Name:Girish Chougule
Div:D15B
Roll No: 10

MPL Experiment 6

Aim:

To integrate Firebase with a Flutter application for both Android and iOS platforms, enabling backend services such as authentication, cloud storage, and real-time database functionality.

Theory:

Firebase is a Backend-as-a-Service (BaaS) platform by Google that provides cloud-based solutions like authentication, real-time database, cloud storage, and push notifications. Flutter allows easy integration with Firebase using the `firebase_core` package.

Key Concepts:

1. **Firebase Project Setup:**
    - A Firebase project needs to be created in the Firebase Console.
    - Firebase services require specific configuration files (`google-services.json` for Android and `GoogleService-Info.plist` for iOS).
2. **Adding Firebase to Flutter:**
    - The `firebase_core` package is required for initializing Firebase in a Flutter app.
    - Additional Firebase services (e.g., authentication, Firestore) require separate dependencies in `pubspec.yaml`.
3. **Platform-Specific Configurations:**
    - **Android:** Requires modifications in `android/app/build.gradle` and `android/build.gradle` to integrate Firebase.
    - **iOS:** Needs configuration in Xcode and inclusion of Firebase dependencies via CocoaPods.
4. **Verifying Firebase Integration:**

- Running the Flutter app on a physical or virtual device.
- Checking Firebase Console to confirm successful connection.

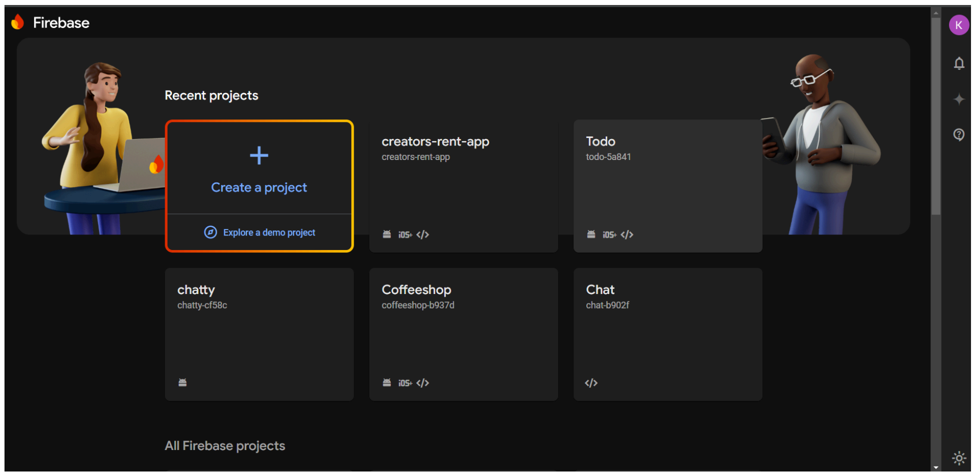
Step-by-Step Guide to Setting Up Firebase with Flutter (iOS & Android)

Step 1: Prerequisites

Ensure you have the following:

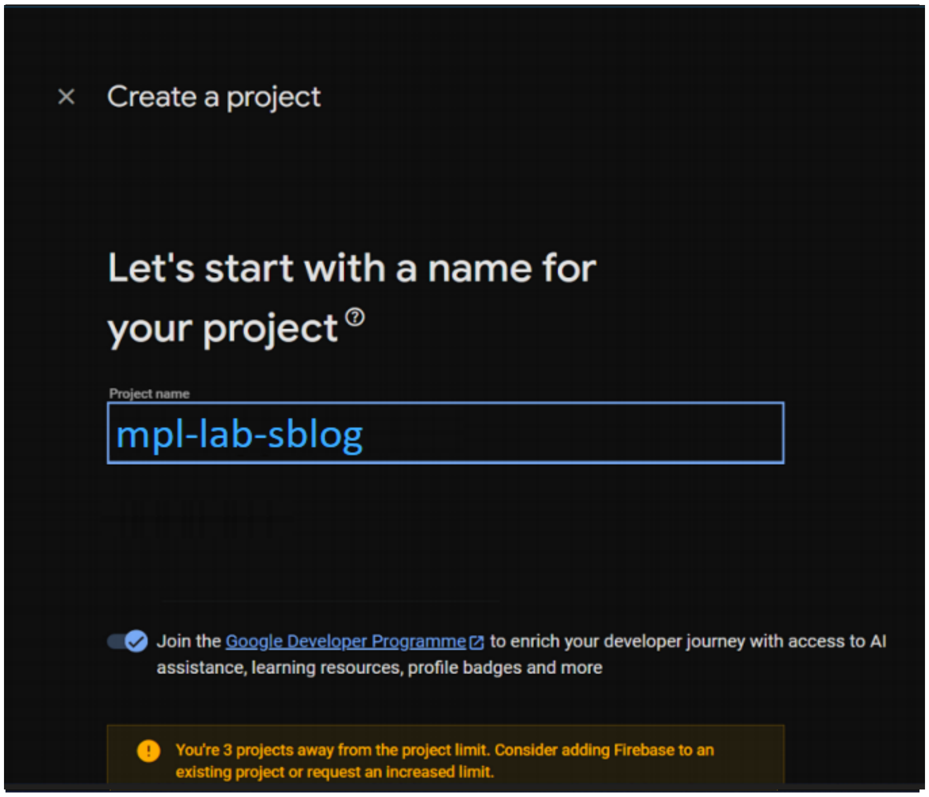
- A **Google account** to access Firebase.
- **Flutter installed** on your system.
- **Android Studio** and **Visual Studio Code** installed.
- **Xcode installed** (for iOS development).
- **Flutter and Dart plugins** installed in Android Studio.
- **Flutter extension** installed in Visual Studio Code.

