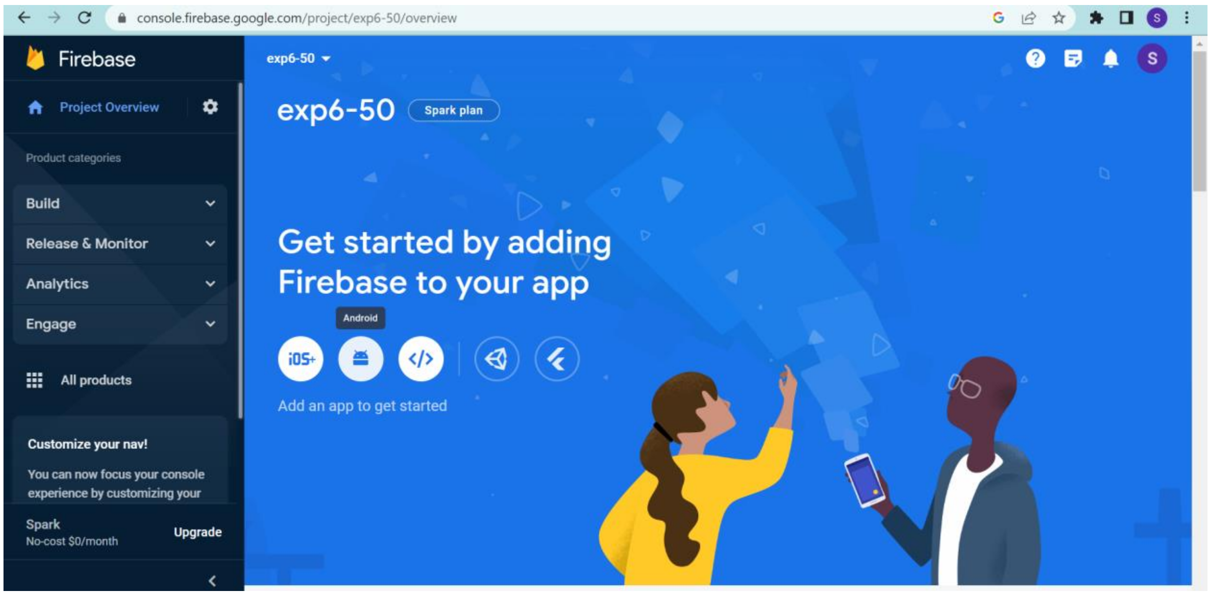
Step 5: Configure Firebase with Android and iOS Application

