

## **EXPERIMENT 06**

**AIM:** To connect firebase with flutter UI

### **STEPS:**

#### **Prerequisites**

- Flutter installed on your system (Download Flutter)
- Firebase account (Go to Firebase Console)
- Android Studio or VS Code installed with Flutter plugin

#### **Step 1: Create a Firebase Project**

1. Go to Firebase Console.
2. Click on "**Create a project**".
3. Enter the project name and agree to terms.
4. Click **Continue** and complete the setup.

#### **Step 2: Register the Flutter App in Firebase**

1. In the Firebase Console, select your project.
2. Click on "**Add app**" and choose:
  - **Android** (for Android setup)
  - **iOS** (for iOS setup)
  - **Web** (for web setup)

3. For Android:
  - Enter the package name (found in `android/app/build.gradle` → `applicationId`).
  - Download the `google-services.json` file and place it in `android/app/` directory.
4. For iOS:
  - Download `GoogleService-Info.plist` and place it in `ios/Runner/`.
  - Open `ios/Runner/Info.plist` and add required Firebase configurations.

### Step 3: Add Firebase SDK to Flutter Project

Open `pubspec.yaml` and add the required Firebase dependencies:

Yaml

dependencies:

flutter:

  sdk: flutter

  firebase\_core: latest\_version

  firebase\_auth: latest\_version # (If using authentication)

  cloud\_firestore: latest\_version # (If using Firestore)

Run:

`flutter pub get`

## Step 4: Configure Firebase for Android

Open `android/build.gradle` and ensure:

```
classpath 'com.google.gms:google-services:latest_version'
```

Open `android/app/build.gradle` and add:

```
apply plugin: 'com.google.gms.google-services'
```

## Step 5: Initialize Firebase in Flutter

1. Open `main.dart` and initialize Firebase before running the app:

```
import 'package:firebase_core/firebase_core.dart';  
import 'package:flutter/material.dart';
```

```
Future<void> main() async {  
  WidgetsFlutterBinding.ensureInitialized();  
  await Firebase.initializeApp();  
  runApp(MyApp());  
}
```

```
class MyApp extends StatelessWidget {  
  @override
```

```
Widget build(BuildContext context) {  
  return MaterialApp(  
    home: Scaffold(  
      appBar: AppBar(title: Text('Firebase Connected')),  
      body: Center(child: Text('Welcome to Firebase with Flutter!')),  
    ),  
  );  
}
```

## **Step 6: Commands to run (for web)**

1. Go to the path of your flutter project.

Cd (your path)

2. Perform commands

Firebase login

dart pub global activate flutterfire\_cli

\$env:Path += ";C:\Users\PC\AppData\Local\Pub\Cache\bin"

flutterfire configure

flutter pub get

flutter pub add cloud\_firestore

Flutter run -d chrome

## Step 7: Run the App

Run the project to verify the connection:

flutter run

### STEPS FROM OFFICIAL DOCUMENT

## Creating a New Flutter Project

This tutorial will require the creation of an example Flutter app.

Once you have your environment set up for Flutter, you can run the following to create a new application:

```
flutter create flutterfirebaseexample
```

1.

[Copy](#)

Navigate to the new project directory:

```
cd flutterfirebaseexample
```

1.

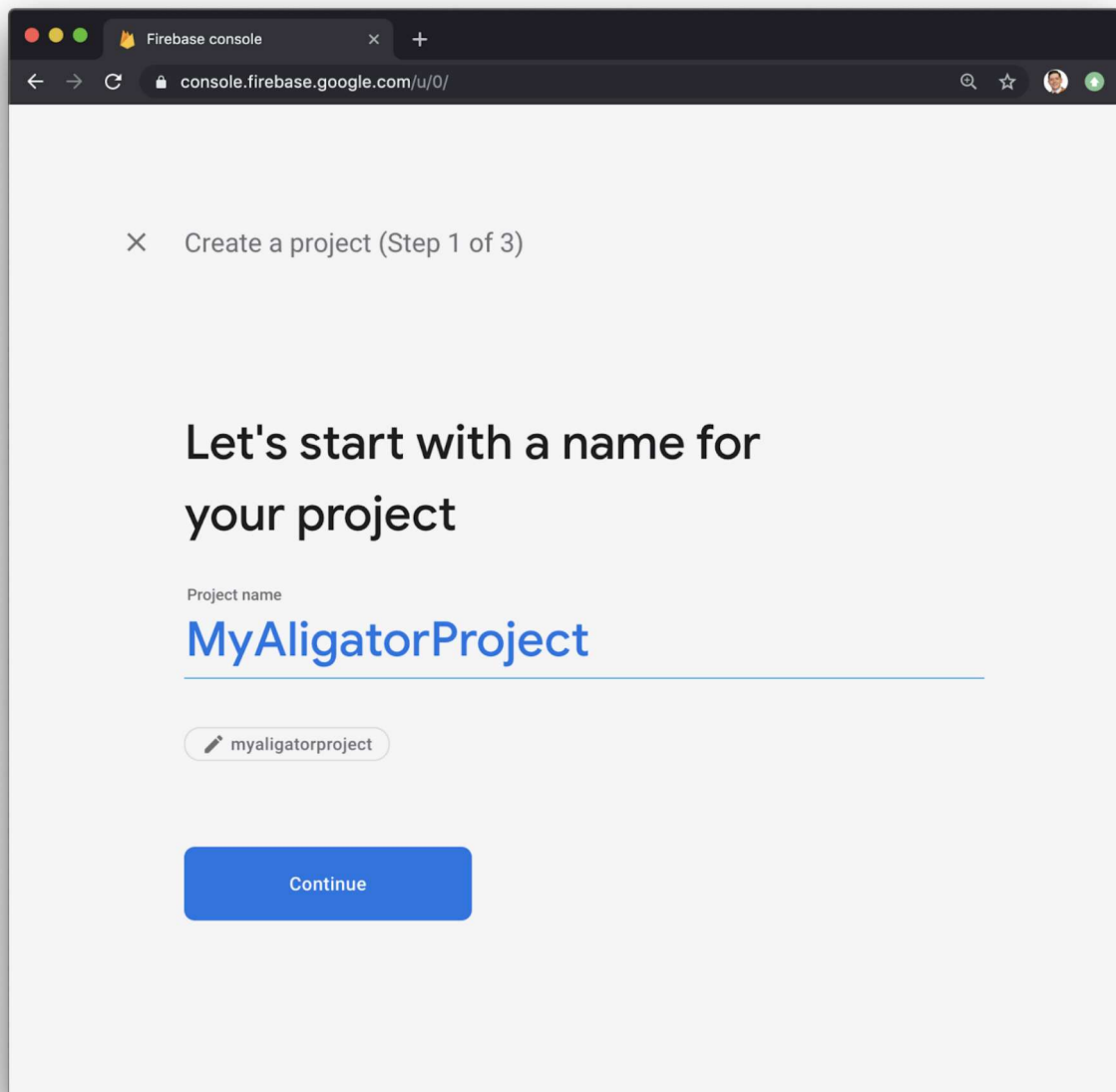
[Copy](#)

