Firebase and Firestore Backend

WHAT ARE FIREBASE AND CLOUD FIRESTORE?

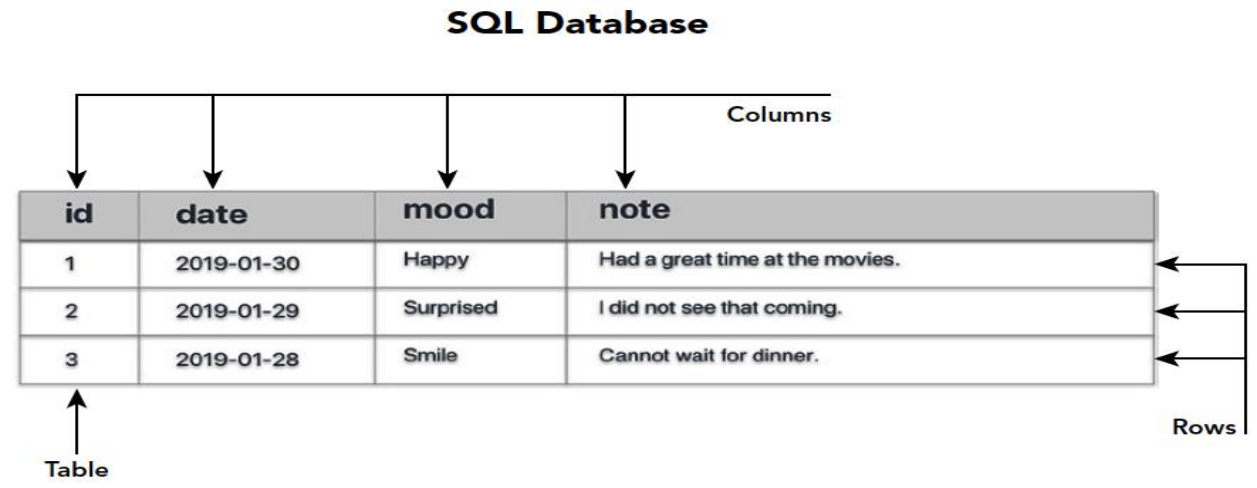
Firebase : a **Backend-as-a-Service (BaaS) platform**, provides a suite of features that simplify building, managing, and scaling apps across iOS, Android.

Firestore a **NoSQL database**, offers advanced data management capabilities, real-time synchronization, and offline support.

Structuring and Data Modeling Cloud Firestore

To understand the **Cloud Firestore** data structure, let's **compare** it to a standard SQL Server database

SQL SERVER DATABASE	CLOUD FIRESTORE
Table	Collection
Row	Document
Columns	Data

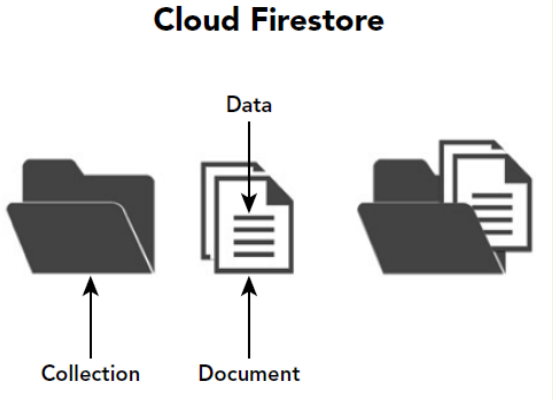


• In Cloud Firestore, a **collection** can contain only documents.
• A **document** is a key-value pair and can optionally point to **subcollections**. Documents cannot point to another document and must be stored in **collections** .

What is the collection's responsibility?
Collections are containers for **documents**; they hold them the same way a folder holds pages.

What is the document's responsibility?

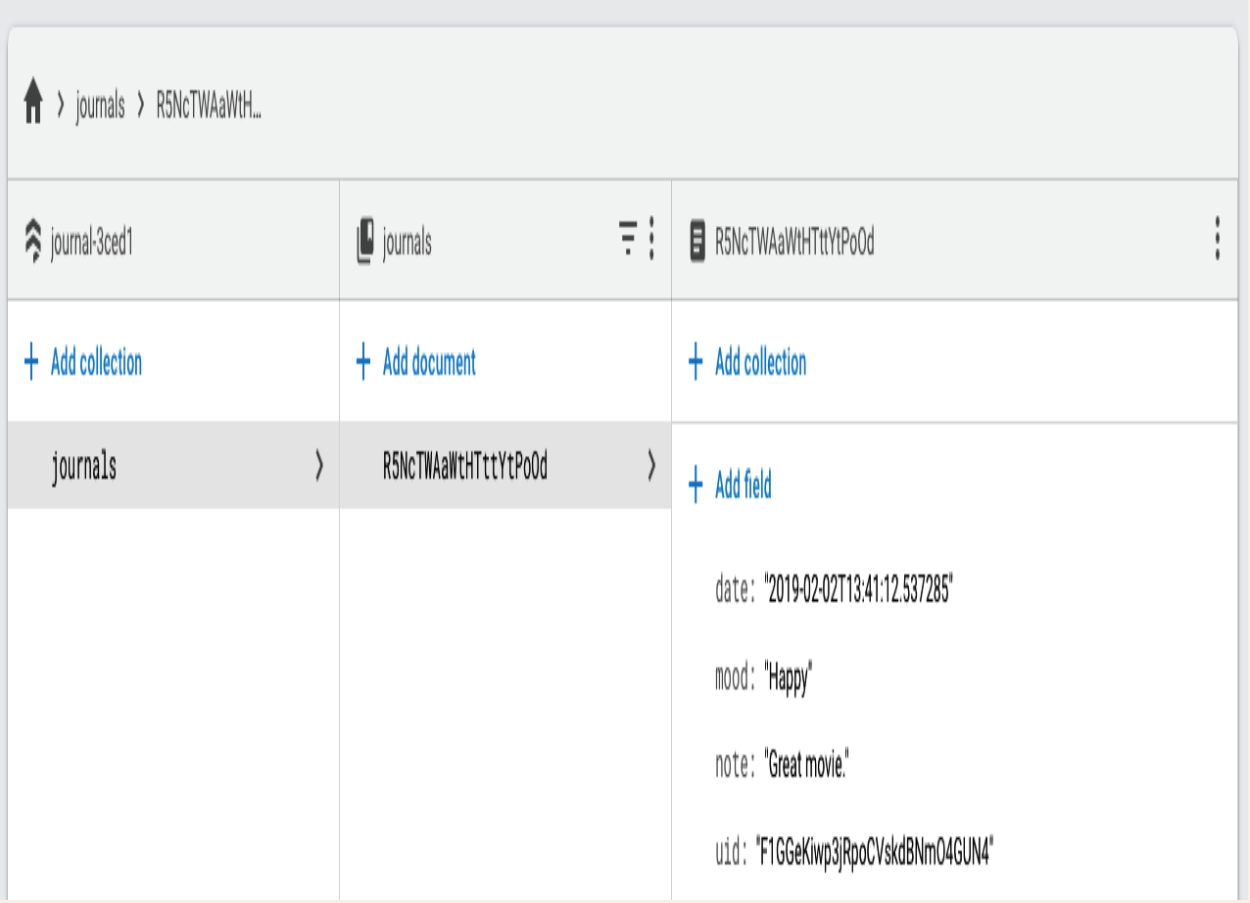
– Documents hold **data** that is stored as a key-value pair similar to JSON.
– Documents support **extra data** types that JSON does not support.
– Each document is identified by name, and they are limited to 1MB in size.