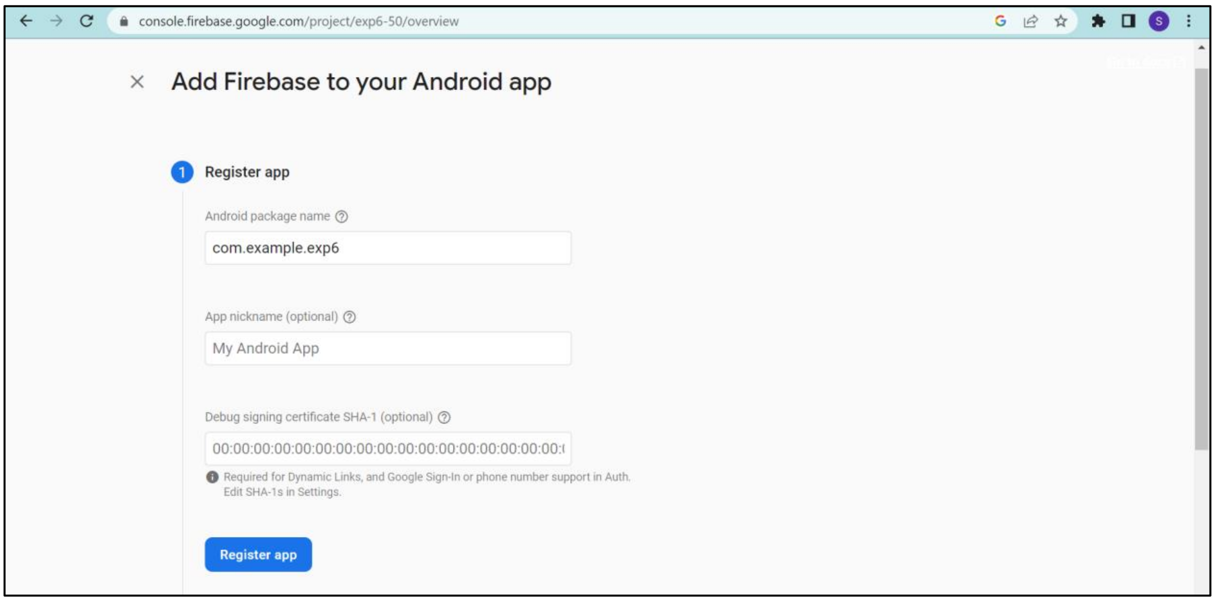
Register Android application

For registering the android app, you have to provide a unique package name. For that, you can find it at *android*→ *app*→ *build.gradle*, by default it will take *com.example.application_name*. The next **field App nickname** is optional.



```
    }

    kotlinOptions {
        jvmTarget = '1.8'
    }

    sourceSets {
        main.java.srcDirs += 'src/main/kotlin'
    }

    defaultConfig {
        // TODO: Specify your own unique Application ID (https://developer.android.com/studio/build/application-id.html).
        applicationId "com.example.exp6"
        // You can update the following values to match your application needs.
        // For more information, see: https://docs.flutter.dev/deployment/android#reviewing-the-build-configuration.
        minSdkVersion flutter.minSdkVersion
        targetSdkVersion flutter.targetSdkVersion
        versionCode flutterVersionCode.toInteger()
        versionName flutterVersionName
    }

    buildTypes {
        release {
            // TODO: Add your own signing config for the release build.
            // Signing with the debug keys for now, so `flutter run --release` works.
            signingConfig signingConfigs.debug
        }
    }
}
```


console.firebase.google.com/project/exp6-50/overview
Firebase
Project Overview
Product categories
Build
Release & Monitor
Analytics
Engage
All products
Customize your nav!
You can now focus your console experience by customizing your
Spark
No-cost $0/month
Upgrade
exp6-50
exp6-50
Spark plan
Get started by adding Firebase to your app
Android
iOS+
Add an app to get started


console.firebase.google.com/project/exp6-50/overview

Add Firebase to your Android app

1 Register app

Android package name: com.example.exp6

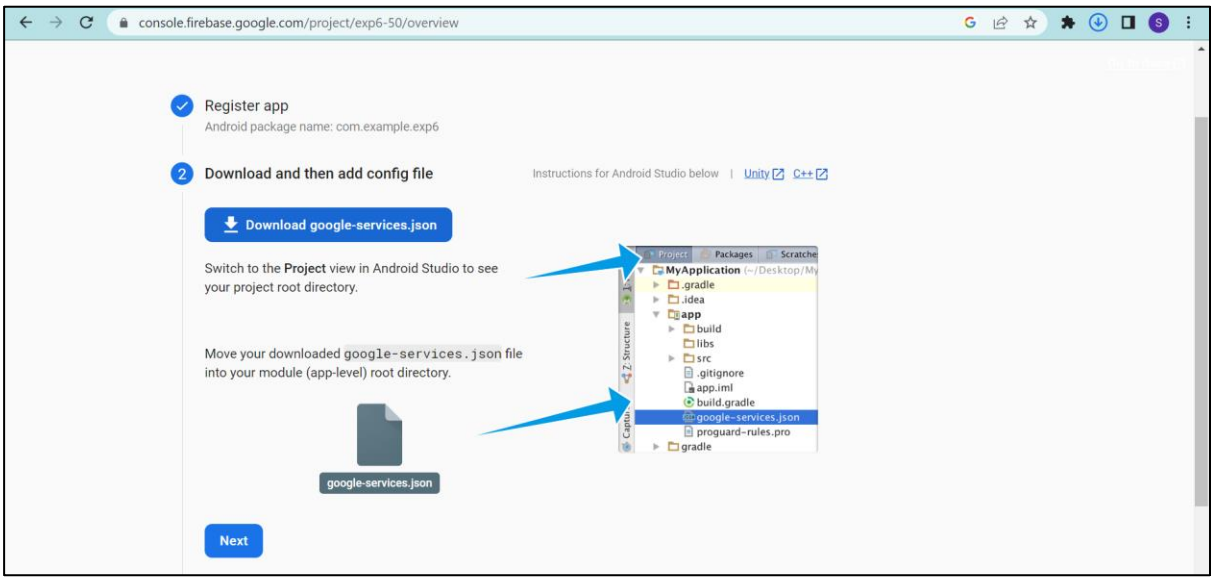
App nickname (optional): My Android App

Debug signing certificate SHA-1 (optional): 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:0