Using `flutter create` will produce a demo application that will display the number of times a button is clicked.

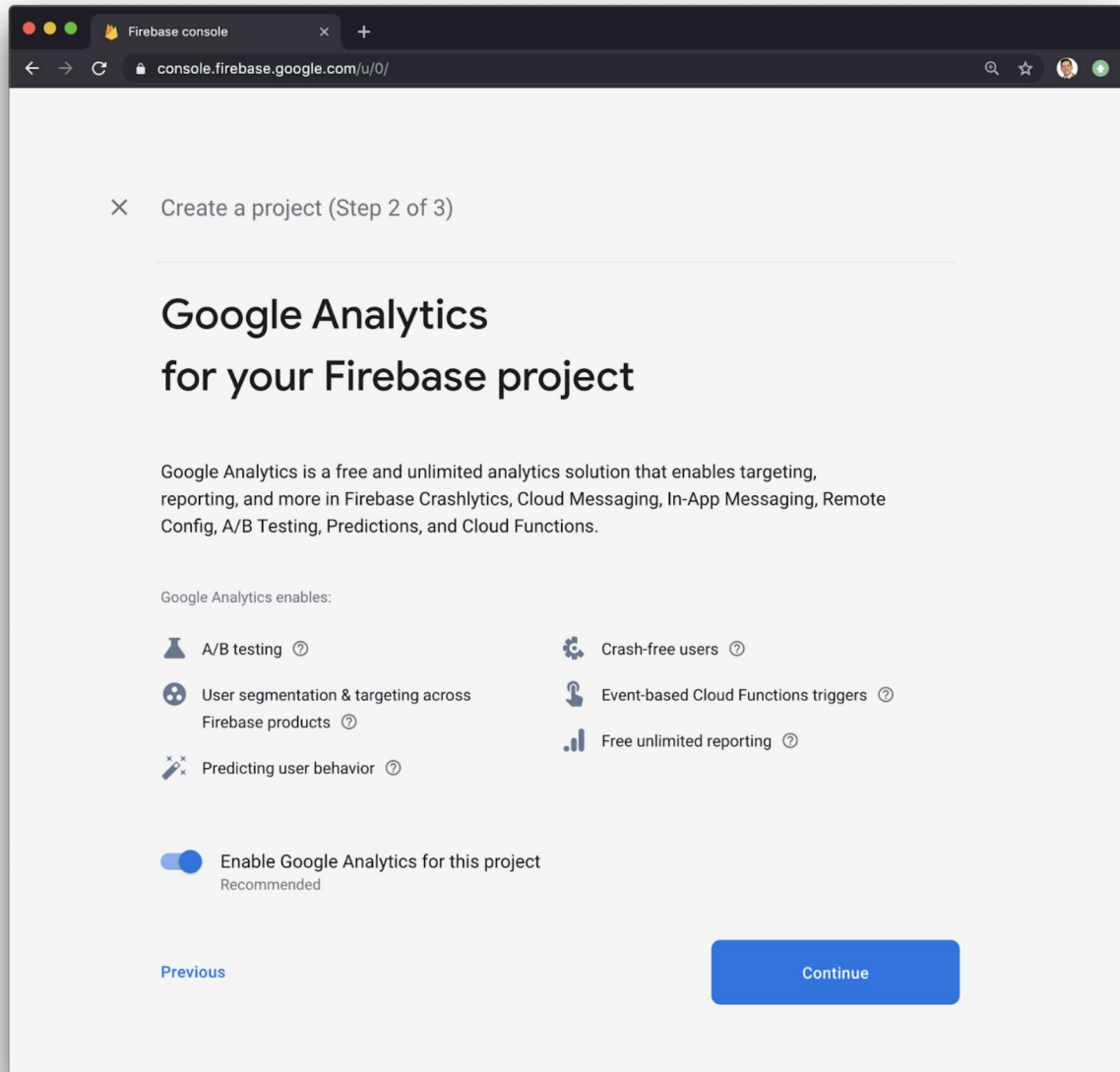
Now that we've got a Flutter project up and running, we can add Firebase.