Step 2: Create a New Flutter Project

- Open a terminal and create a new Flutter project.
- Navigate to the project directory.
- 

Step 3: Create a Firebase Project

1. Go to the Firebase Console.
2. Click **"Create a project"** and provide a project name.
3. Choose whether to enable **Google Analytics** (optional).
4. Click **"Continue"** and wait for Firebase to set up the project.



Create a project

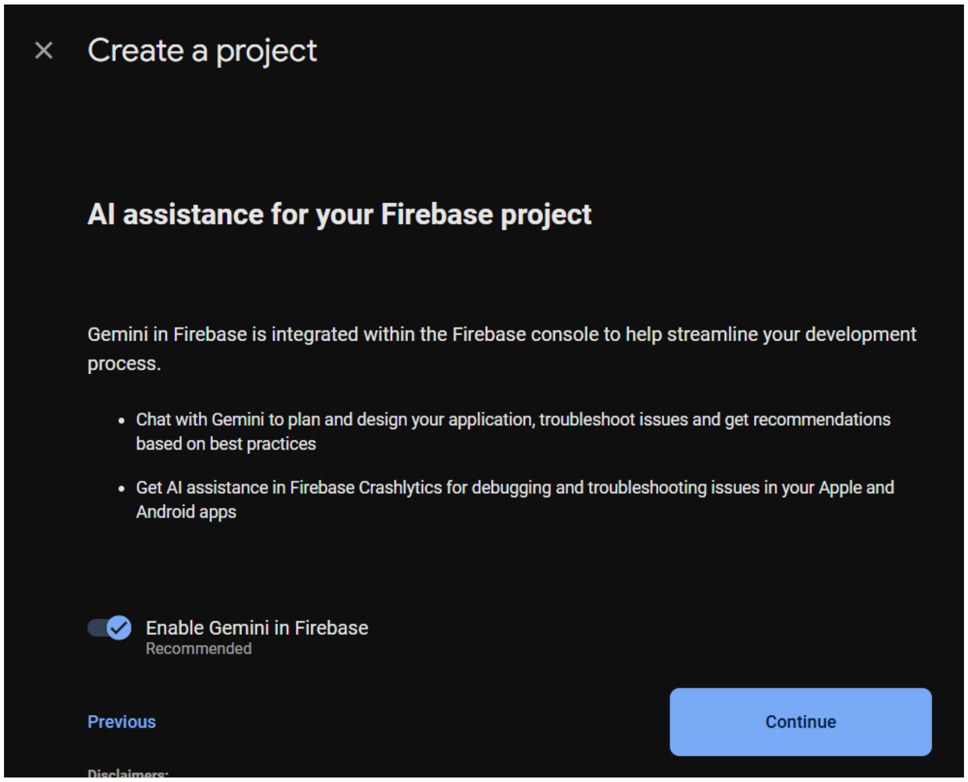
Let's start with a name for your project

Project name

mpl-lab-sblog

Join the Google Developer Programme to enrich your developer journey with access to AI assistance, learning resources, profile badges and more

You're 3 projects away from the project limit. Consider adding Firebase to an existing project or request an increased limit.



