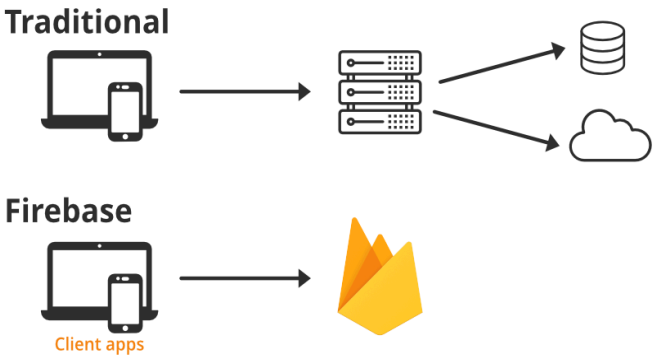
Experiment-6

Aim: To Connect Flutter UI with firebase database.

Theory:

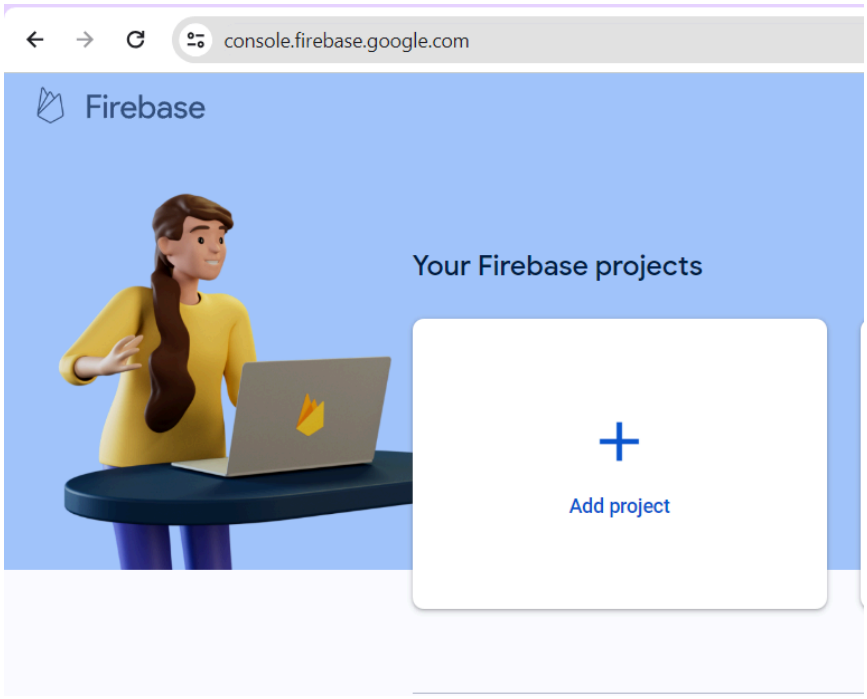
Firebase Authentication is a preconfigured backend service that makes it really easy to integrate with a mobile app using an SDK. You don't have to maintain any backend infrastructure for the authentication process and Firebase supports integration with popular identity providers such as Google, Facebook, and GitHub.

Firebase is a product of Google which helps developers to build, manage, and grow their apps easily. It helps developers to build their apps faster and in a more secure way. No programming is required on the firebase side which makes it easy to use its features more efficiently. It provides services to android, ios, web, and unity. It provides cloud storage. It uses NoSQL for the database for the storage of data.

