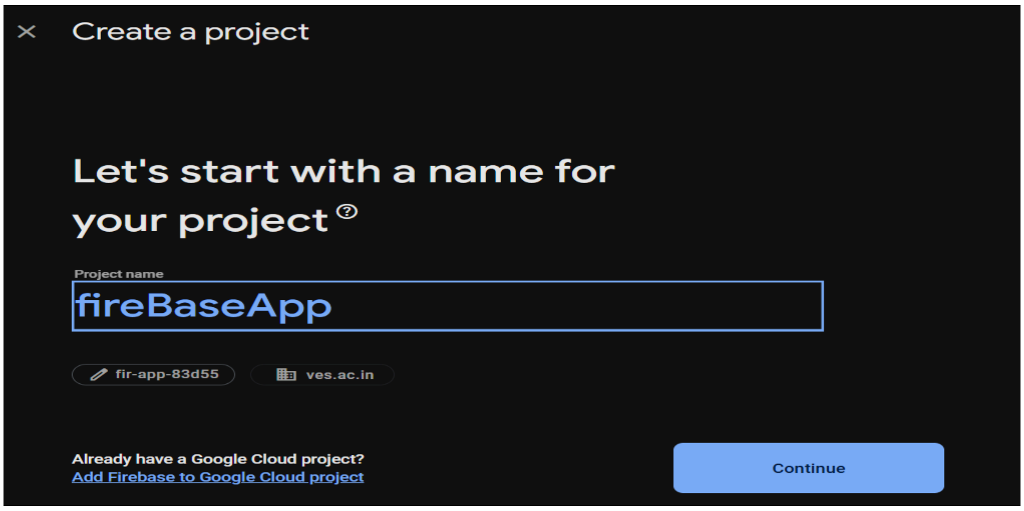
Installation:


Create a project

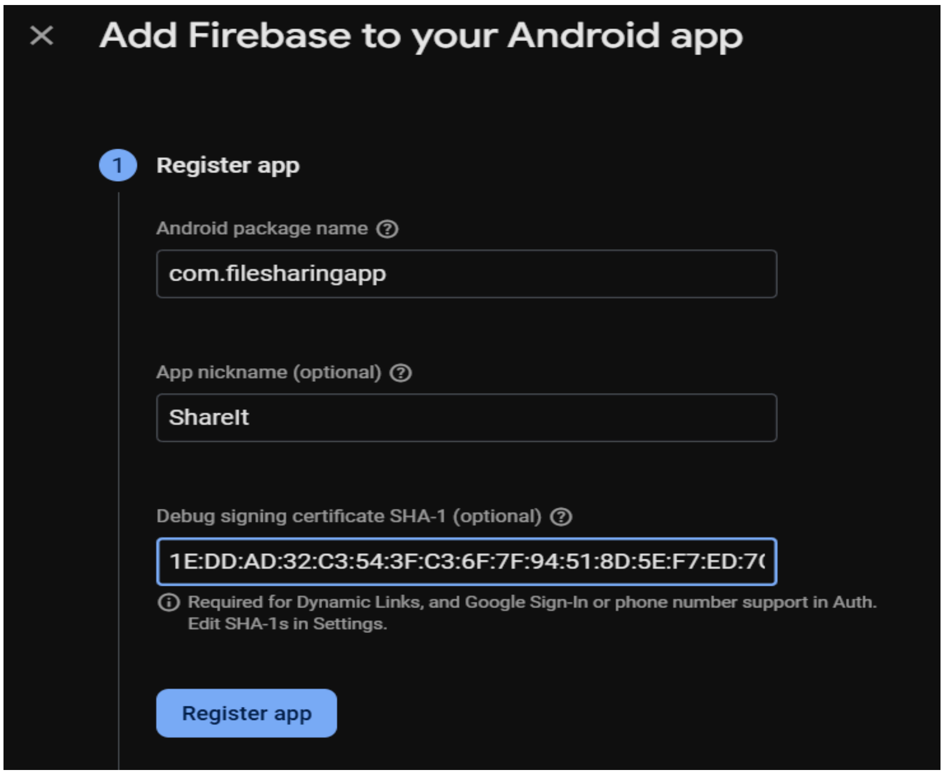
Let's start with a name for your project