Required for Dynamic Links, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

Download Android Config file

After filling the required fields, you'll see something like this. Download *google-services.json* and place it in *MyApplication/app* folder.



console.firebase.google.com/project/exp6-50/overview

Register app
Android package name: com.example.exp6

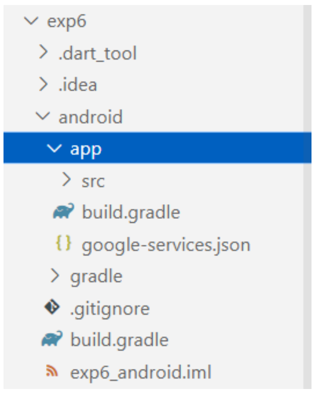
Download and then add config file

Instructions for Android Studio below | Unity C++

Download google-services.json

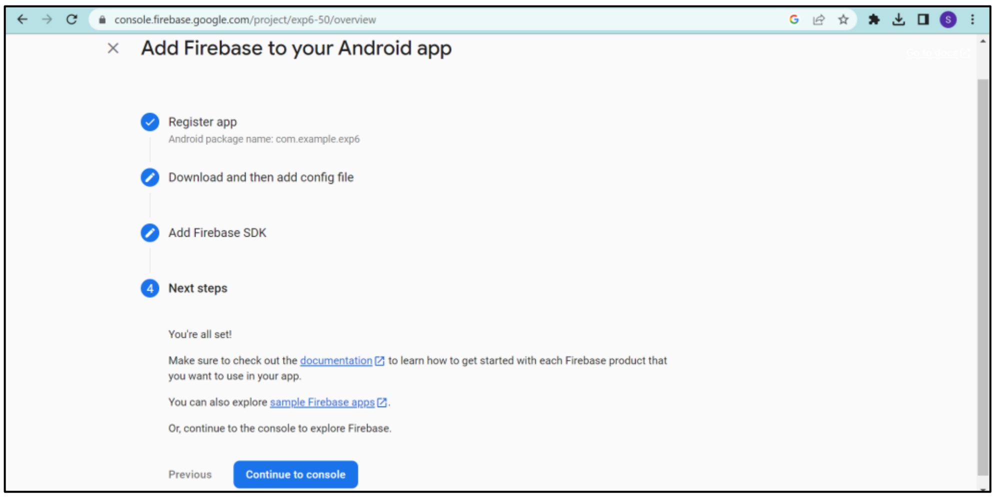
Switch to the **Project** view in Android Studio to see your project root directory.

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



```
exp6
  .dart_tool
  .idea
  android
    app
      src
      build.gradle
      google-services.json
    gradle
    .gitignore
    build.gradle
    exp6_android.iml
```

You can skip the next step as we are using: *Firebase core and Firebase auth packages*. Click 'Continue to Console' to complete the process.



console.firebase.google.com/project/exp6-50/overview

Add Firebase to your Android app

Register app
Android package name: com.example.exp6

Download and then add config file

Add Firebase SDK

Next steps

You're all set!

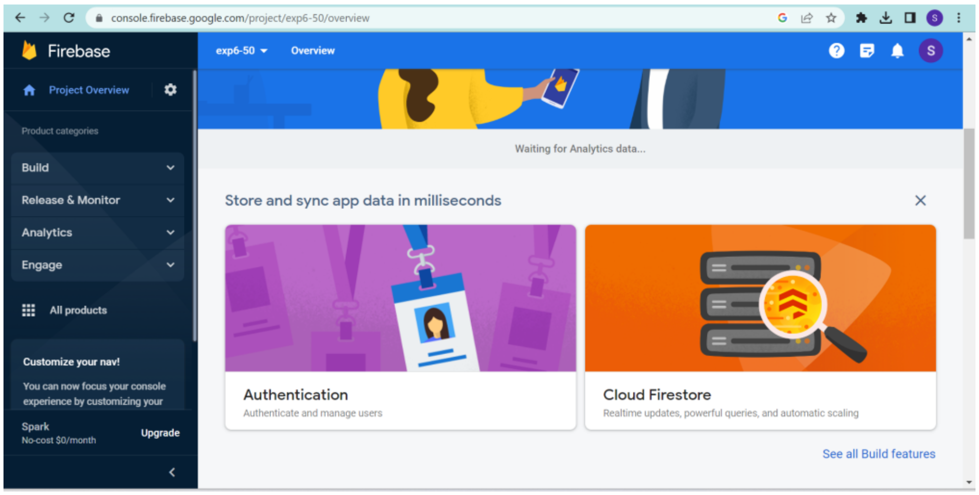
Make sure to check out the documentation to learn how to get started with each Firebase product that you want to use in your app.