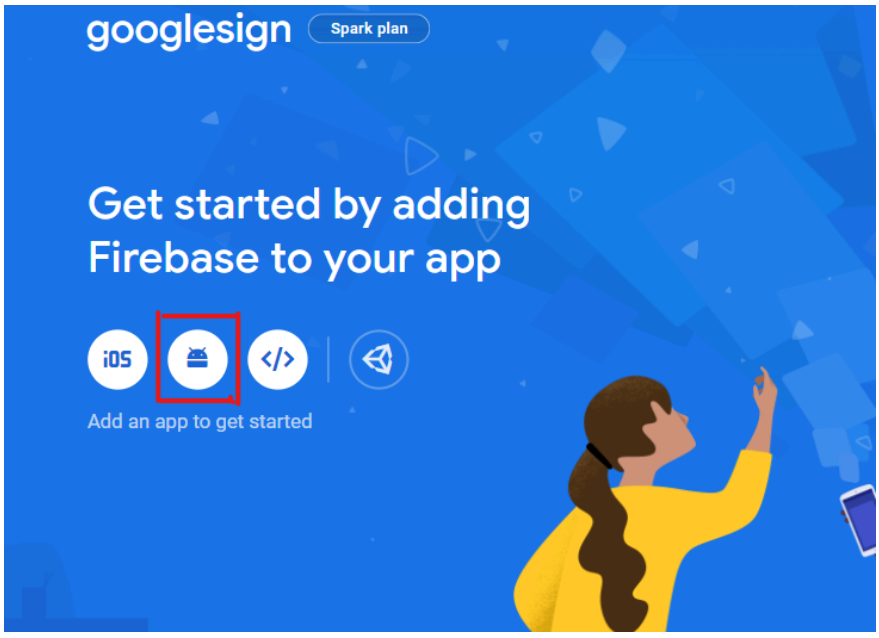


Step by Step Implementation

Step 1: First, you have to visit the Firebase console. Now let's move to the next step. Click on the "Add project" as shown in the below image.

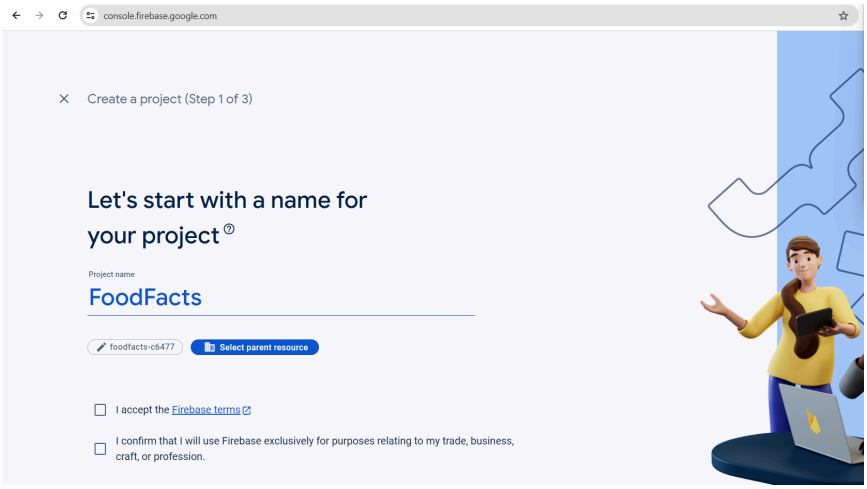


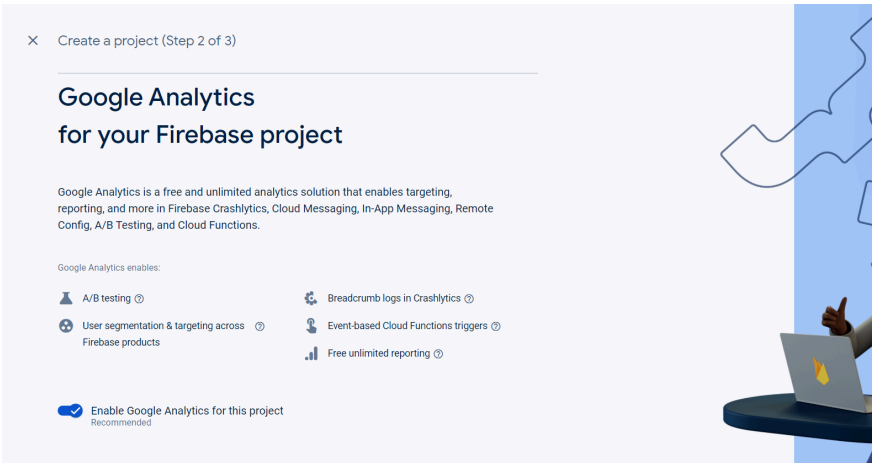
Step 2: Provide the firebase project name. Let's give it "geeksforgeeks". Then click on "Continue". Disable the "Enable Google Analytics for this project" because we don't need this and click on "Create project". It takes some time to wait till the project is created, After that click on "Continue". Now there will be a screen, You to find the Android button and click on it as shown in the below image.