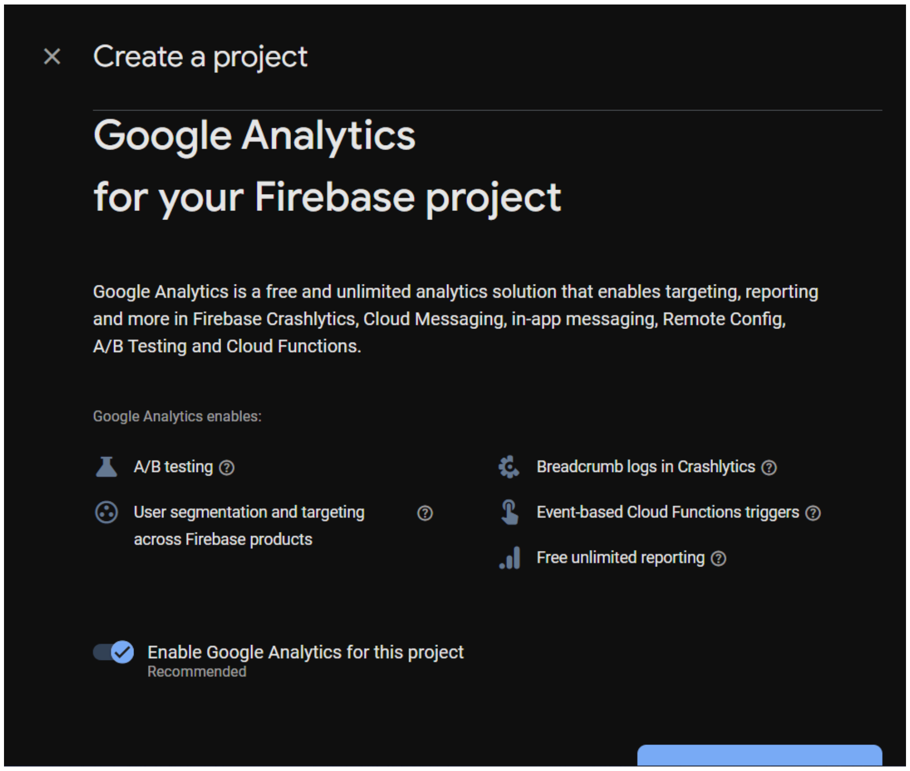
Create a project

AI assistance for your Firebase project

Gemini in Firebase is integrated within the Firebase console to help streamline your development process.

- Chat with Gemini to plan and design your application, troubleshoot issues and get recommendations based on best practices
- Get AI assistance in Firebase Crashlytics for debugging and troubleshooting issues in your Apple and Android apps

Enable Gemini in Firebase
Recommended



Create a project

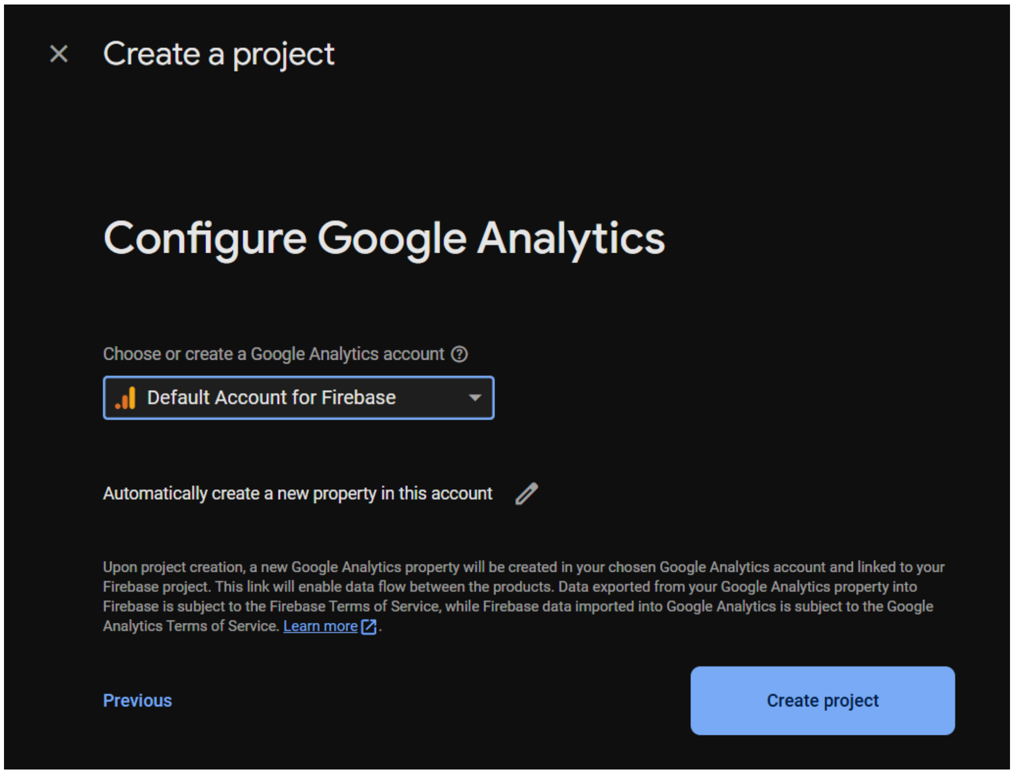
Google Analytics
for your Firebase project

