

# Experiment No: 06

## How To Set Up Firebase with Flutter for Android Apps

Firebase is a great backend solution for anyone that wants to use authentication, databases, cloud functions, ads, and countless other features within an app.

In this article, you will create a Firebase project for iOS and Android platforms using Flutter.

### Prerequisites

To complete this tutorial, you will need:

- A Google account to use Firebase.
- Developing for iOS will require XCode.
- To download and install Flutter.
- To download and install Android Studio and Visual Studio Code.
- It is recommended to install plugins for your code editor:
  - `Flutter` and `Dart` plugins installed for Android Studio.
  - `Flutter` extension installed for Visual Studio Code.

## 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
```

Navigate to the new project directory:

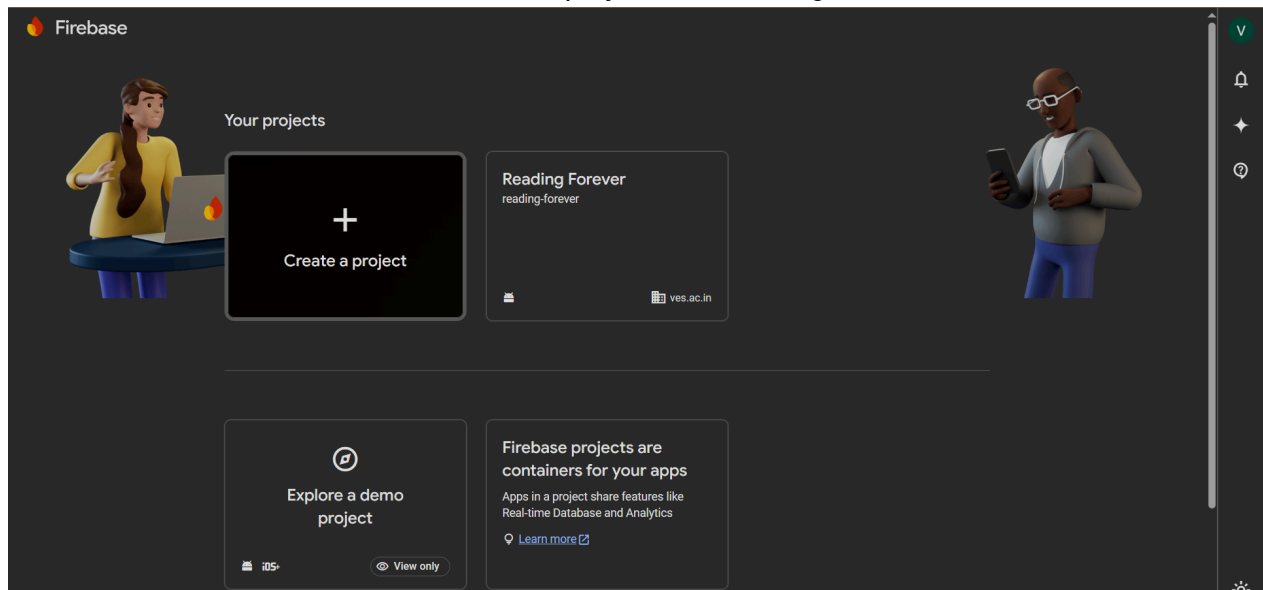
```
cd flutterfirebaseexample
```

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

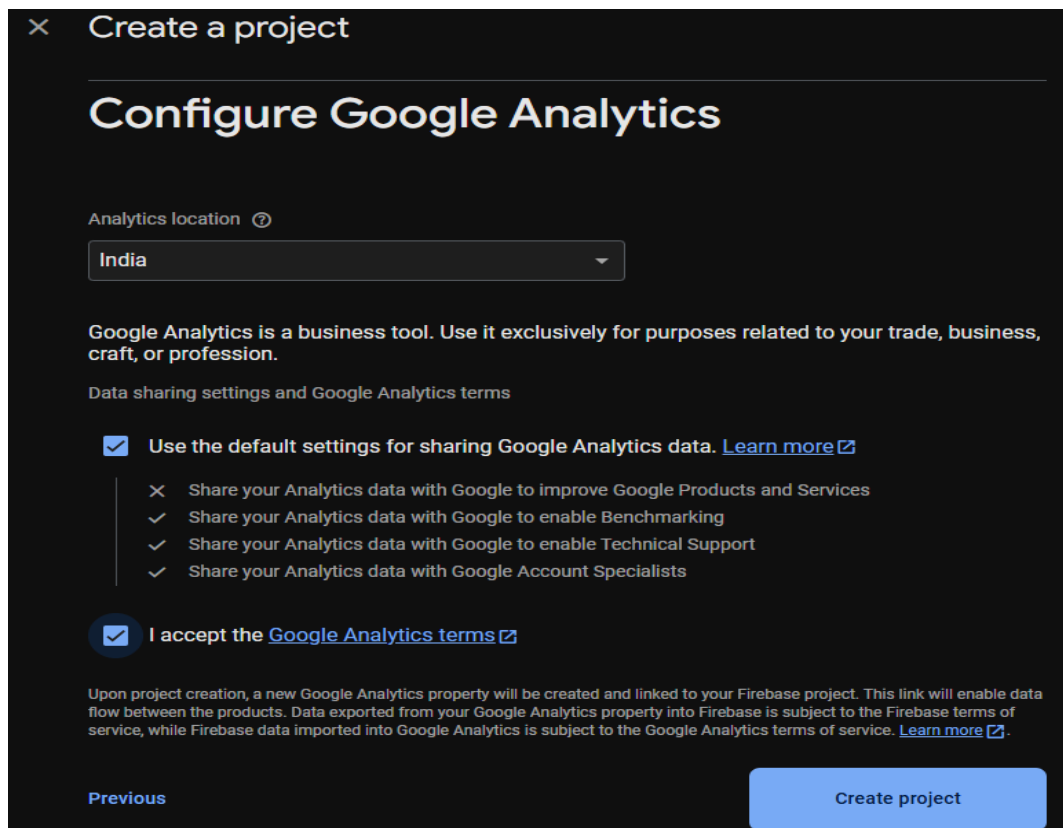
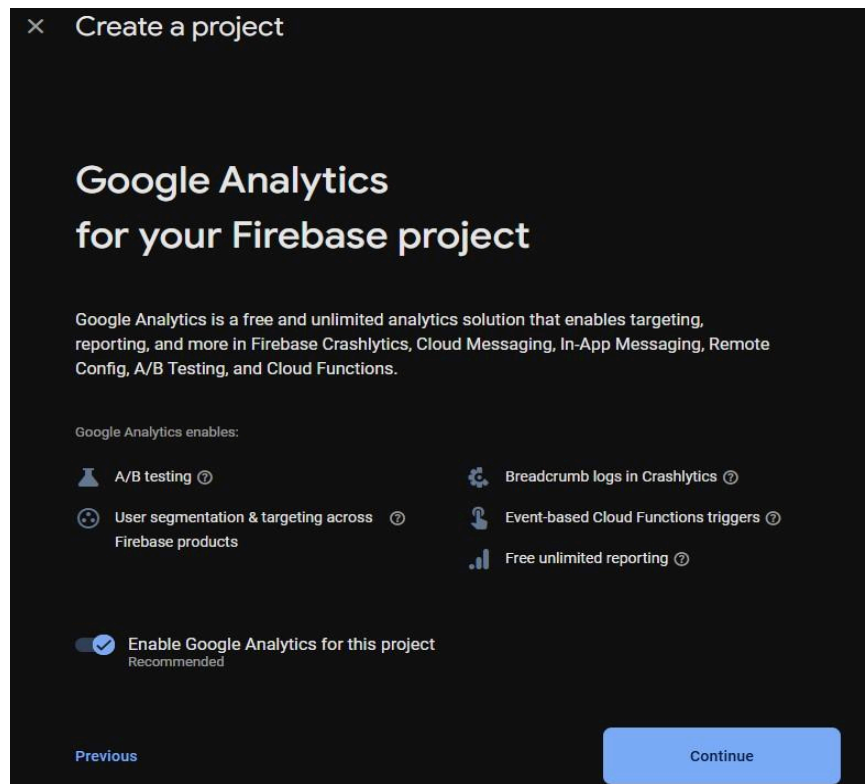
Now that we've got a Flutter project up and running, we can add Firebase.

## 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:



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.