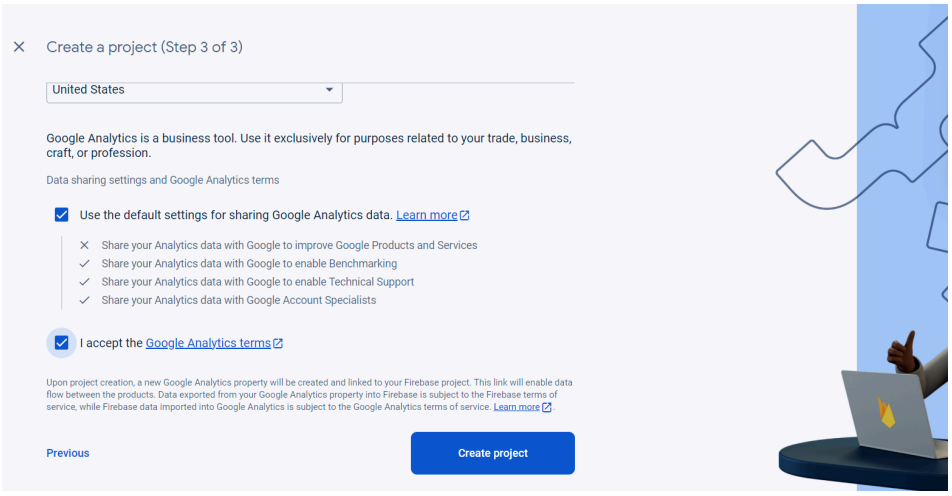


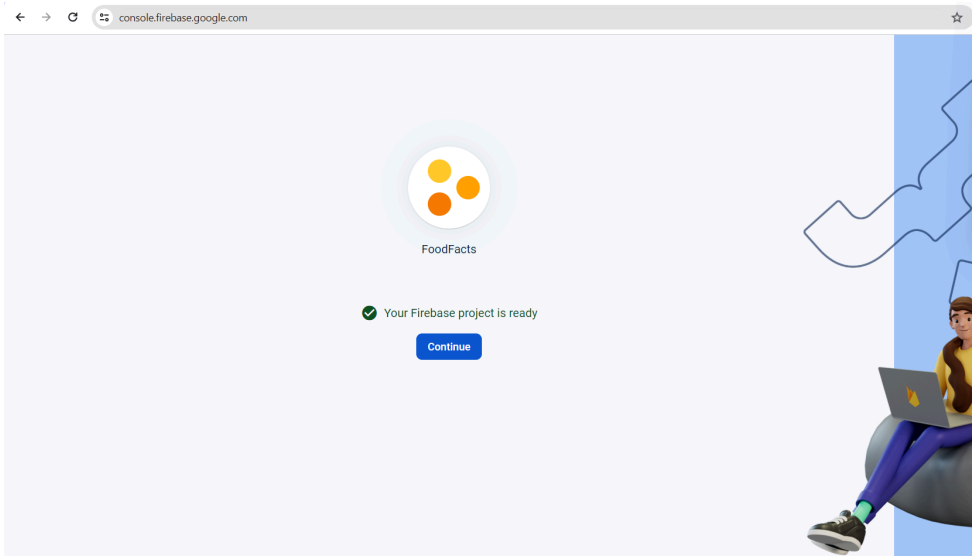
Now time to add firebase to your Android App.