You can also explore sample Firebase apps.

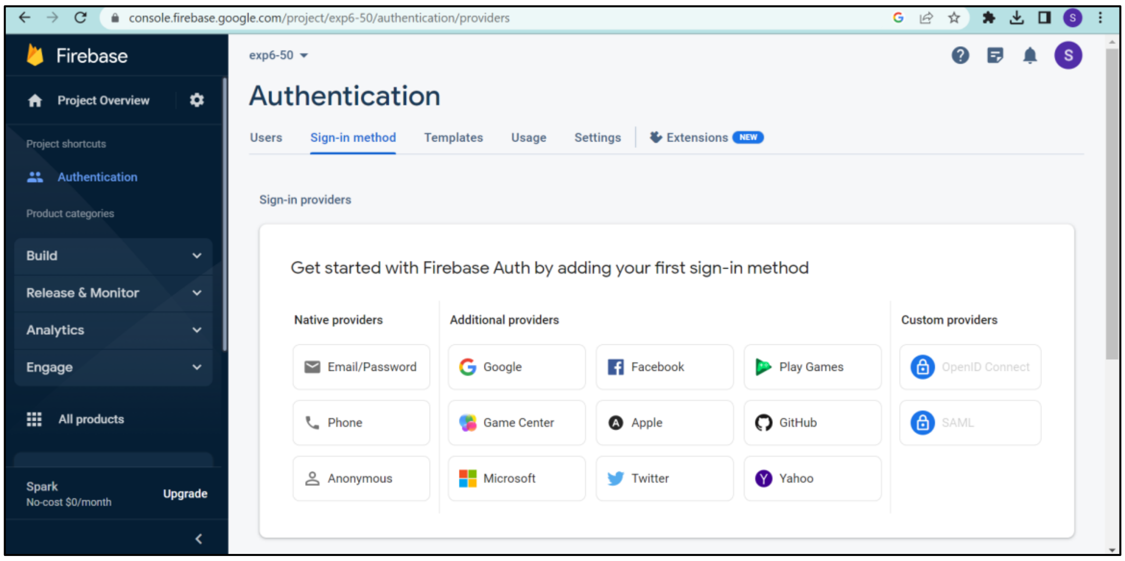
Or, continue to the console to explore Firebase.

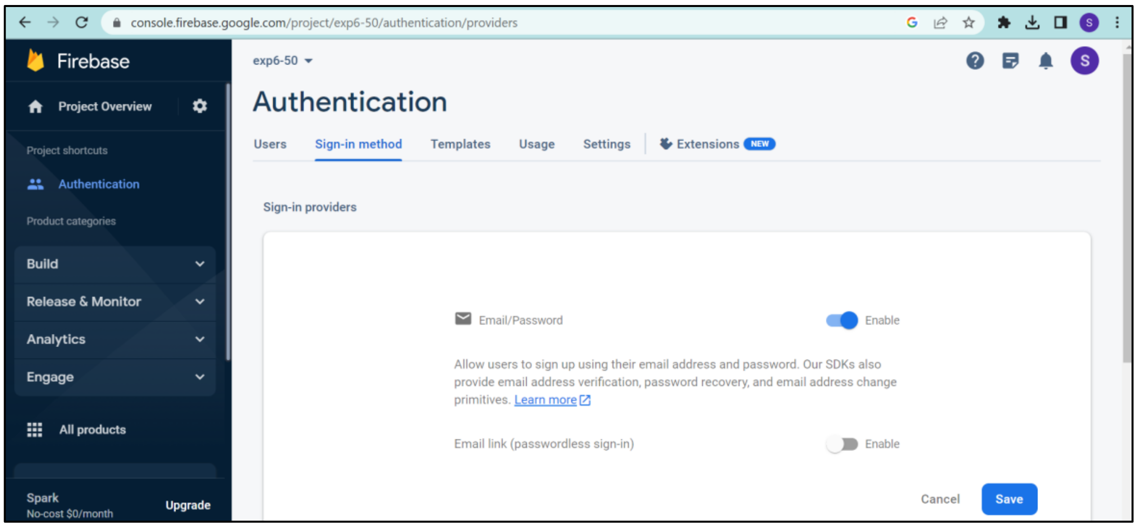
Enable Email/Password Authentication in Firebase