Google Analytics is a free and unlimited analytics solution that enables targeting, reporting and more in Firebase Crashlytics, Cloud Messaging, in-app messaging, Remote Config, A/B Testing and Cloud Functions.

Google Analytics enables:

- A/B testing
- User segmentation and targeting across Firebase products
- Breadcrumb logs in Crashlytics
- Event-based Cloud Functions triggers
- Free unlimited reporting

Enable Google Analytics for this project ✓
Recommended



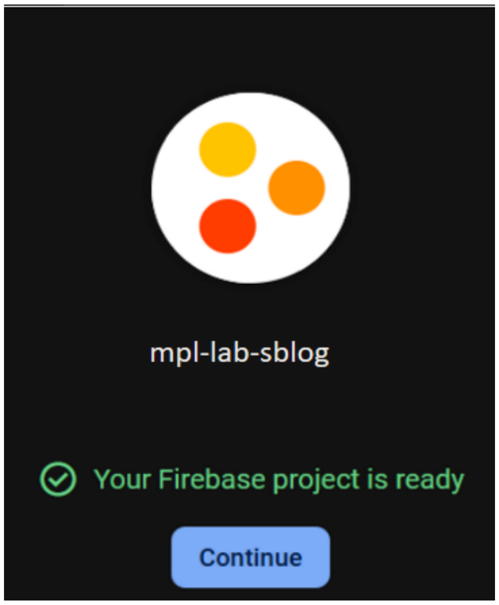
Create a project

Configure Google Analytics

Choose or create a Google Analytics account
Default Account for Firebase

Automatically create a new property in this account

Upon project creation, a new Google Analytics property will be created in your chosen Google Analytics account and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase Terms of Service, while Firebase data imported into Google Analytics is subject to the Google Analytics Terms of Service. Learn more.

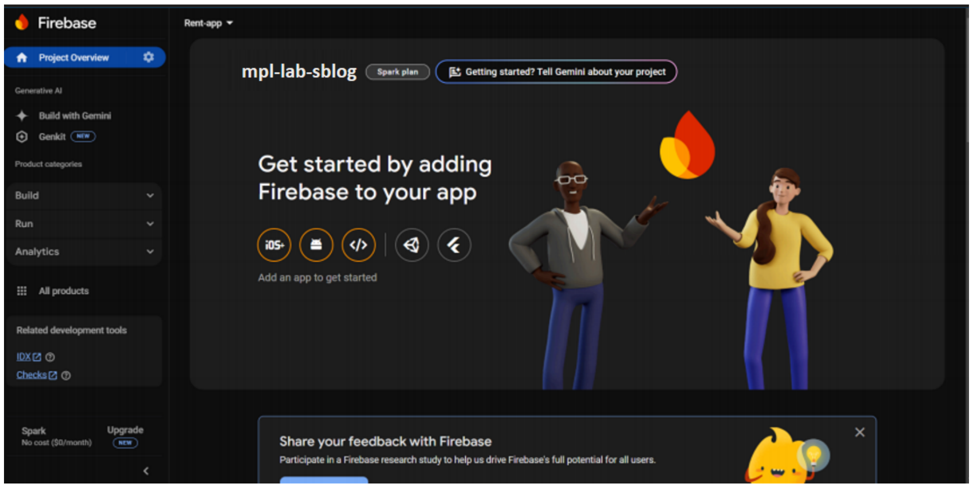


Step 4: Add Firebase to Android

1. In Firebase, select **Android** and register your app.
2. Enter the **Android package name** (must match the one in your app).
3. Download the **google-services.json** file from Firebase.
4. Move the file to the appropriate location in your Flutter project.
5. Update the project's build configuration files to include Firebase dependencies.
6. Run the Flutter app on an Android device or emulator to verify the setup.