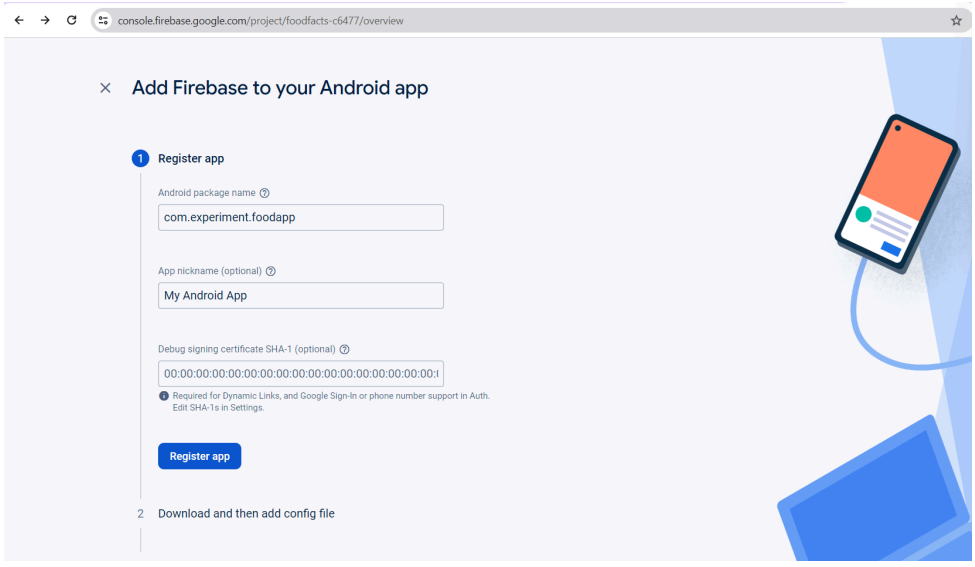
Step 3: We have to give the “Android package name”, where it can be found????. Don’t know, Let’s find this in the next step.

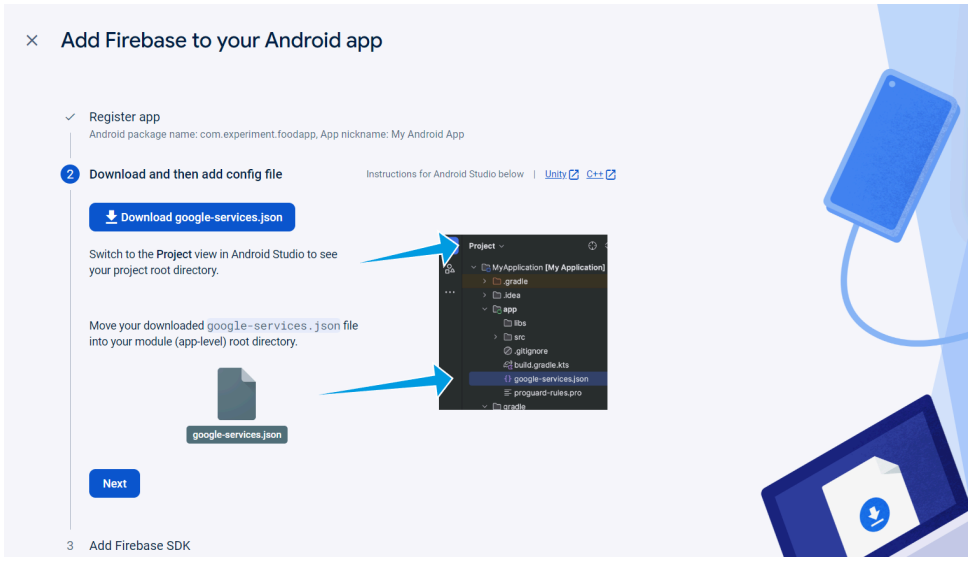







console.firebase.google.com
FoodFacts
Your Firebase project is ready
Continue


console.firebase.google.com/project/foodfacts-c6477/overview
Add Firebase to your Android app
1 Register app
Android package name
com.experiment.foodapp
App nickname (optional)
My Android App
Debug signing certificate SHA-1 (optional)
Required for Dynamic Links, and Google Sign-In or phone number support in Auth.
Edit SHA-1s in Settings.
Register app
2 Download and then add config file


Add Firebase to your Android app
Register app
Android package name: com.experiment.foodapp, App nickname: My Android App
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
Project
MyApplication [My Application]
.gradle
.idea
app
libs
src
.gitignore
build.gradle.kts
google-services.json
proguard-rules.pro
gradle
Next
3 Add Firebase SDK

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 "4.4.1" 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`):

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:32.7.3"))

  // TODO: Add the dependencies for Firebase products you want to use
  // When using the BoM, don't specify versions in Firebase dependencies
  implementation("com.google.firebase:firebase-analytics")

  // Add the dependencies for any other desired Firebase products
  // 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

3. After adding the plugin and the desired SDKs, sync your Android project with Gradle files.

Previous **Next**

× Add Firebase to your Android app

✓ Register app
Android package name: com.experiment.foodapp, App nickname: My Android App

✎ Download and then add config file

✎ Add Firebase SDK

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.