TYPE	VALUE
Collection	journals
Document	R5NcTWAaWtHTttYtPoOd
Document data as key-value pair	date: "2019-0202T13:41:12.537285" mood: "Happy" note: "Great movie." uid: "F1GGeKiwp3jRpoCVskdBNmO4GUN4"

Here Cloud Firestore sample data as **JSON** objects and in the Cloud Firestore console. the document name is a **unique ID** that can be automatically created by Cloud Firestore, or you can **manually** generate it.


```
{
    "journals":[
        {
            " R5NcTWAaWtHTttYtPoOd1":{
                "date": "2019-0202T13:41:12.537285",
                "mood":"Happy",
                "note":"Great movie."
                "uid": "
F1GGeKiwp3jRpoCVskdBNmO4GUN4",
            }
        }
    ]
}
```

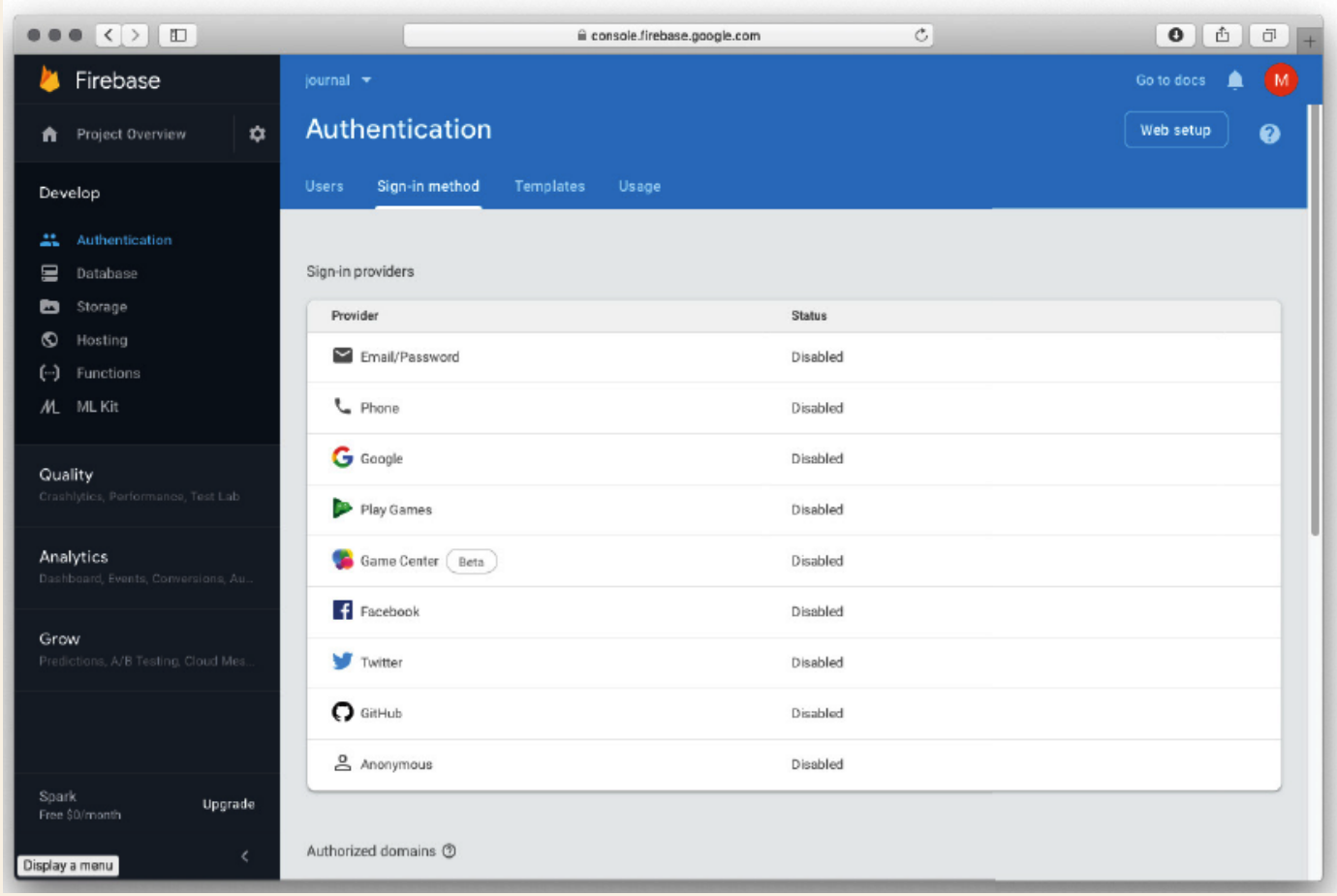



Cloud Firestore supports many data types such as:
array, Boolean, byte, date and time, floating-point number, geographical point, integer, map, reference, text string, and null.