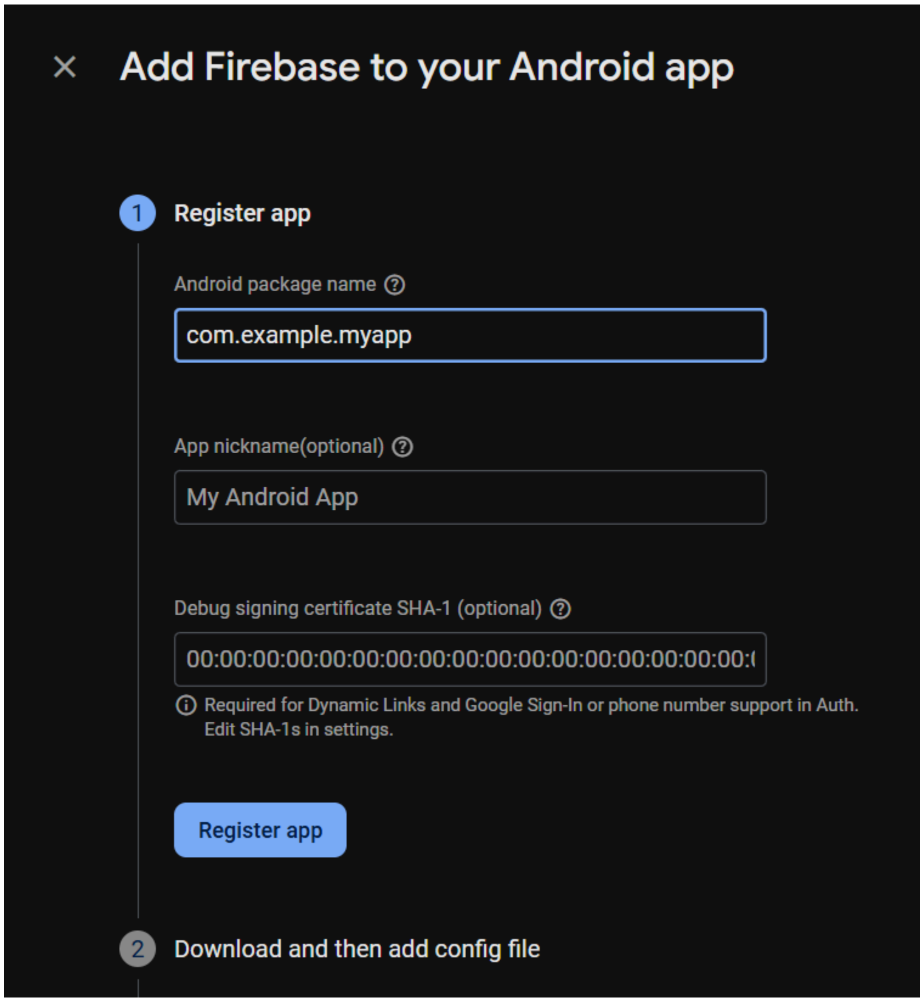
Step 5: Add Firebase to iOS

1. In Firebase, select **iOS** and register your app.
2. Enter the **iOS Bundle ID** (should match the one in your project).
3. Open the iOS project in Xcode and update the **Bundle Identifier**.
4. Download the **GoogleService-Info.plist** file from Firebase.
5. Move the file into the correct directory inside your Xcode project.
6. Ensure Firebase is initialized properly for iOS.



Share your feedback with Firebase

Participate in a Firebase research study to help us drive Firebase's full potential for all users.