## 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:

A screenshot of a web browser window showing the Firebase console. The browser's address bar displays 'console.firebase.google.com/u/0/'. The page title is 'Firebase console'. The main content area shows a dialog titled 'Create a project (Step 1 of 3)'. The dialog prompts the user to 'Let's start with a name for your project'. Below this, there is a text input field labeled 'Project name' containing the text 'MyAligatorProject'. Underneath the input field is a small button with a pencil icon and the text 'myaligatorproject'. At the bottom of the dialog is a large blue button labeled 'Continue'.

Next, we're given the option to enable Google Analytics. This tutorial will not require Google Analytics, but you can also choose to add it to your project.



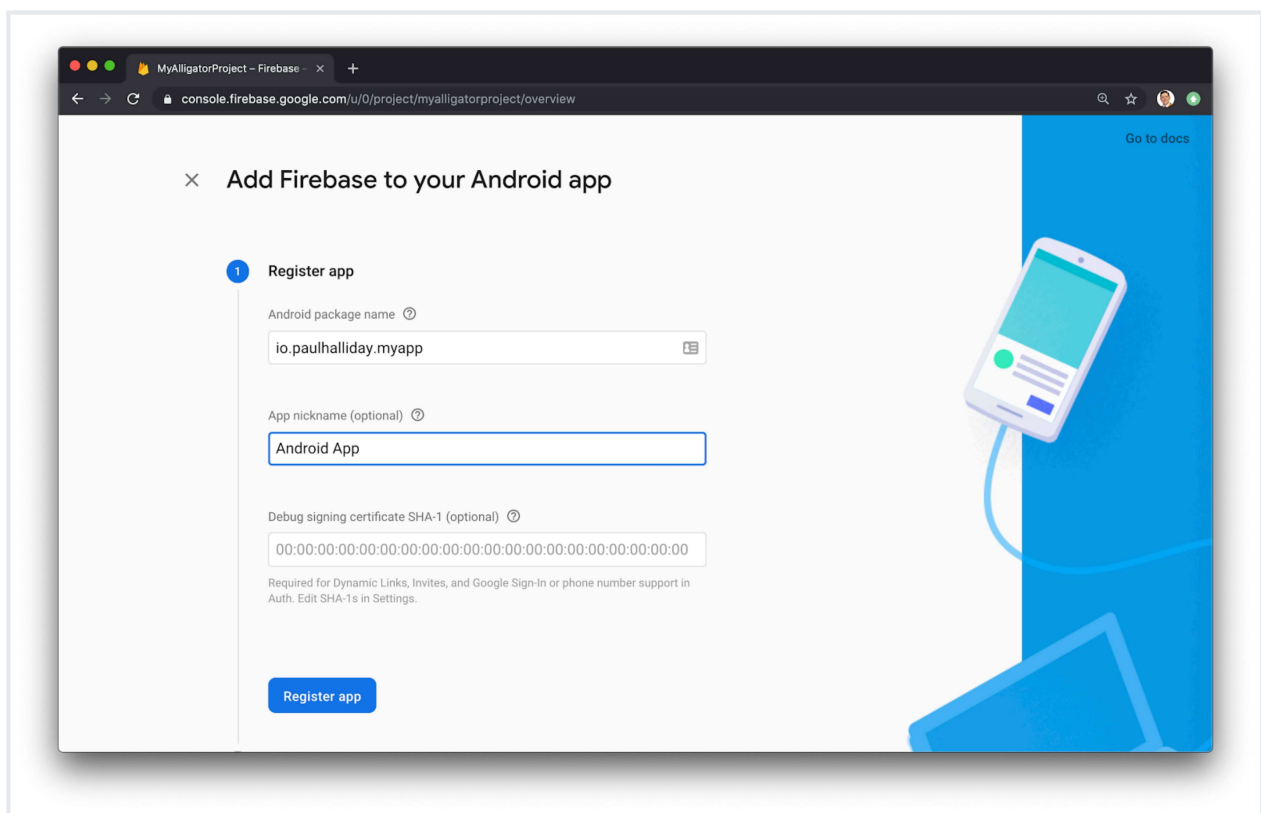
If you choose to use Google Analytics, you will need to review and accept the terms and conditions prior to project creation.

After pressing Continue, your project will be created and resources will be provisioned. You will then be directed to the dashboard for the new project.