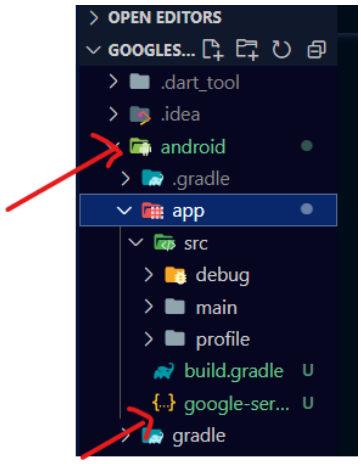
You can also explore sample Firebase apps.

Or, continue to the console to explore Firebase.

Previous **Continue to console**

Step 4: Let's go to your flutter project, and click on the "Android" folder, and in the App-level build gradle file you find Application id just copy it and paste it to "Android package name".

Step 5: In "App nick-name" you give any name or leave it blank because it is optional. In "Debug signing certificate SHA-1 (optional) " also leave it blank for the time or you can give the debug SHA keys. Now click on "Register app". Now you have to "Download Config file", Switch to the Project view in Android Studio/vs code to see your project root directory. Move the "google-services.json" file you just downloaded into your Android app module root directory.



Now click on "Next".

Step 6: Add Firebase SDK. The Google services plugin for Gradle loads the google-services.json file you just downloaded. Modify your build.gradle files to use the plugin. Project-level build.gradle (<project>/build.gradle):

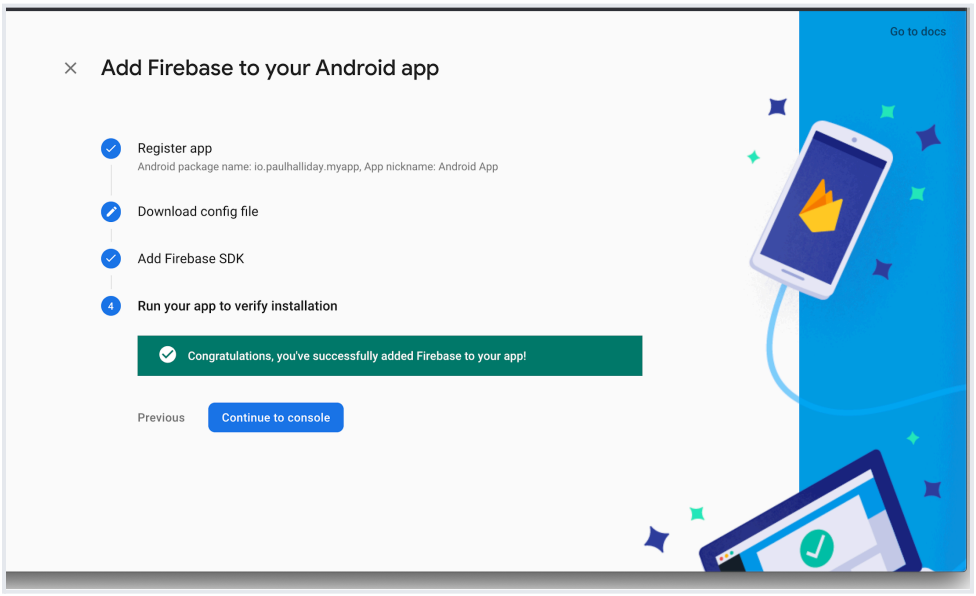
```
dependencies {
    classpath 'com.android.tools.build:gradle:4.1.0'
    classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
    classpath 'com.google.gms:google-services:4.3.8'
}
```

Step 7: App-level build.gradle (<project>/<app-module>/build.gradle):

```
apply plugin: 'com.android.application'
apply plugin: 'com.google.gms.google-services'
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply from: "$flutterRoot/packages/flutter_tools/gradle/flutter.gradle"
```

```
dependencies {
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation platform('com.google.firebase:firebase-bom:28.0.1')
}
```

Step 8: After back to the firebase console and click “Next”. And then Click on “Continue to Console”.



To demonstrate with a practical example, we’ll walk you through the process of building an email-password registration and login process.