Add Firebase to your Android app

1 Register app

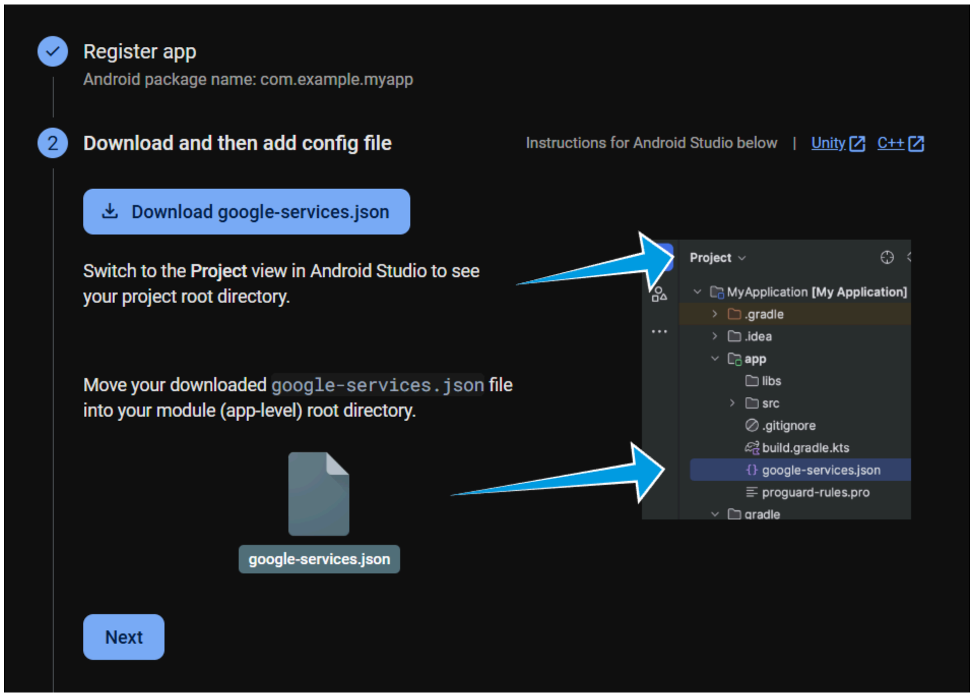
Android package name: com.example.myapp

App nickname(optional): My Android App

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

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

2 Download and then add config file


Register app

Android package name: com.example.myapp

2 Download and then add config file

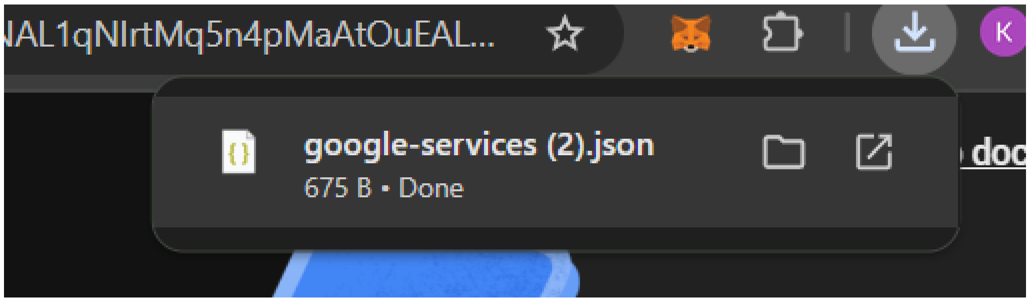
Instructions for Android Studio below | Unity C++

Download google-services.json

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

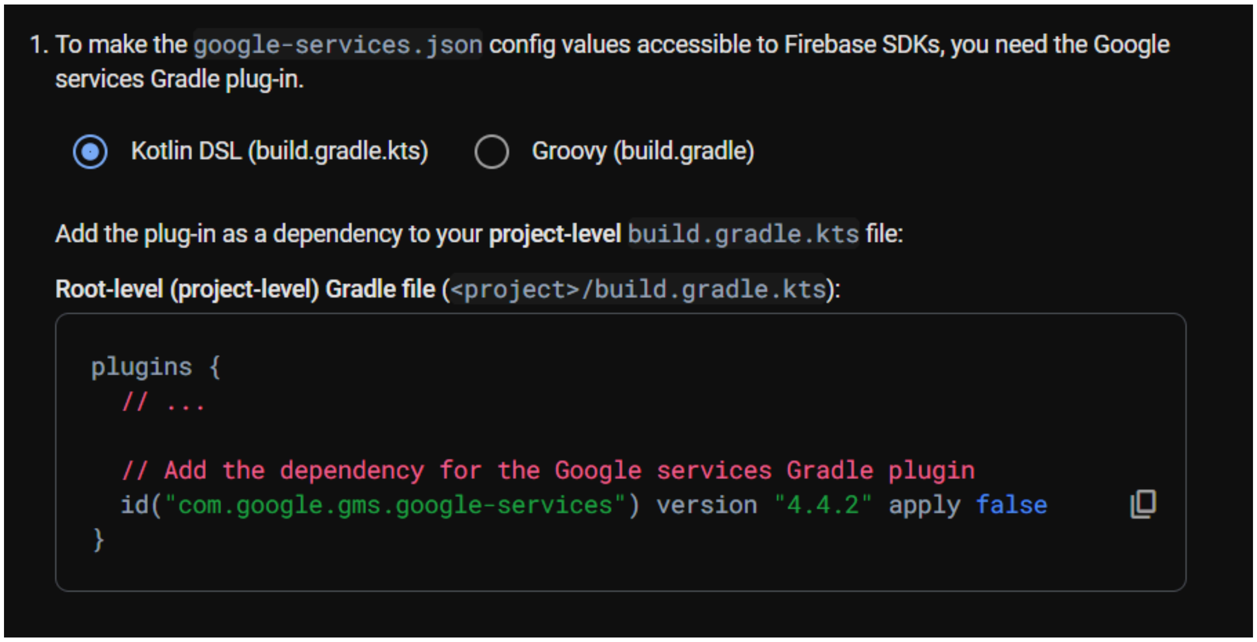
Move your downloaded `google-services.json` file into your module (app-level) root directory.