## Adding Android support

### Registering the App

In order to add Android support to our Flutter application, select the Android logo from the dashboard. This brings us to the following screen:



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:

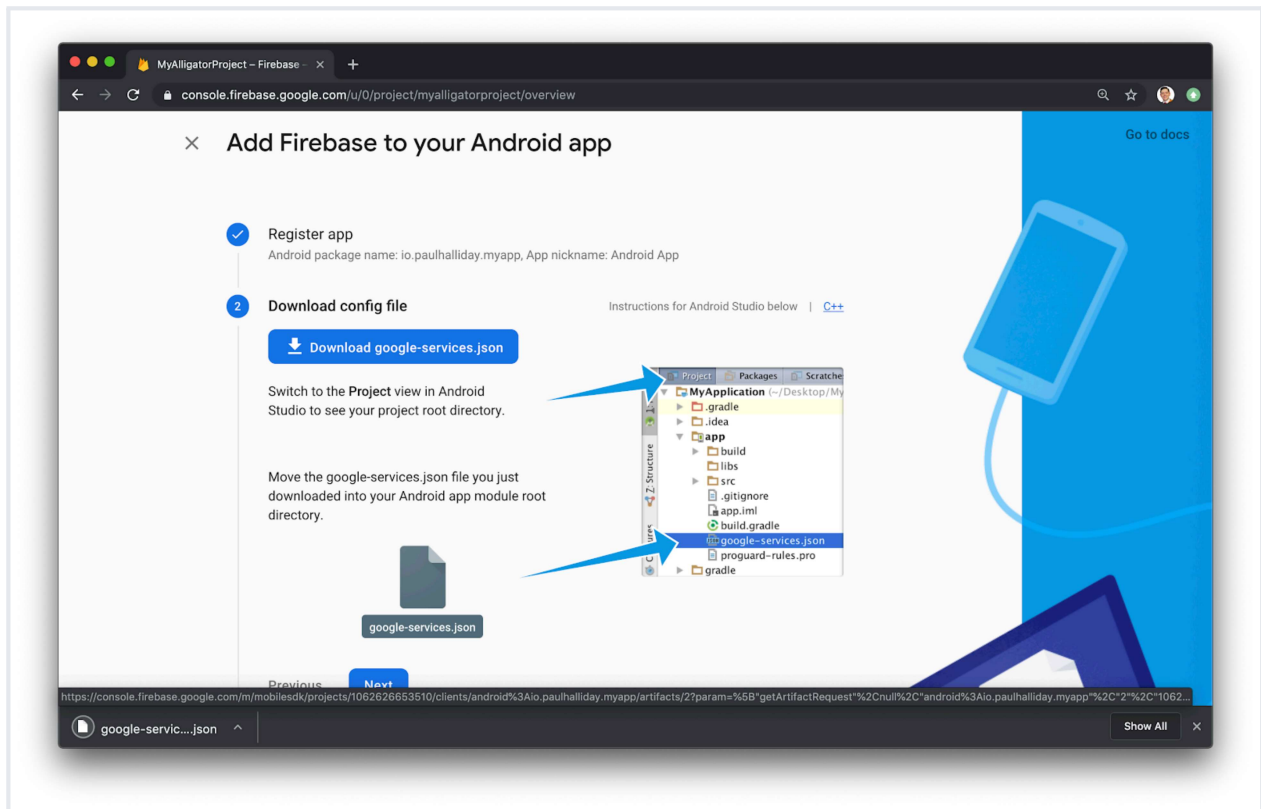
```
android/app/build.gradle
...
defaultConfig {
    // TODO: Specify your own unique Application ID
    (https://developer.android.com/studio/build/application-id.html).
    applicationId 'com.example.flutterfirebaseexample'
    ...
}
...
```

You can skip the app nickname and debug signing keys at this stage. Select Register app to continue.

## 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:



Next, move the `google-services.json` file to the `android/app` directory within the Flutter project.

## Adding the Firebase SDK

We'll now need to update our Gradle configuration to include the Google Services plugin.

Open `android/build.gradle` in your code editor and modify it to include the following:

```
android/build.gradle

buildscript {
  repositories {
    // Check that you have the following line (if not, add it):
    google() // Google's Maven repository
  }
  dependencies {
```

```

...
// Add this line
classpath 'com.google.gms:google-services:4.3.6'
}
}

allprojects {
...
repositories {
// Check that you have the following line (if not, add it):
google() // Google's Maven repository
...
}
}