Project name

fireBaseApp

fir-app-83d55 | ves.ac.in

Already have a Google Cloud project?
Add Firebase to Google Cloud project


Add Firebase to your Android app

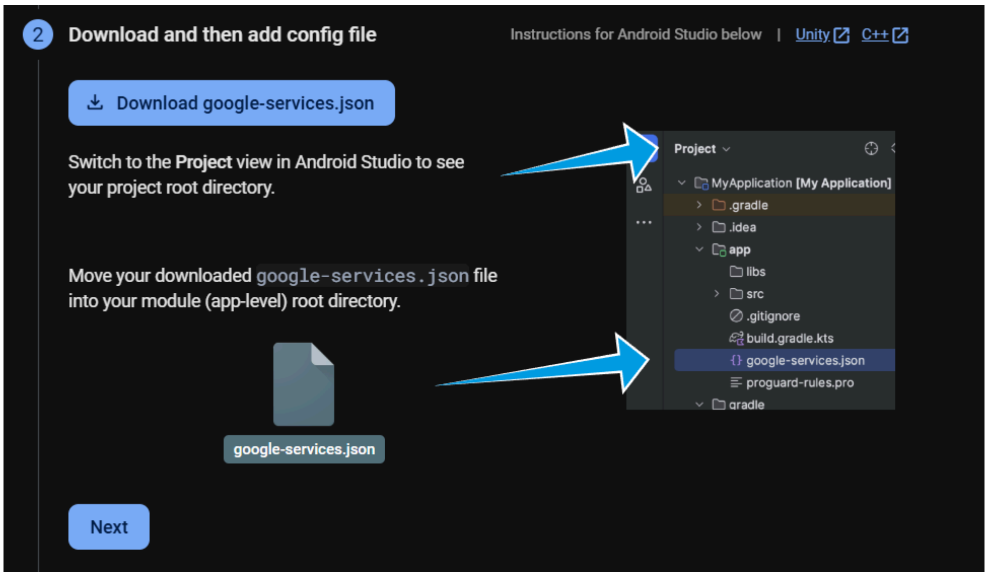
1 Register app

Android package name: com.filesharingapp

App nickname (optional): ShareIt

Debug signing certificate SHA-1 (optional): 1E:DD:AD:32:C3:54:3F:C3:6F:7F:94:51:8D:5E:F7:ED:7(

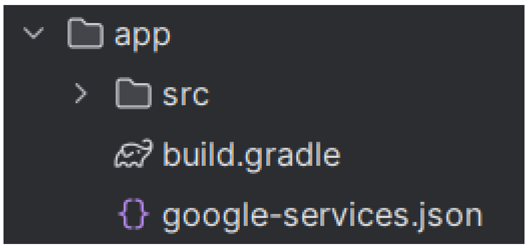
Required for Dynamic Links, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