## 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:

Add Firebase to your Android app

1 Register app

Android package name ?  
com.yourname.readingforever

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:f

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

Register app

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

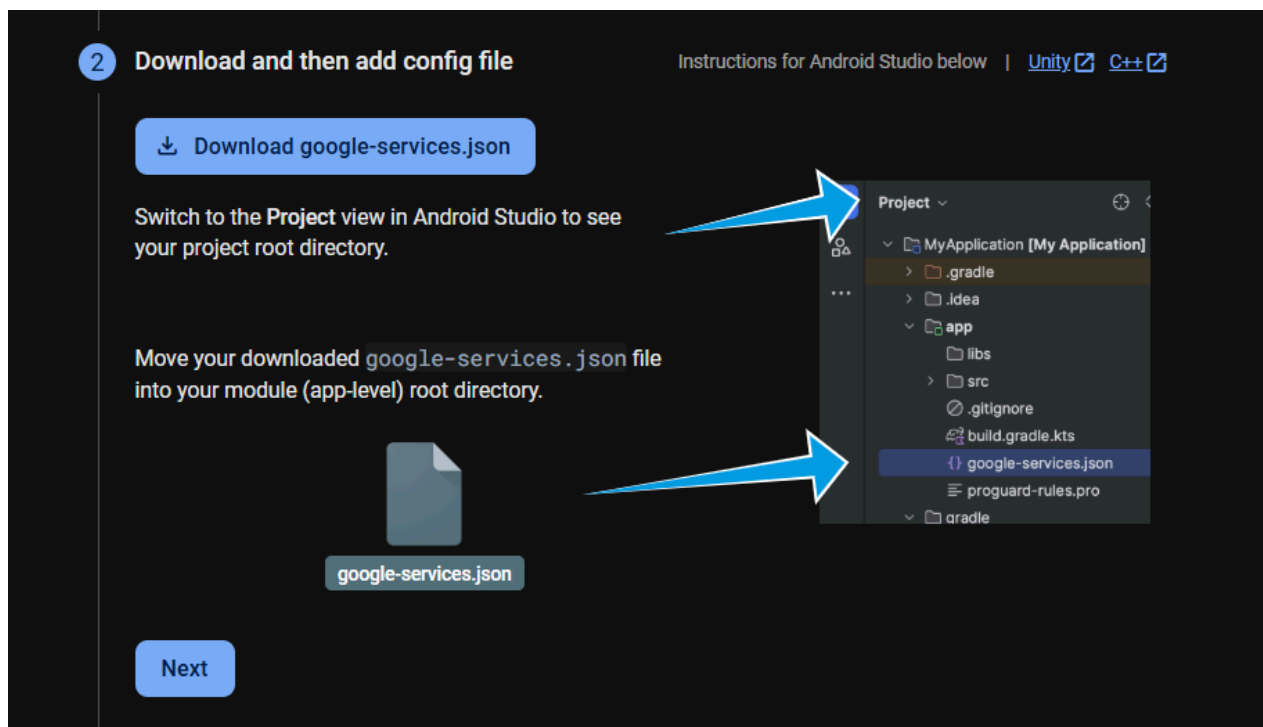
...
defaultConfig {
    // TODO: Specify your own unique Application ID
    (https://developer.android.com/studio/build/application-id.html)
    .  applicationId 'com.example.flutterfirebaseexample'
    ...
}
...
```