```

Finally, update the app level file at `android/app/build.gradle` to include the following:

```

                                android/app/build.gradle
apply plugin: 'com.android.application'
// Add this line
apply plugin: 'com.google.gms:google-services'

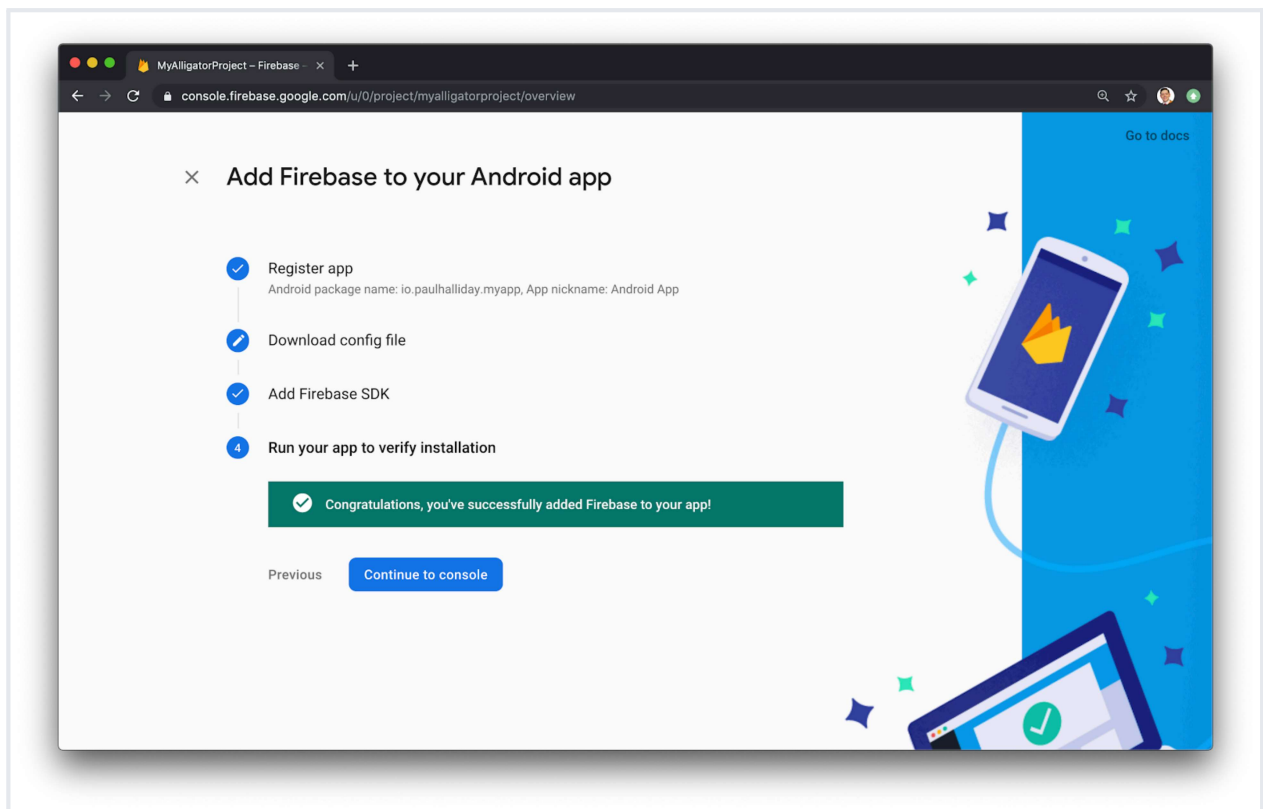
dependencies {
// Import the Firebase BoM
implementation platform('com.google.firebase:firebase-bom:28.0.0')

// Add the dependencies for any other desired Firebase products
// https://firebase.google.com/docs/android/setup#available-libraries
}

```

With this update, we're essentially applying the Google Services plugin as well as looking at how other Flutter Firebase plugins can be activated such as Analytics.

From here, run your application on an Android device or simulator. If everything has worked correctly, you should get the following message in the dashboard:



Next up, let's add iOS support!