Viewing Firebase Authentication Capabilities

The following is a list of the currently available authentication sign-in providers:

- Email/Password
- Phone
- Google
- Play Games (Google)
- Game Center (Apple)
- Facebook
- Twitter
- GitHub
- Anonymous



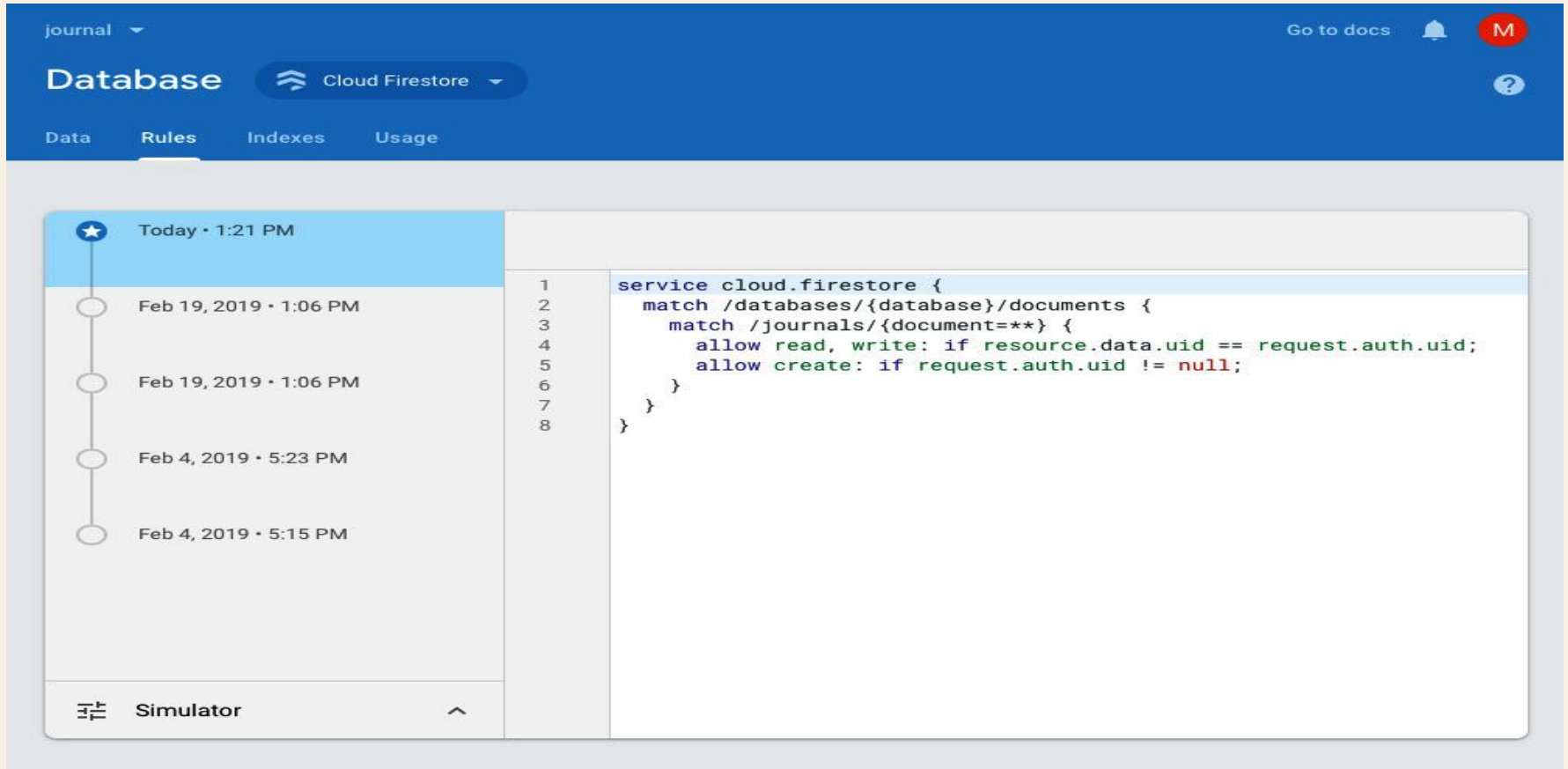
Viewing Cloud Firestore Security Rules

Once you have the **Firebase User object's unique ID**, you use it with Cloud Firestore security rules to secure and lock data to each user.

– The following code shows the security rules that you'll create for securing the Cloud Firestore database:

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /journals/{document=**} {
      allow read, write: if resource.data.uid == request.auth.uid;
      allow create: if request.auth.uid != null;      }    } }
```

1. The first **match /databases/{database}/documents** declaration tells the rules to match any Cloud Firestore Database in the project.
2. The second and the main part of understanding is to use the match statement ,as in **match /journals/{document=**}.**
   - The **journals** declaration is the container name,
   - The expression to evaluate is **document=**** (all documents for the journals collection)
3. Inside this **particular match**, you use the **allow** expression for the **read** and **write** privileges .



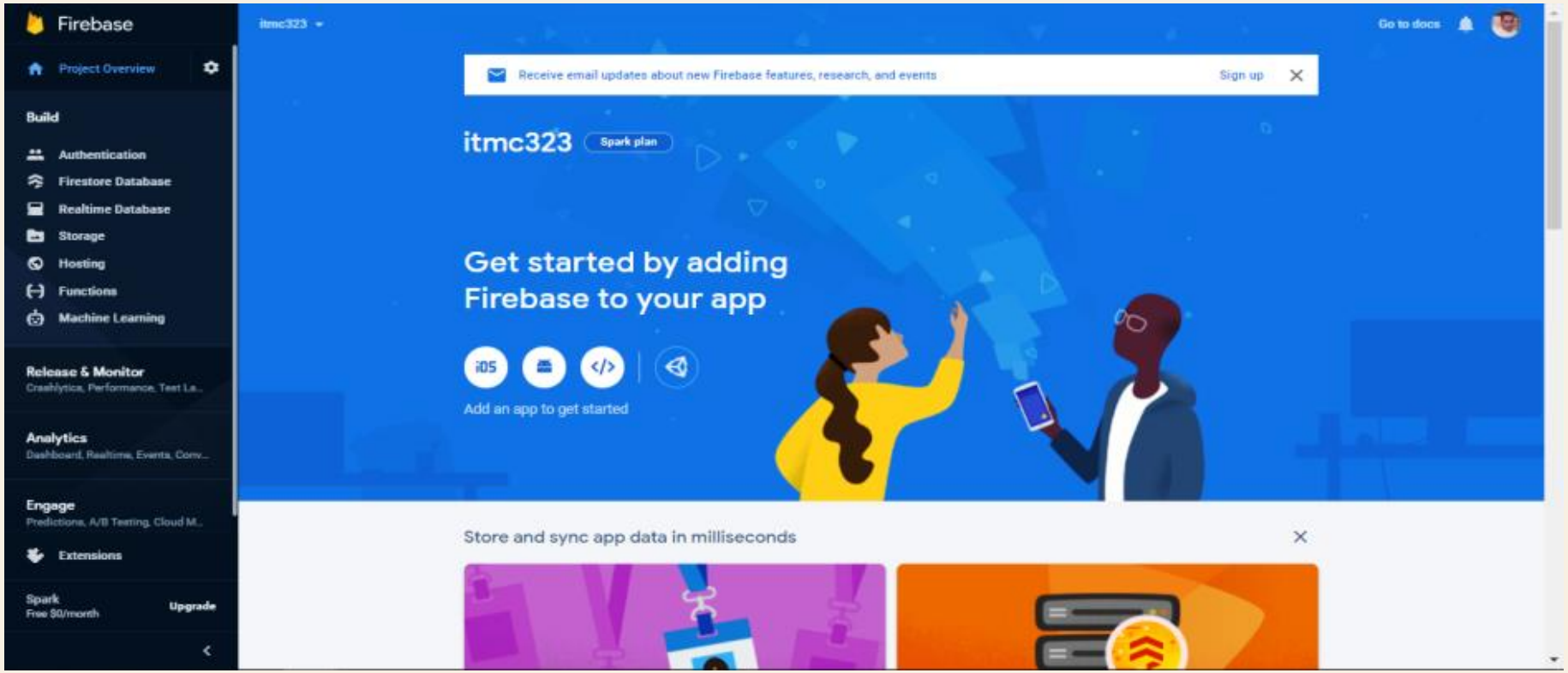
CONFIGURING THE FIREBASE PROJECT

- A **Firebase project** is backed by the Google Cloud Platform, which allows apps to scale.
- The **Firebase project** is a **container** that supports sharing features such as the **database**, **notifications**, **users**, **remote config**, **crash reports**, and **analytics** (many more) between the **iOS**, **Android**, and **web apps**.