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:

★ 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` file:

Root-level (project-level) Gradle file (`<project>/build.gradle`):

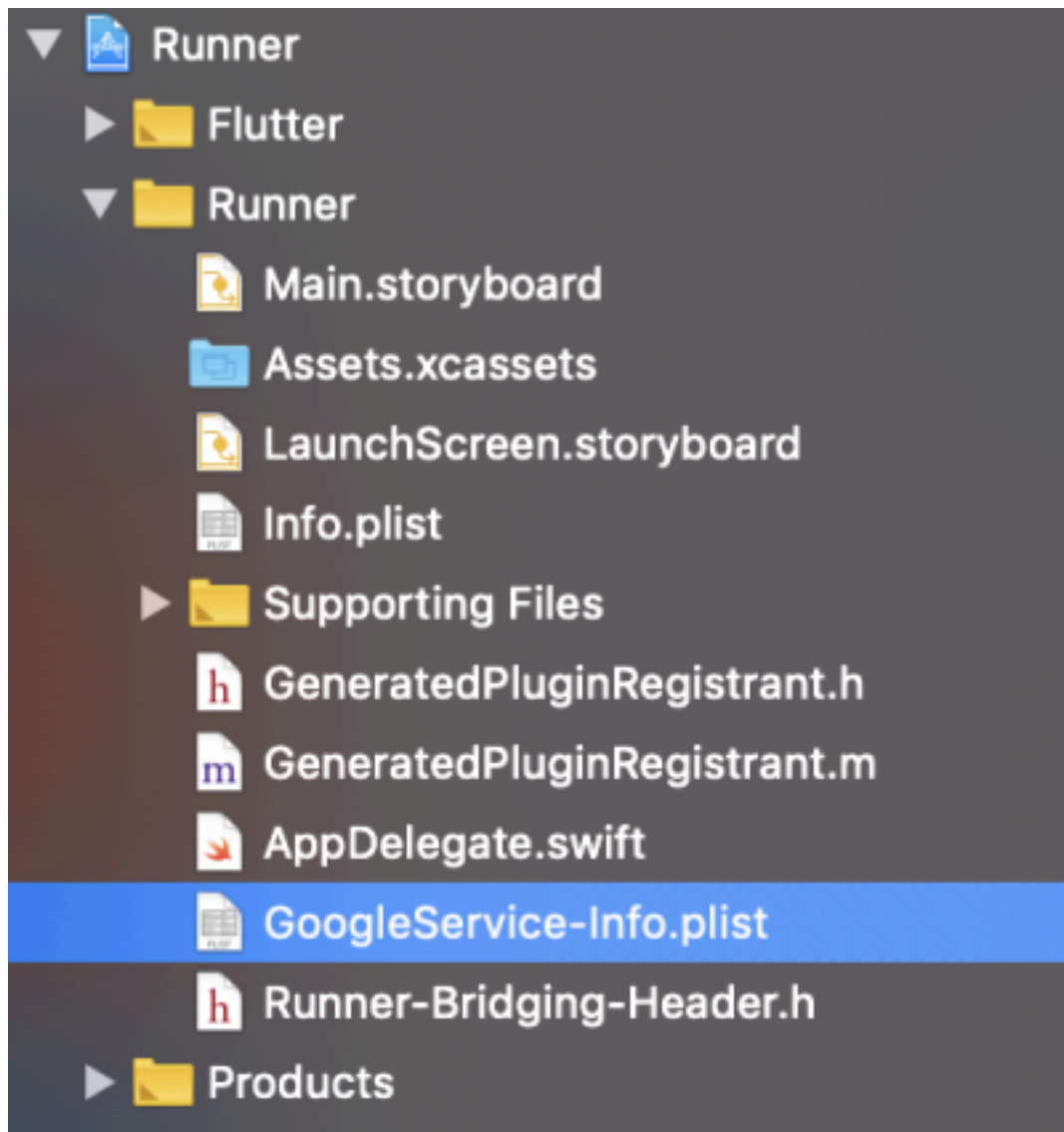
```
plugins {  
    // ...  
  
    // Add the dependency for the Google services Gradle plugin  
    id 'com.google.gms.google-services' version '4.4.2' apply false  
}
```