Next



google-services (2).json

675 B • Done


1. To make the `google-services.json` config values accessible to Firebase SDKs, you need the Google services Gradle plug-in.

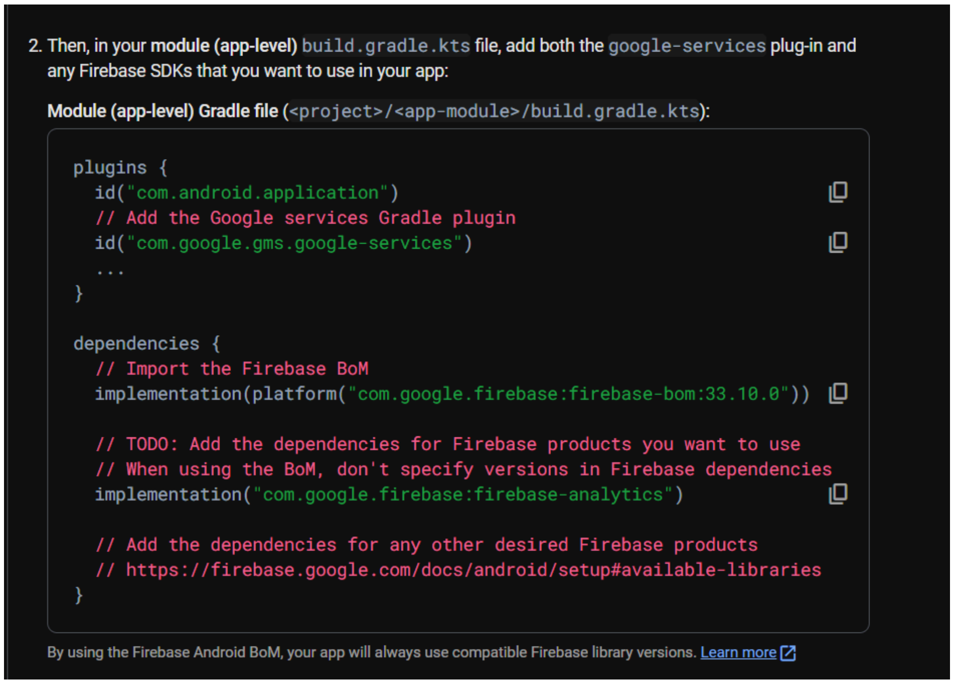
[✓ Kotlin DSL (build.gradle.kts)] Groovy (build.gradle)

Add the plug-in 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 "4.4.2" apply false
}
```



2. Then, in your **module (app-level)** `build.gradle.kts` file, add both the `google-services` plug-in and any Firebase SDKs that you want to use in your app:

Module (app-level) Gradle file (`<project>/<app-module>/build.gradle.kts`):

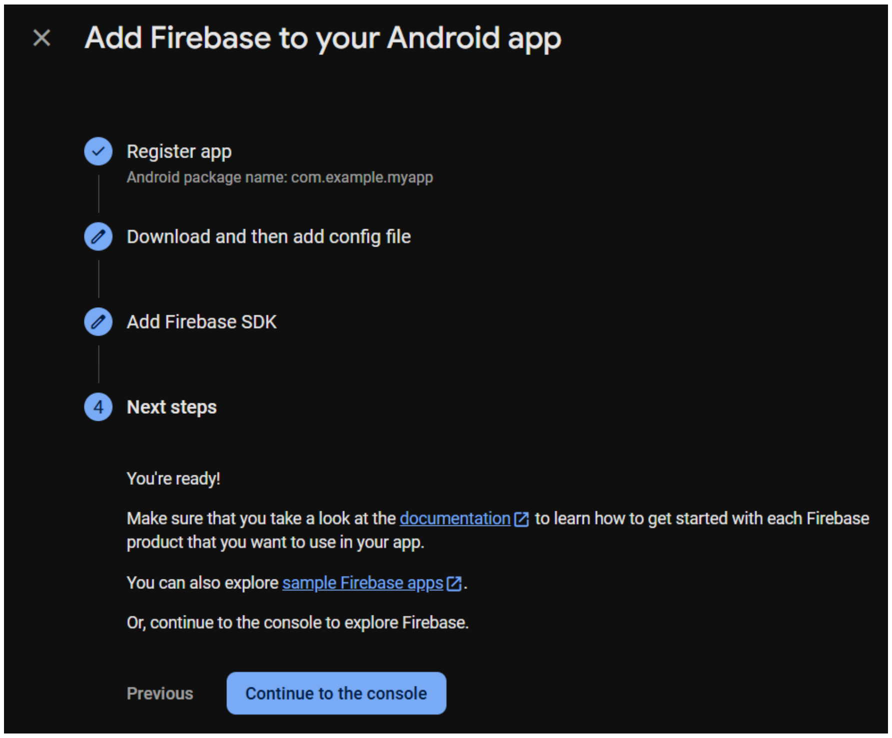
```
plugins {
  id("com.android.application")
  // Add the Google services Gradle plugin
  id("com.google.gms.google-services")
  ...
}