- Each account can have multiple projects.

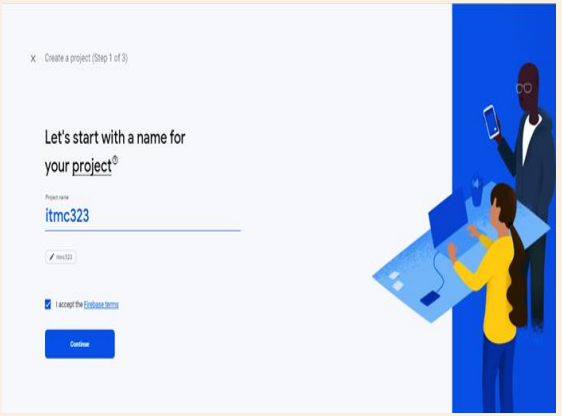
Creating the Firebase Project

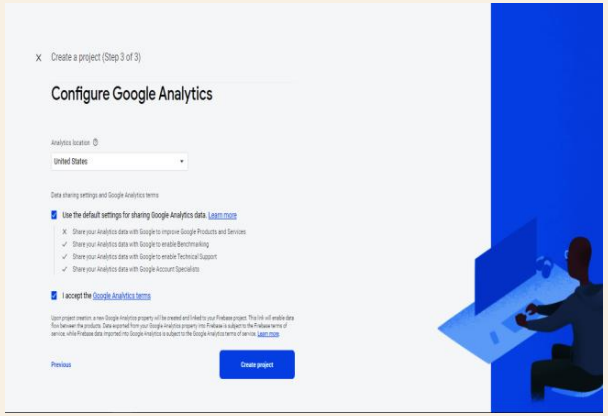
to create a Firebase project that sets up a **container** to start **adding** your **Cloud Firestore database** and enabling **authentication**. You will start by adding the **iOS** app, and then you'll continue by adding the **Android** app.

1. **Navigate** to https://console.firebase.google.com log in to Google Firebase with your Google account.

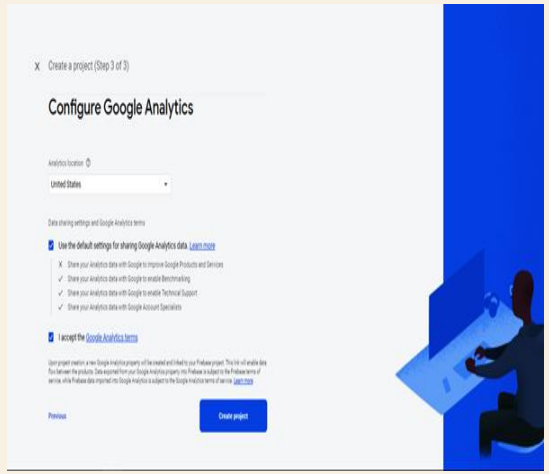


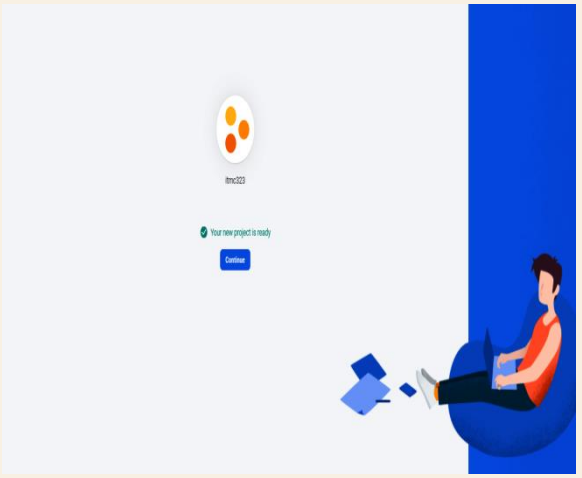
2. Click the **Add Project button** in Firebase; the Add A Project dialog will open. For the project name, enter **project name**. (Your ID will be different because each project name must be **unique**.) , click the **Continue** button.





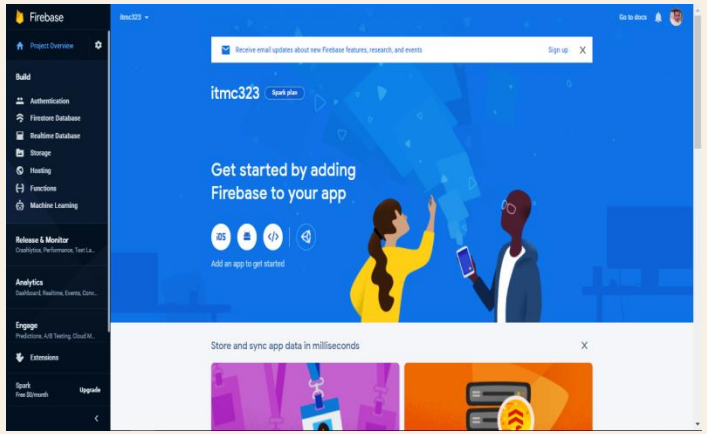
3. In **Configure Google Analytics** page **click** the **Create project** button.





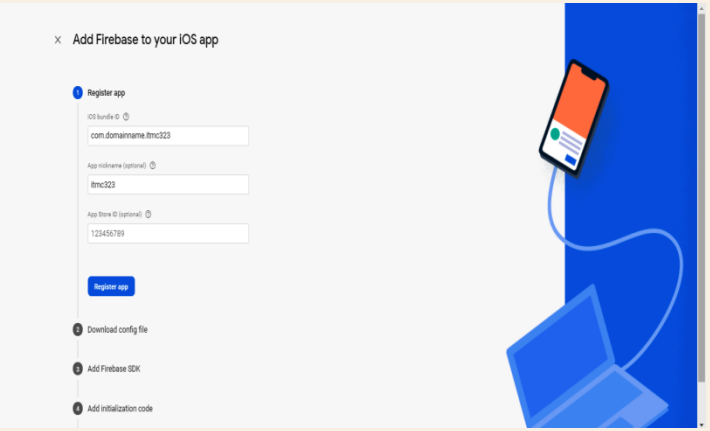
IOS

4. From the main Firebase project page, **click** the **iOS button** to **add** Firebase to the iOS app.