2. Then, in your module (app-level) `build.gradle` 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`):

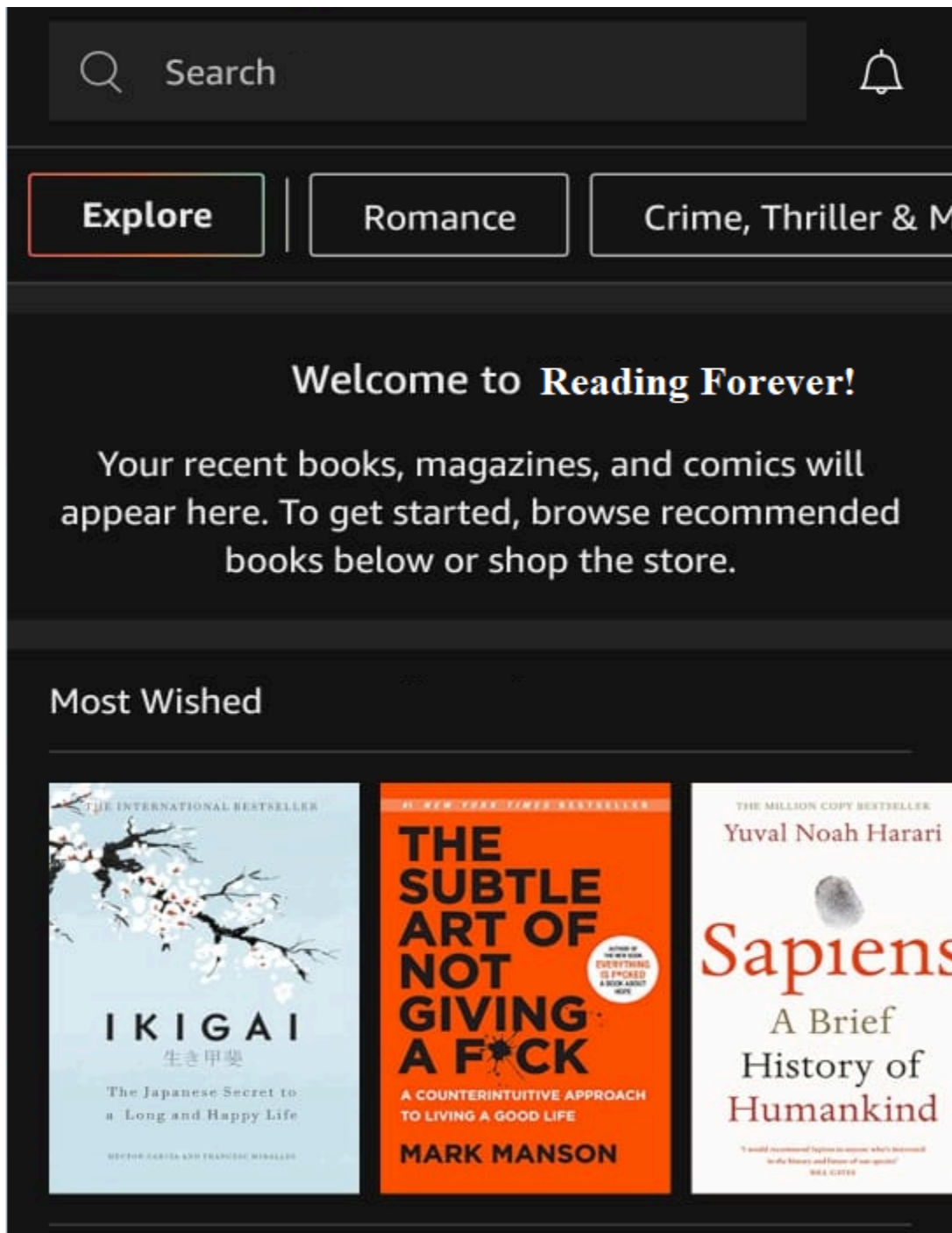
```
plugins {  
    id 'com.android.application'  
    // Add the Google services Gradle plugin  
    id 'com.google.gms.google-services'  
    ...  
}  
  
dependencies {  
    // Import the Firebase BoM  
    implementation platform('com.google.firebase:firebase-bom:33.10.0')  
  
    // TODO: Add 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  
}
```





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.

## After Setup



## 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.