On visiting the Firebase dashboard, click 'Authentication.'


console.firebase.google.com/project/exp6-50/overview
Firebase
exp6-50
Overview
Project Overview
Product categories
Build
Release & Monitor
Analytics
Engage
All products
Customize your nav!
You can now focus your console experience by customizing your
Spark
No-cost $0/month
Upgrade
Waiting for Analytics data...
Store and sync app data in milliseconds
Authentication
Authenticate and manage users
Cloud Firestore
Realtime updates, powerful queries, and automatic scaling
See all Build features

Under the Sign-in method click 'Email/Password' and enable it using the toggle button. Use the below screenshots for your reference.





Email/Password — Enable: on

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: off

Initialize Firebase App

Open *main.dart* and use the following code to initialize Firebase App:

```
future main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(MyApp());
}
```

Create a file authentication.dart and use the following code for creating helper class for authentication.

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

class AuthenticationHelper {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  get user => _auth.currentUser;

 //SIGN UP METHOD
  Future signUp({String email, String password}) async {
    try {
      await _auth.createUserWithEmailAndPassword(
        email: email,
        password: password,
      );
      return null;
    } on FirebaseAuthException catch (e) {
      return e.message;
    }
  }

  //SIGN IN METHOD
  Future signIn({String email, String password}) async {
    try {
      await _auth.signInWithEmailAndPassword(email: email, password: password);
      return null;
    } on FirebaseAuthException catch (e) {
      return e.message;
    }
  }

  //SIGN OUT METHOD
  Future signOut() async {
    await _auth.signOut();

    print('signout');
  }
}
```

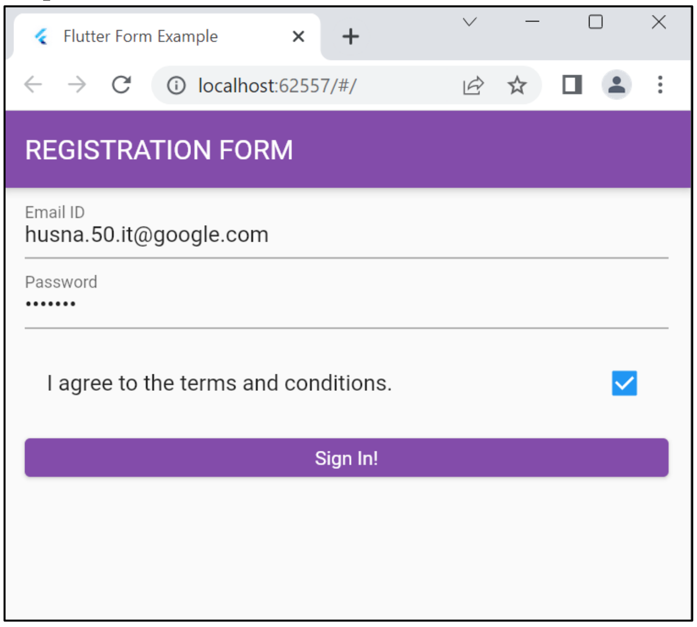
Signup.dart

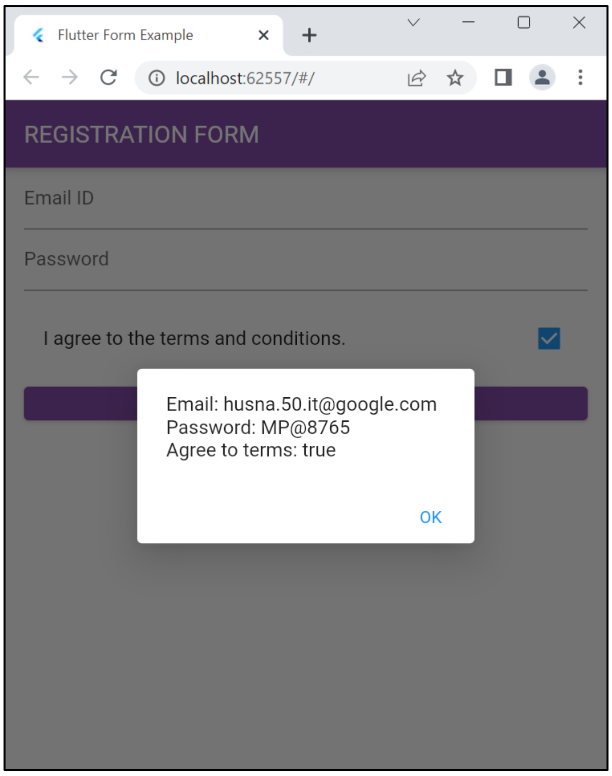
```
// Get username and password from the user.Pass the data to helper method
AuthenticationHelper()
   .signUp(email: email, password: password)
   .then((result) {
         if (result == null) {
        Navigator.pushReplacement(context,
           MaterialPageRoute(builder: (context) => Home()));
       } else {
          Scaffold.of(context).showSnackBar(SnackBar(
          content: Text(
              result,
              style: TextStyle(fontSize: 16),
             ),
          ));
       }});
```

Login.dart