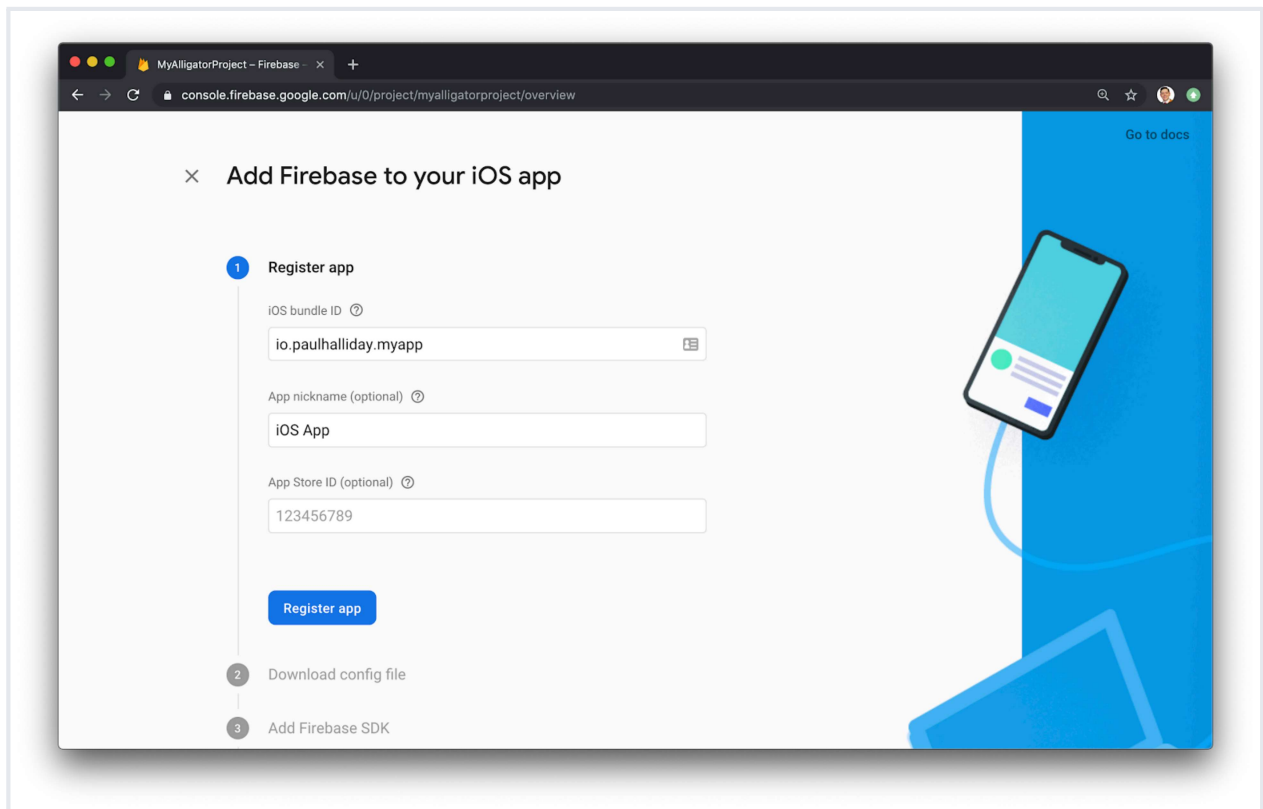
## Adding iOS Support

In order to add Firebase support for iOS, we have to follow a similar set of instructions.

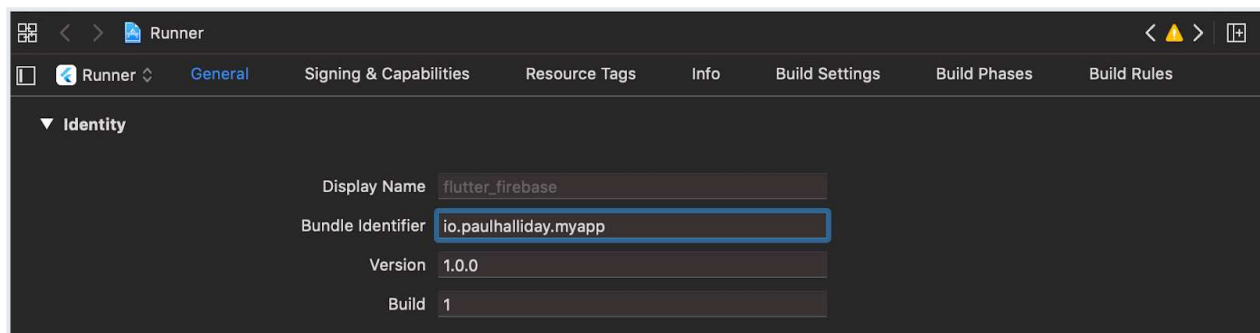
Head back over to the dashboard and select Add app and then iOS icon to be navigated to the setup process.

## Registering an App

Once again, we'll need to add an "iOS Bundle ID". It is possible to use the "Android package name" for consistency:



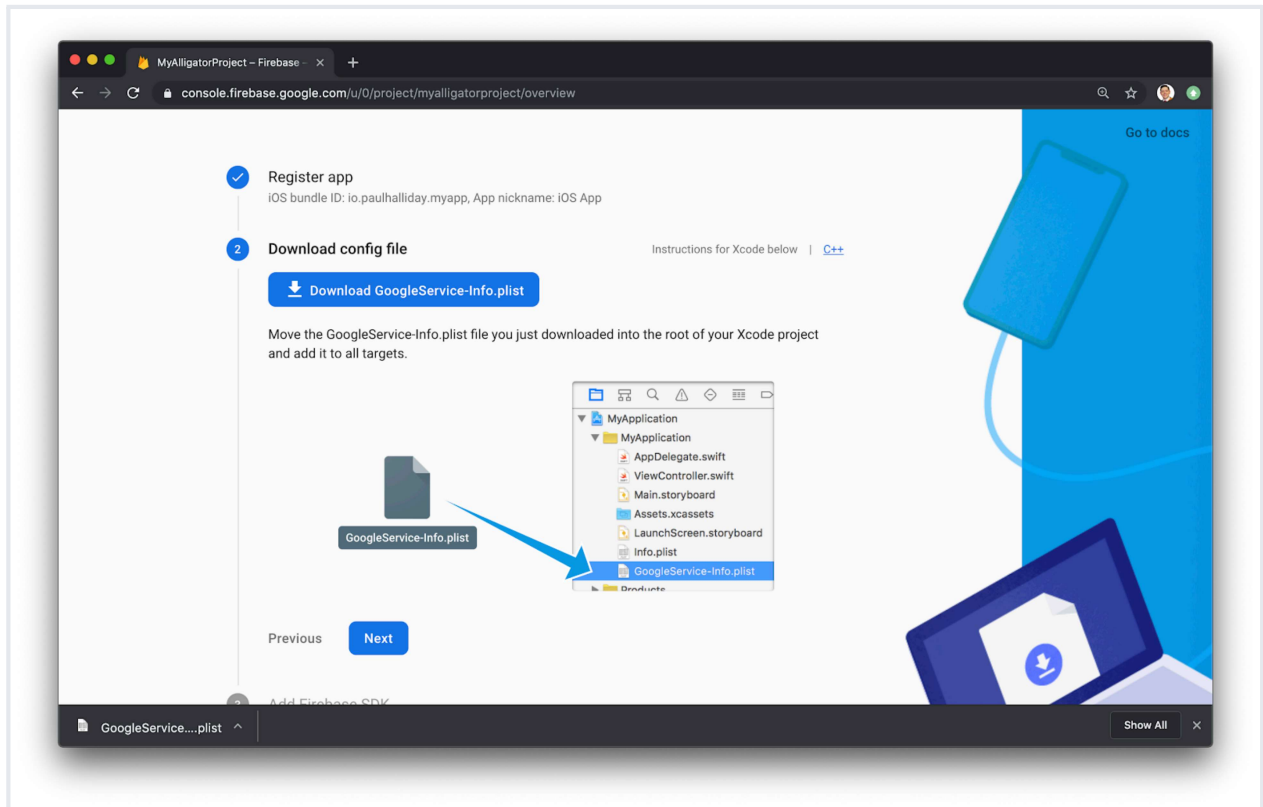
You'll then need to make sure this matches up by opening the iOS project up in Xcode at `ios/Runner/Runner.xcodeproj` and changing the Bundle identifier under General:



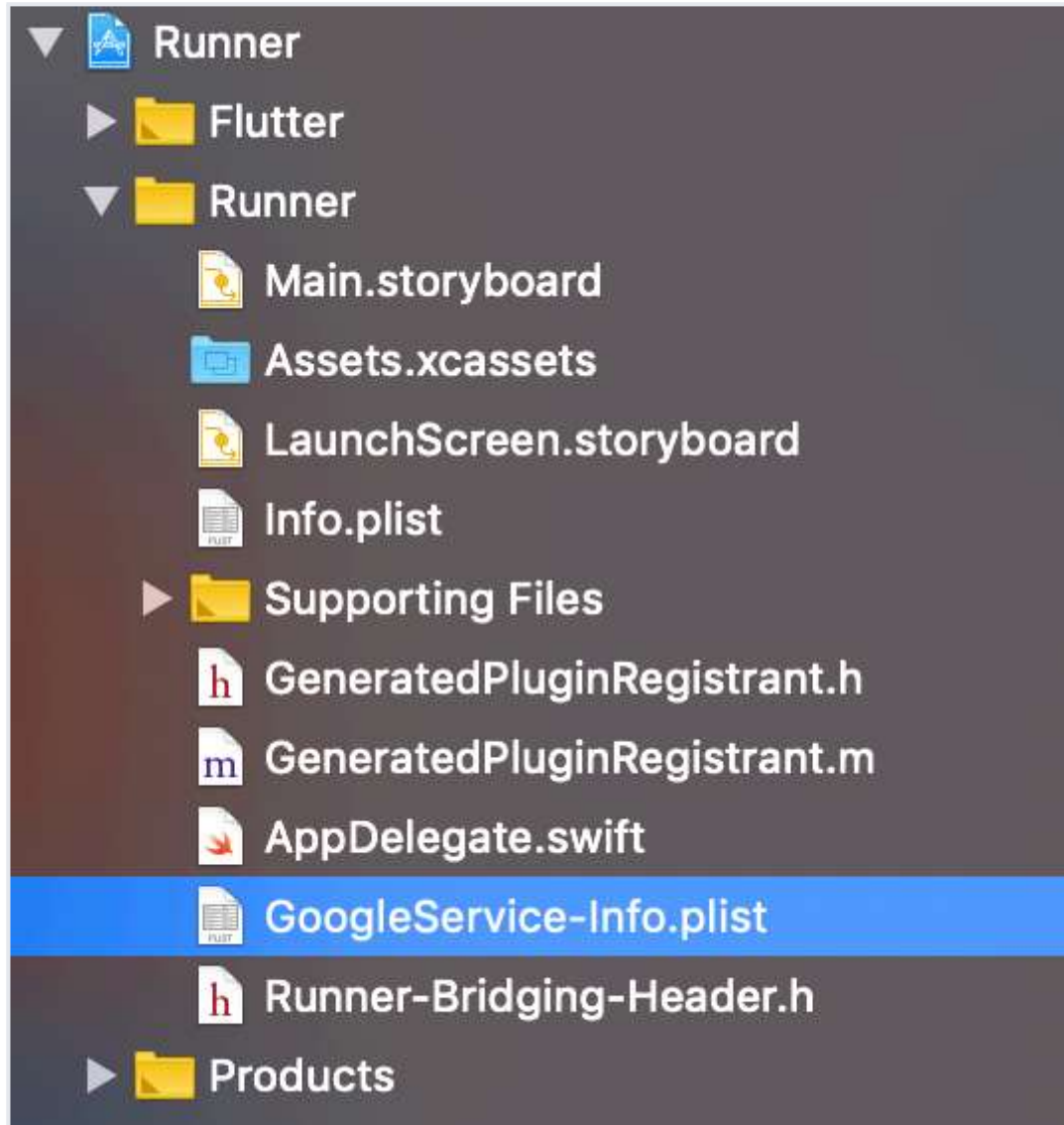
Click Register app to move to the next screen.

## Downloading the Config File

In this step, we'll need to download the configuration file and add this to our Xcode project.



Download `GoogleService-Info.plist` and move this into the root of your Xcode project within `Runner`:



Be sure to move this file within Xcode to create the proper file references.

There are additional steps for installing the Firebase SDK and adding initialization code, but they are not necessary for this tutorial.

That's it!