5. Enter the **iOS bundle ID**, as in **com.domainname.YourProjectName**.

6. Enter the optional app **nickname** and skip the optional App Store ID.



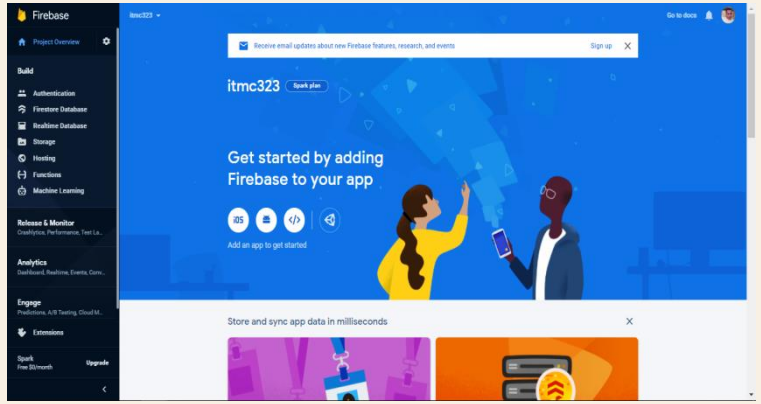
7. Click the **Download GoogleService-Info.plist button**.

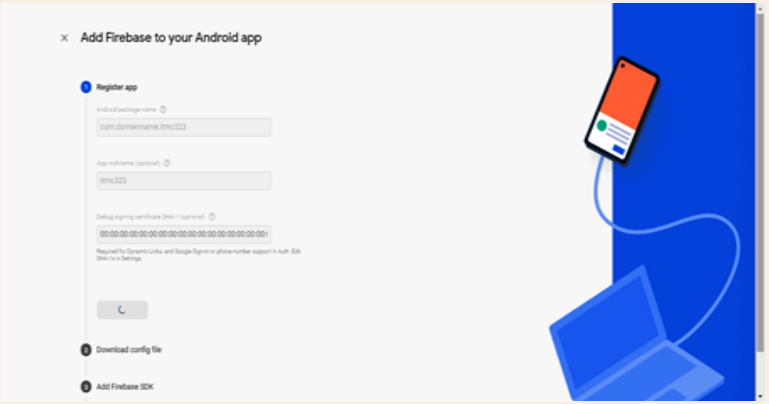
8. Click the **Next button**, and **skip** the Add Firebase SDK and Add Initialization Code steps. Then in **Next step click continue to console**.

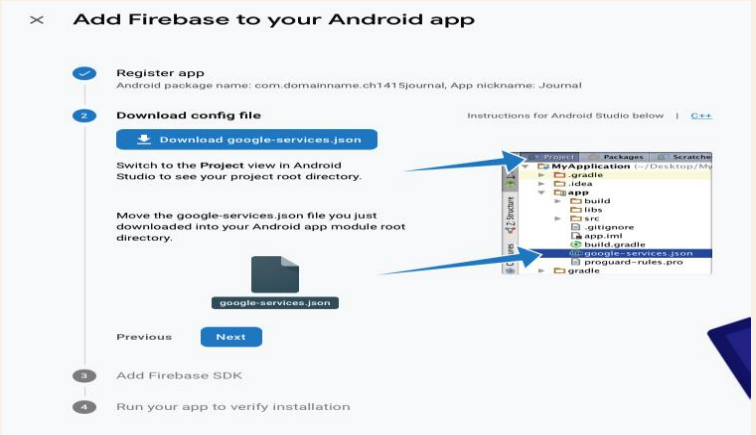
ANDROID

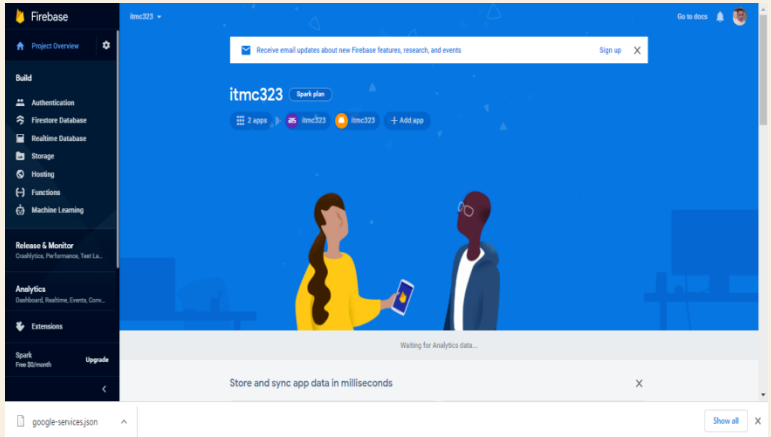
9. On the **main Firebase project** page, **click** the **Add App** button.

10. **Follow Same Steps For IOS**









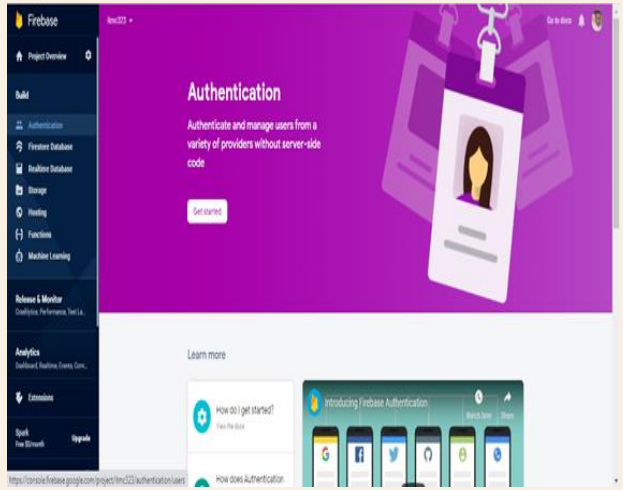
ADDING A CLOUD FIRESTORE DATABASE AND IMPLEMENTING SECURITY

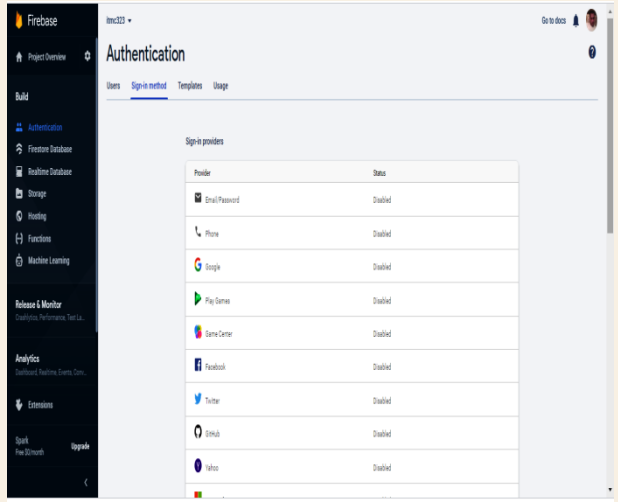
Creating the Cloud Firestore Database and Enabling Authentication

1. **Navigate** to https://console.firebase.google.com and select **your project**.
2. **From** the **menu on the left**, **click** the **Authentication** link in the **Bulid section submenu.**