```
AuthenticationHelper()
   .signIn(email: email, password: password)
      .then((result) {
         if (result == null) {
           Navigator.pushReplacement(context,
             MaterialPageRoute(builder: (context) => Home()));
          } else {
             Scaffold.of(context).showSnackBar(SnackBar(
                 content: Text(
                 result,
                style: TextStyle(fontSize: 16),
                   ),
               ));
           }});
```

Step 6: Run the code and fill the form. Then check firebase to see if the entry has been added.




Flutter Form Example

localhost:62557/#/

REGISTRATION FORM

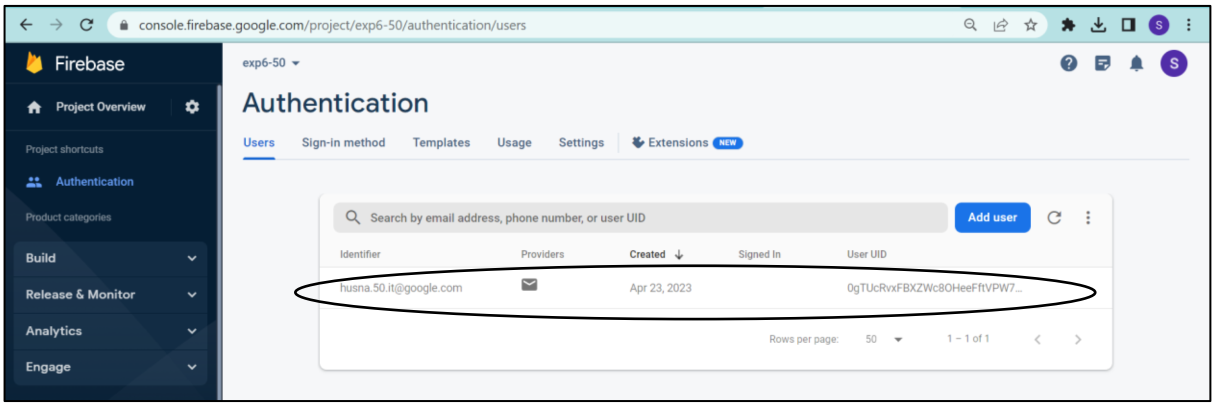
Email ID

Password

I agree to the terms and conditions.

Email: husna.50.it@google.com
Password: MP@8765
Agree to terms: true

OK


console.firebase.google.com/project/exp6-50/authentication/users

Identifier	Providers	Created	Signed In	User UID
husna.50.it@google.com	✉	Apr 23, 2023		0gTUcRvxFBXZWc8OHeeFftVPW7...

Rows per page: 50 1 – 1 of 1

Tools used: Visual Studio Code, Flutter, Firebase.

Conclusion:

From this experiment we have studies about firebase. Further, we have established a connection between our flutter form and firebase.