## Errors encountered

1. our `pubspec.yaml` file has a major issue:
  - You're using `firebase_core: ^3.11.0`, but your `environment` constraint is `sdk: ">=2.12.0 <3.0.0"`, which is outdated.
  - `firebase_core: ^3.11.0` does not exist; the latest stable version is `firebase_core: ^2.15.0`.
  - Your dependencies might be outdated and incompatible.

To correct this update your dependencies in `.yaml` file

2. Clean your project run

Flutter clean

Flutter pub get

Flutter run

## CODE :

```
// File generated by FlutterFire CLI.
// ignore for file: type=lint
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
import 'package:flutter/foundation.dart' show kIsWeb;

/// Default [FirebaseOptions] for use with your Firebase apps.
///
/// Example:
/// ```dart
/// import 'firebase_options.dart';
/// // ...
/// await Firebase.initializeApp(
///   options: DefaultFirebaseOptions.currentPlatform,
/// );
/// ```
class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    if (kIsWeb) {
      return web;
    }
    throw UnsupportedError(
```



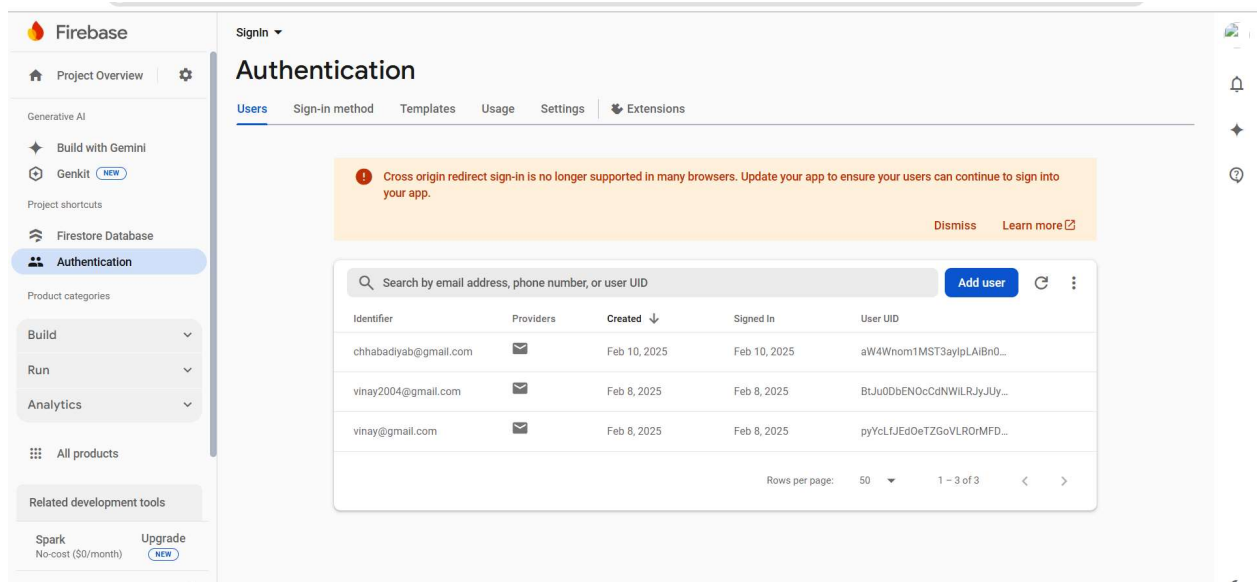
```

    'DefaultFirebaseOptions are only configured for the Web. '
    'Ensure you are running this on a supported platform.',
  );
}

static const FirebaseOptions web = FirebaseOptions(
  apiKey: 'AIzaSyAnkE7k_0ITmKzf0wmNRvEC_Mxp0dJlvos',
  appId: '1:805862802694:web:a28bd48a90d2ce650e4a87',
  messagingSenderId: '805862802694',
  projectId: 'signin-8c879',
  authDomain: 'signin-8c879.firebaseio.com',
  storageBucket: 'signin-8c879.appspot.com',
  measurementId: 'G-E75S3K7S2Z',
);
}

```

## OUTPUT:



The screenshot shows the Firebase Authentication console. On the left is a sidebar with navigation options: Project Overview, Generative AI (Build with Gemini, Genkit), Project shortcuts (Firestore Database, Authentication), Product categories (Build, Run, Analytics), All products, and Related development tools (Spark, Upgrade). The main area is titled 'Authentication' and has tabs for Users, Sign-in method, Templates, Usage, Settings, and Extensions. A warning message states: 'Cross origin redirect sign-in is no longer supported in many browsers. Update your app to ensure your users can continue to sign into your app.' Below this is a table of users with columns: Identifier, Providers, Created, Signed In, and User UID. The table contains three users, all created on Feb 8, 2025. At the bottom right, it shows 'Rows per page: 50' and '1 - 3 of 3'.

Identifier	Providers	Created	Signed In	User UID
chhabadiyab@gmail.com		Feb 10, 2025	Feb 10, 2025	aW4Wnom1MST3aylpLAIbn0...
vinay2004@gmail.com		Feb 8, 2025	Feb 8, 2025	BtJu0DbENocCdNWILRjyJuy...
vinay@gmail.com		Feb 8, 2025	Feb 8, 2025	pyYcLfJEdOeTZGoVLR0rMFD...

## Conclusion

In this article, you learned how to set up and ready our Flutter applications to be used with Firebase.

Flutter has official support for Firebase with the FlutterFire set of libraries.