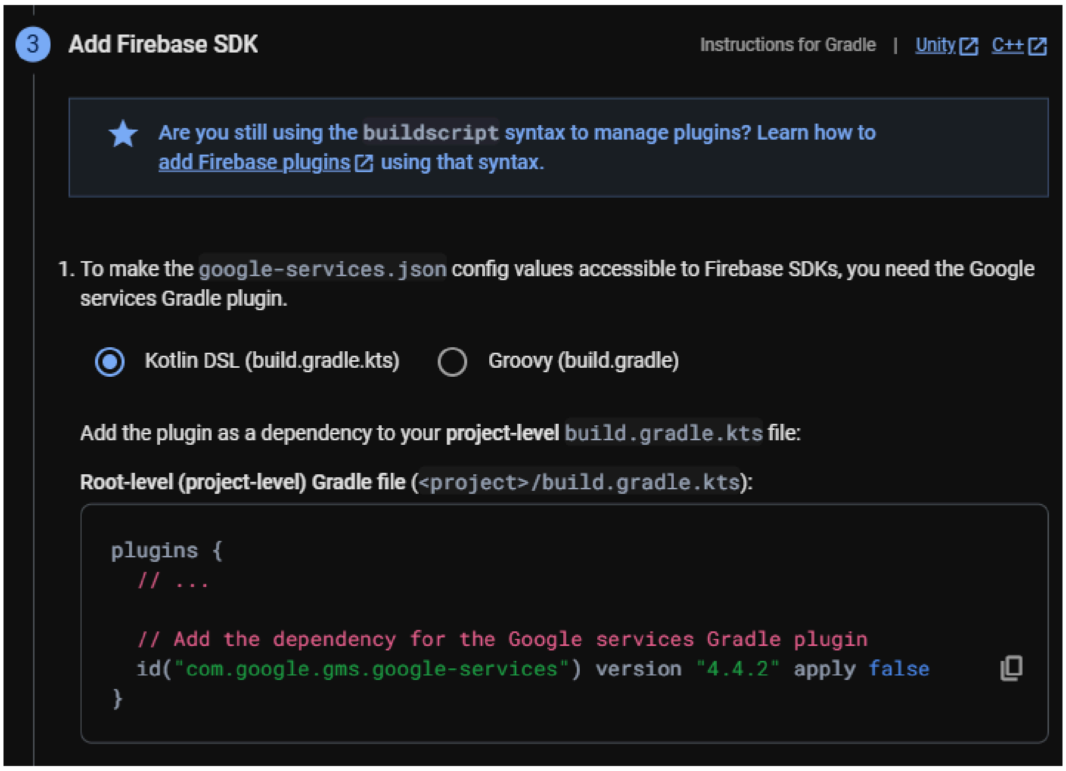

2 Download and then add config file

Instructions for Android Studio below | Unity C++

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.




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.

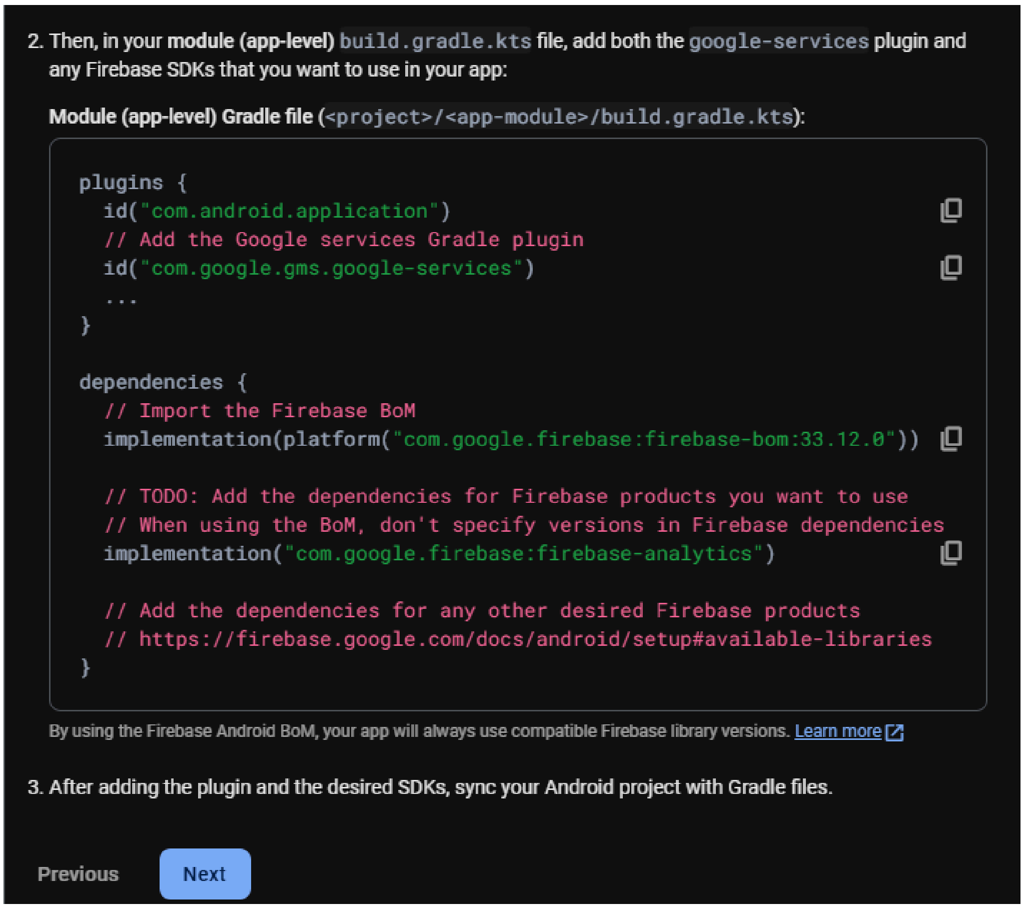
1. To make the `google-services.json` config values accessible to Firebase SDKs, you need the Google services Gradle plugin.