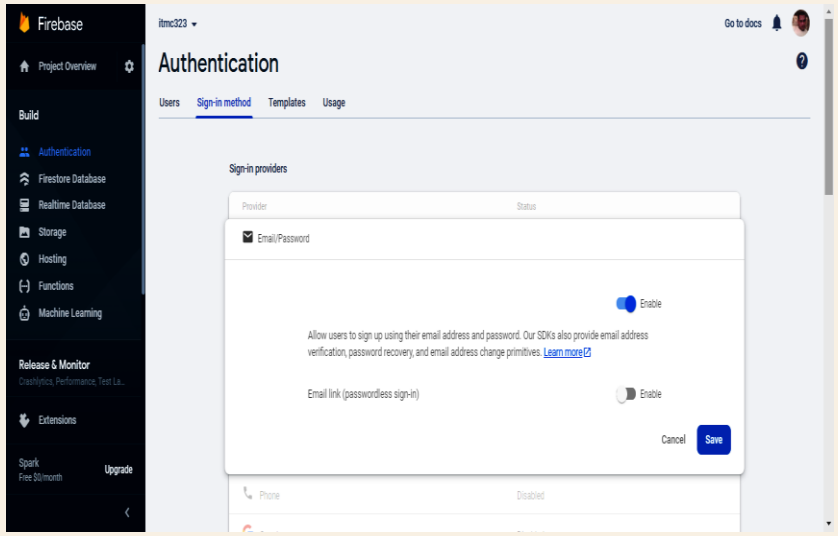
– In **Authentication** page click **Get started**

– **Click** the **Sign-in Method** tab showing a list of available sign-in providers.

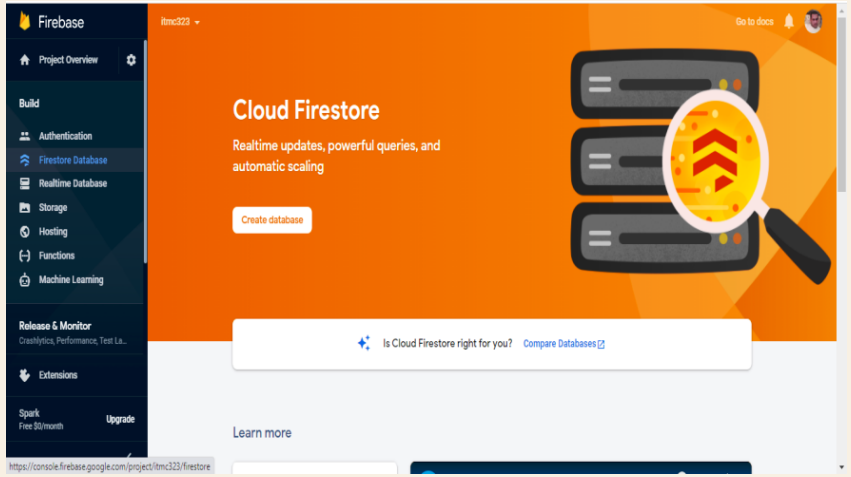




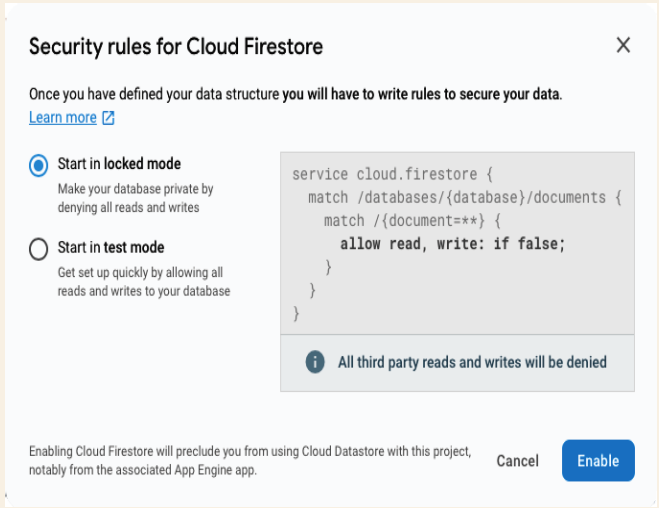
3, Click the **Email/Password** option, **click Enable** to turn on the feature, and **click** the **Save** button.

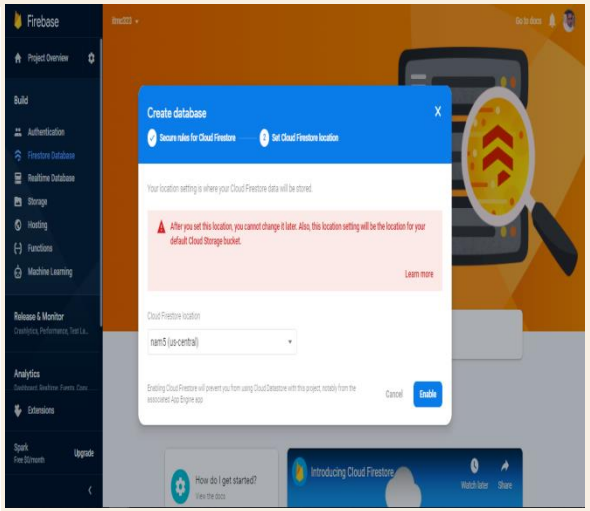


4. In the left menu, **click** the **Firestore Database** link and **click** the **Create Database** (Cloud Firestore) button.



5. In the **Security Rules For Cloud Firestore dialog**, leave the locked mode radio button selected and click the **Next** button Then **Enable**.



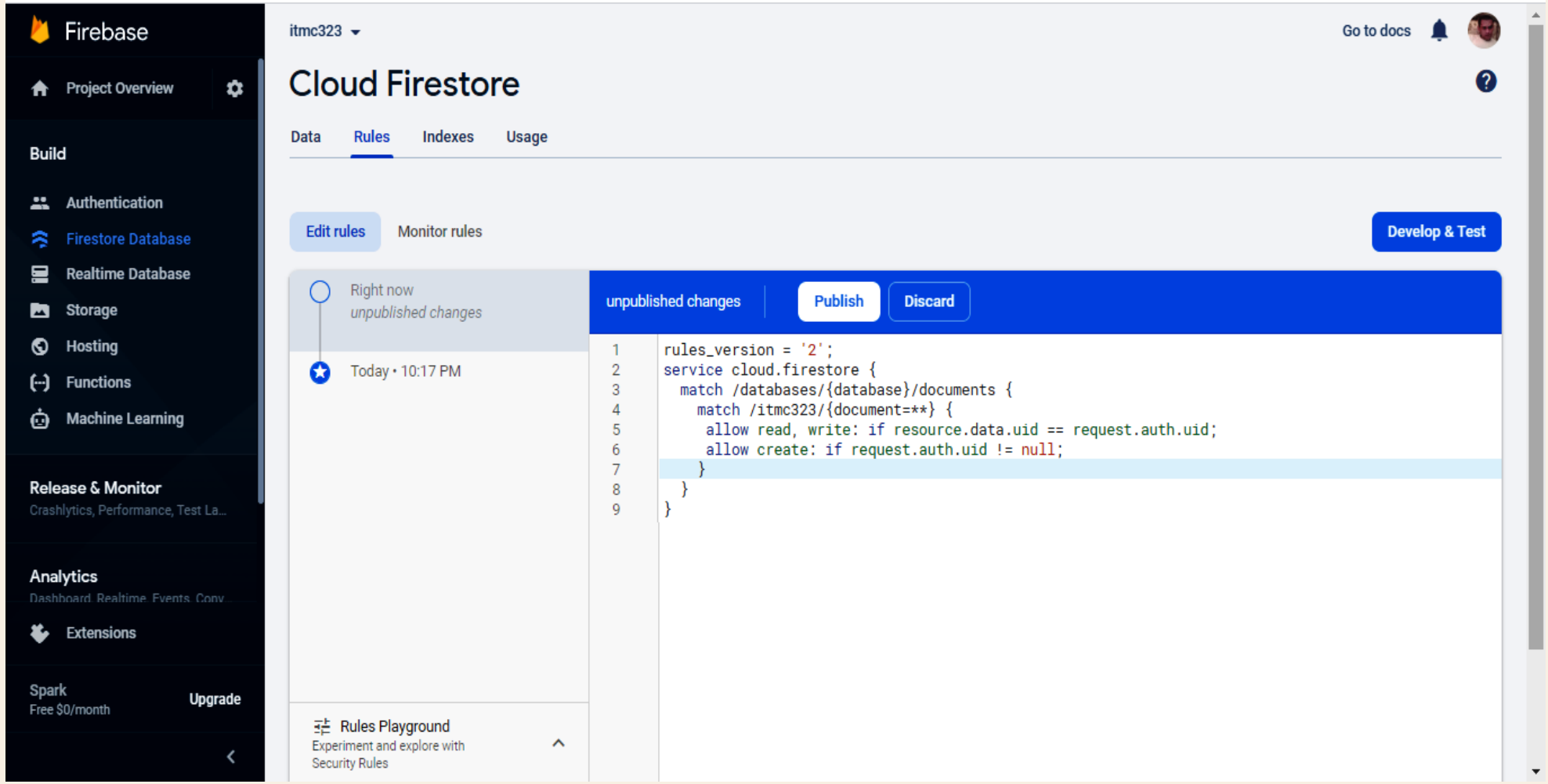


6. Tap the **Rules** tab to edit the default locked rules , **change match /{document=**}** to **match/Your_Project_Name/{document=**}**.

7. Change allow read, write: if false; to **allow read, write: if resource.data.uid == request.auth.uid;** ,. **Add allow create: if request.auth.uid != null;** to allow **creating new records** if the user is authenticated.