Create a Flutter and Firebase project

Create a new Flutter project using the following command:

flutter create flutter_authentication

Open the project in your favorite code editor. Here’s how to open it using VS Code:

code flutter_authentication

To integrate Firebase with your Flutter project, you have to create a new Firebase project by going to the console.

Register a new user

When a new user arrives, before logging in, they have to register to the Firebase authentication.

Create a new dart file called fire_auth.dart and define a new method called registerUsingEmailPassword():

```
class FireAuth {
 static Future<User?> registerUsingEmailPassword({
```

```
  required String name,
  required String email,
  required String password,
}) async {
  FirebaseAuth auth = FirebaseAuth.instance;
  User? user;
  try {
    UserCredential userCredential = await auth.createUserWithEmailAndPassword(
      email: email,
      password: password,
    );
    user = userCredential.user;
    await user!.updateProfile(displayName: name);
    await user.reload();
    user = auth.currentUser;
  } on FirebaseAuthException catch (e) {
    if (e.code == 'weak-password') {
      print('The password provided is too weak.');
    } else if (e.code == 'email-already-in-use') {
      print('The account already exists for that email.');
    }
  } catch (e) {
    print(e);
  }
  return user;
}
}
```

Here we are registering a new user using the email and password provided and associating the name of the user with this profile.

There can be various FirebaseAuthException errors, which we have handled in the above code snippet.

User sign-in and sign-out

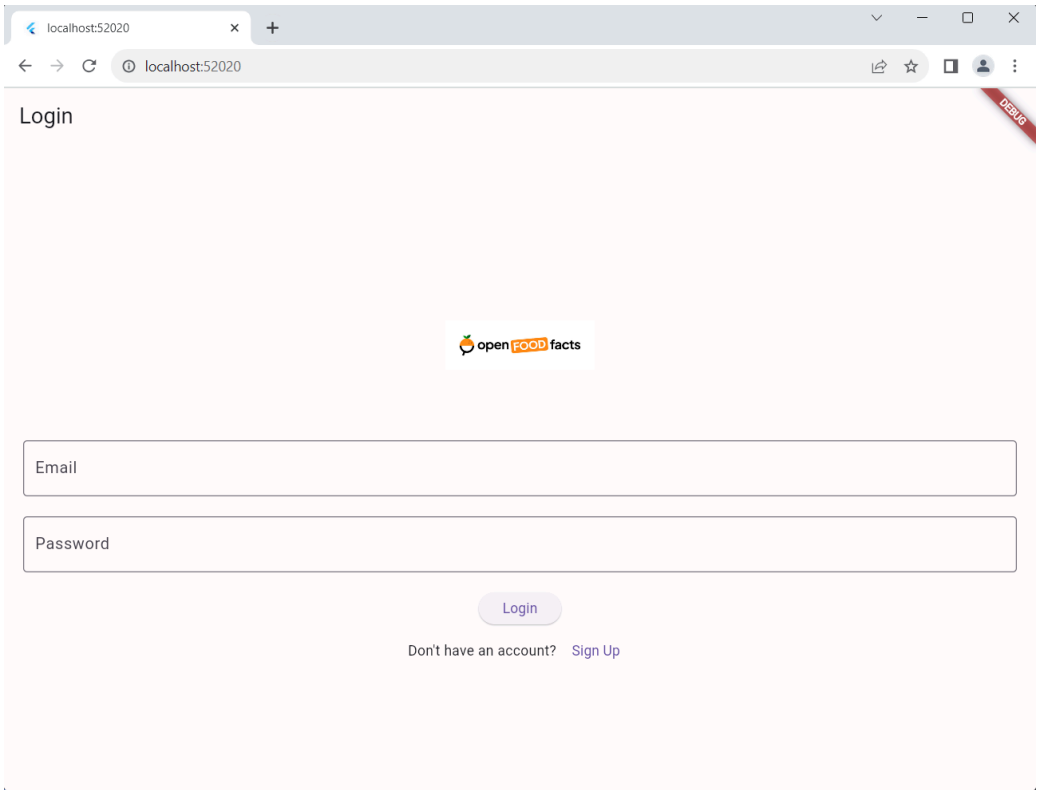
To sign in a user who has already registered in our app, define a new method called signInUsingEmailPassword(), passing the user email and password:

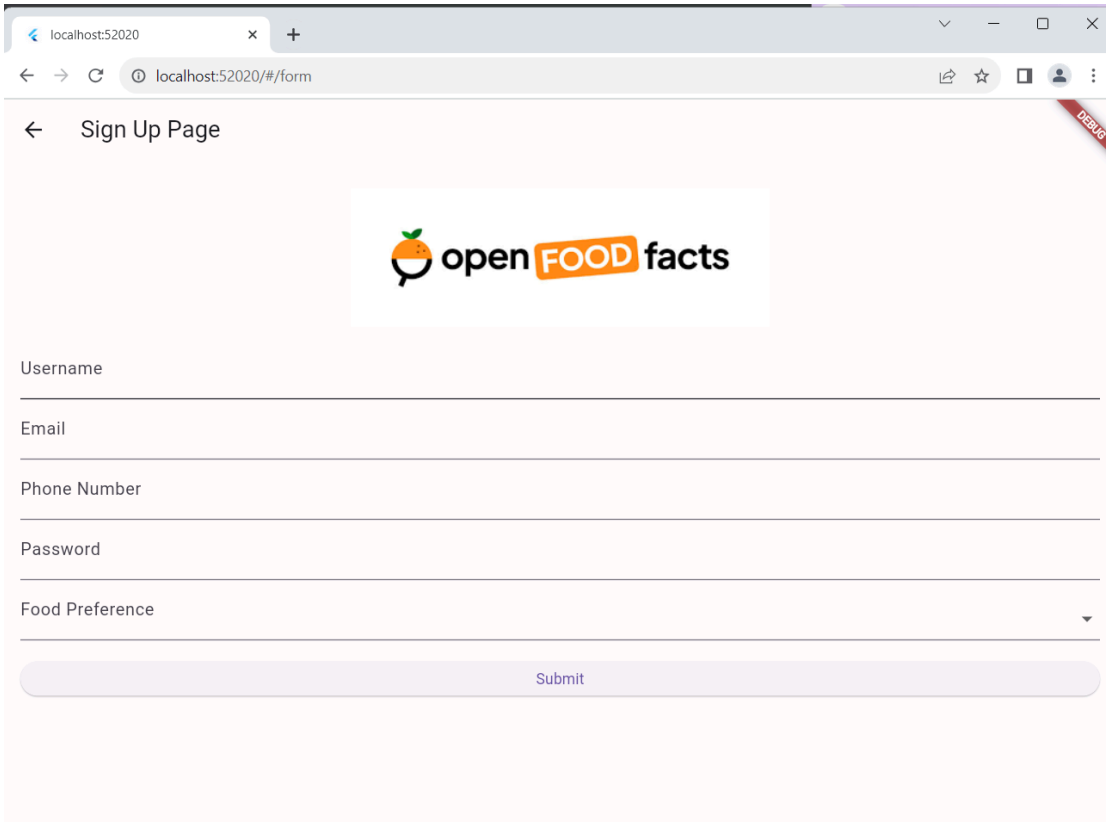
```
static Future<User?> signInUsingEmailPassword({
required String email,
required String password,
required BuildContext context,
}) async {
FirebaseAuth auth = FirebaseAuth.instance;
User? user;

try {
```

```
UserCredential userCredential = await auth.signInWithEmailAndPassword(
email: email,
password: password,
);
user = userCredential.user;
} on FirebaseAuthException catch (e) {
if (e.code == 'user-not-found') {
print('No user found for that email.');
} else if (e.code == 'wrong-password') {
print('Wrong password provided.');
}
}

return user;
}
```

The email and password are used to generate the User object provided by Firebase. The User can be used later to retrieve any additional data (e.g., user name, profile picture, etc.) stored in that account.

You can use the signOut() method to log a user out. There is no need to create a separate method for signing out because it's just a single line of code:
FirebaseAuth.instance.signOut();





Conclusion

We have successfully integrated Firebase Authentication with your Flutter app. As we may have noticed, Firebase Authentication not only provides the backend infrastructure for authenticating users easily, but also the predefined methods for auto login and email verification. And there's a lot more to explore; Firebase Authentication also provides support for integration with a number of identity providers, including Google, Facebook, Twitter, Apple, etc.