dependencies {
  // Import the Firebase BoM
  implementation(platform("com.google.firebase:firebase-bom: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
}
```

By using the Firebase Android BoM, your app will always use compatible Firebase library versions. Learn more



Add Firebase to your Android app

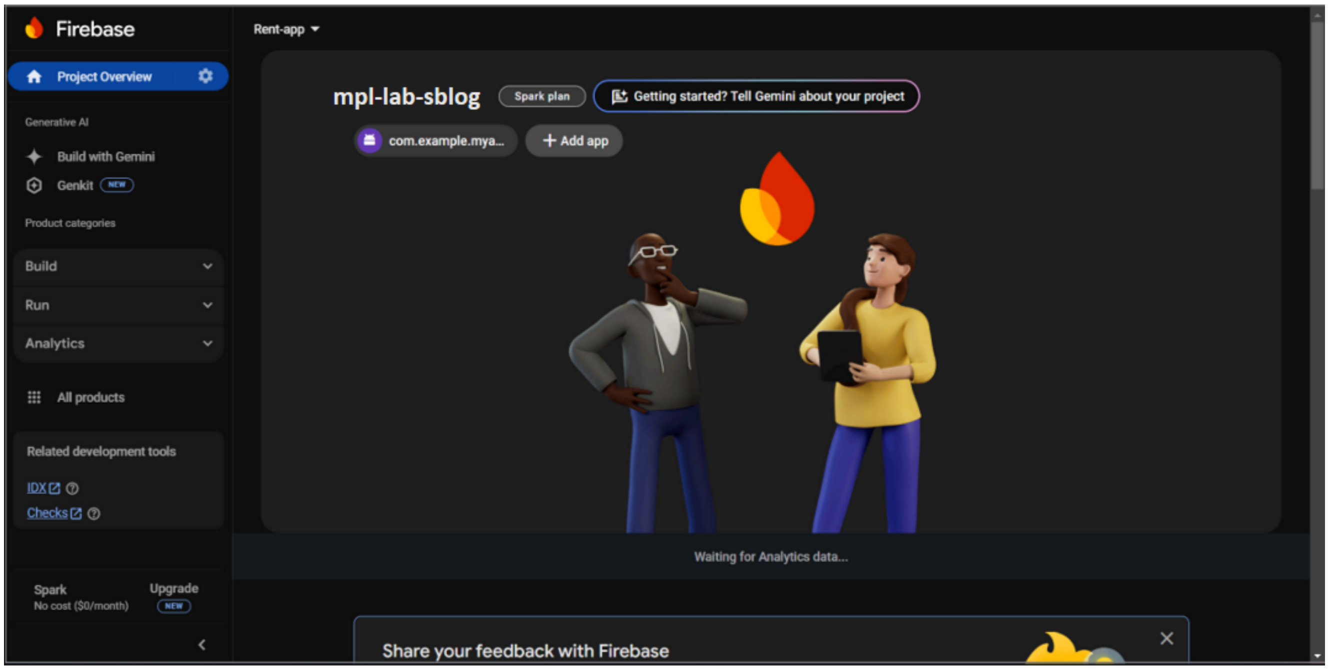
- ✓ Register app
  Android package name: com.example.myapp
- Download and then add config file
- Add Firebase SDK
- 4 **Next steps**

You're ready!

Make sure that you take a look at 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.



Step 6: Run the Flutter App

1. Run the Flutter app on a real device or simulator.
2. Check the Firebase dashboard to confirm that the app is successfully connected.

Conclusion:

By integrating Firebase into a Flutter application, developers can utilize a wide range of backend services without managing their own servers. Proper configuration of Firebase for both Android and iOS ensures a seamless connection, enabling features like user authentication, database management, and cloud storage within the app.