   [✓ 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 "4.4.2" 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:

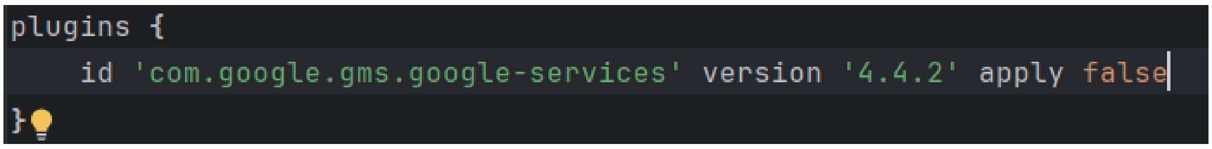
   Module (app-level) Gradle file (`<project>/<app-module>/build.gradle.kts`):

```
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:33.12.0"))

  // 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
}
```

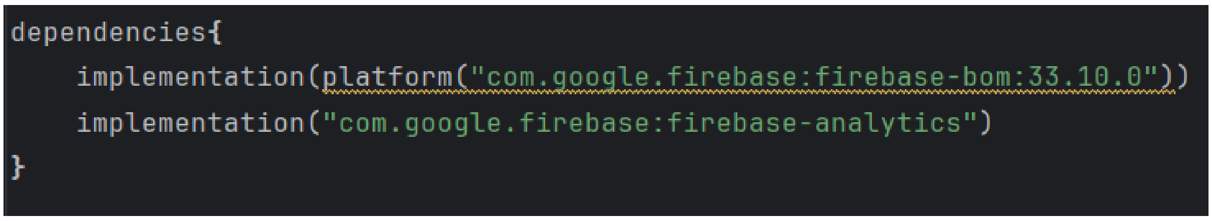
   By using the Firebase Android BoM, your app will always use compatible Firebase library versions. Learn more

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

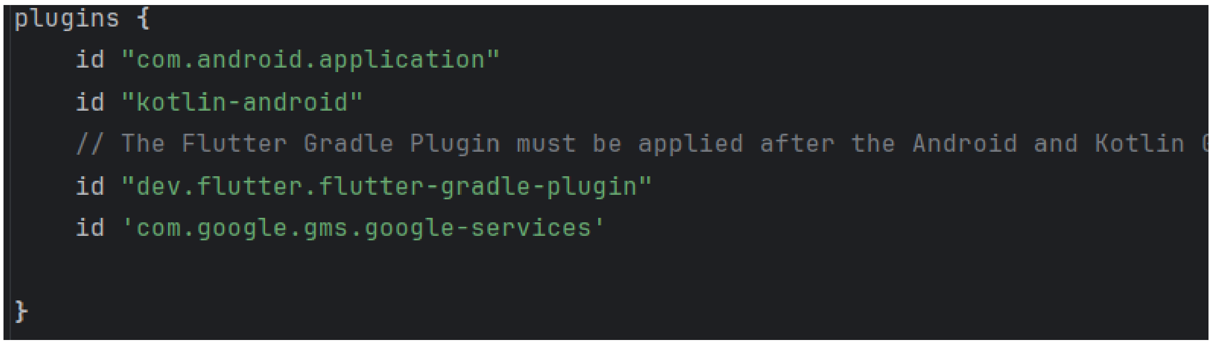
Previous Next



```
plugins {
    id 'com.google.gms.google-services' version '4.4.2' apply false
}
```



```
dependencies{
    implementation(platform("com.google.firebase:firebase-bom:33.10.0"))
    implementation("com.google.firebase:firebase-analytics")
}
```



```
plugins {
    id "com.android.application"
    id "kotlin-android"
    // The Flutter Gradle Plugin must be applied after the Android and Kotlin
    id "dev.flutter.flutter-gradle-plugin"
    id 'com.google.gms.google-services'

}
```

Conclusion:

By following the above steps, Firebase setup for both Android and iOS platforms in a Flutter project is completed. This allows the app to use powerful backend services like authentication, real-time database, cloud messaging, and more, with minimal configuration. Firebase simplifies backend development and helps in building scalable and secure applications efficiently.