```
service cloud.firestore {

    match /databases/{database}/documents {

        match /journals/{document=**} {

            allow read, write: if resource.data.uid == request.auth.uid;

            allow create: if request.auth.uid != null;      }      }}
```

8. Click Publish.



BUILDING THE CLIENT ITMC323 APP

Creating the ITMC323 App

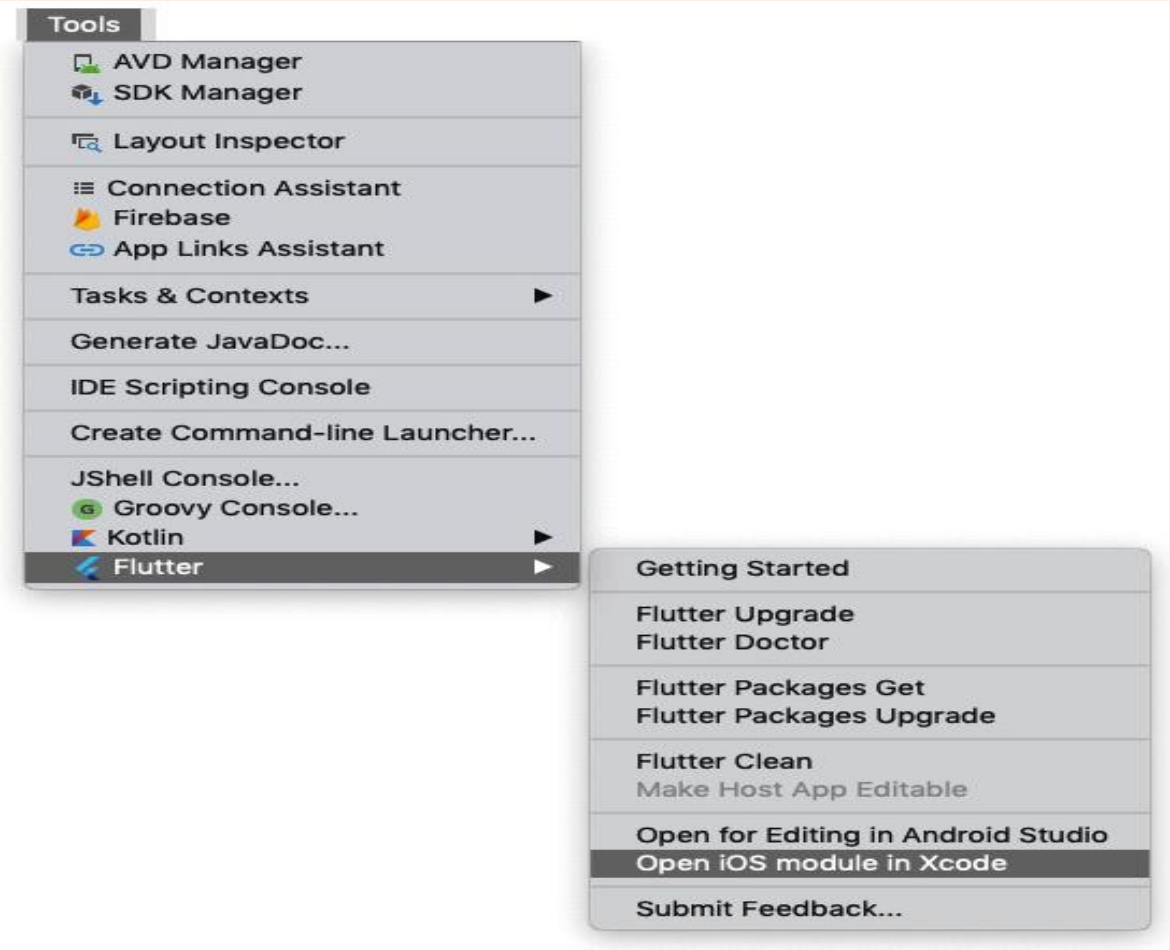
1. **Create** a **new Flutter project** and name it **ITMC323**. For this project, you need to **create** the **pages**, **classes**, **services**, **models**, and **blocs** folders.

2. **Open** the **pubspec.yaml** file to **add** resources. In the **dependencies**: section, **add** the **firebase_auth:^0.11.1+6** and **cloud_firestore:^0.12.5** and **intl:^0.15.8** declarations.

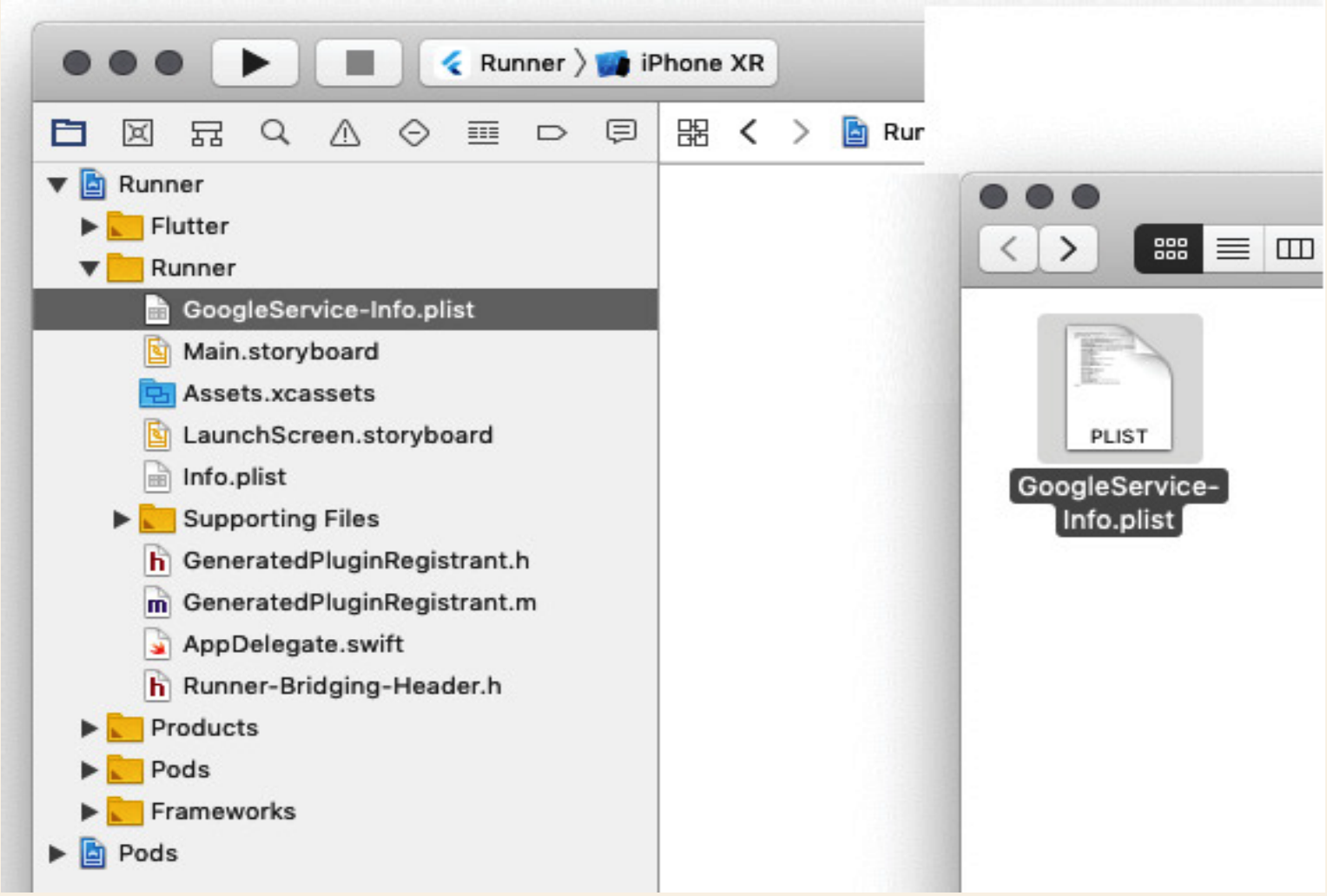
```
dependencies:
     flutter:
          sdk: flutter
    # The following adds the Cupertino Icons font to your application.
    # Use with the CupertinoIcons class for iOS style icons.
    cupertino_icons: ^0.1.2


 firebase_auth: ^0.11.1+6
 cloud_firestore: ^0.12.5
 intl: ^0.15.8
```

3. Click the **Save** button, runs the flutter packages get; once finished, it shows the message **Process finished with exit code 0**.

4. From the **Flutter project**, **open** the **iOS Xcode** project to **add Firebase**. From Android Studio, **click** the menu bar and select **Tools** ➪ **Flutter** ➪ **Open iOS Module In Xcode**.

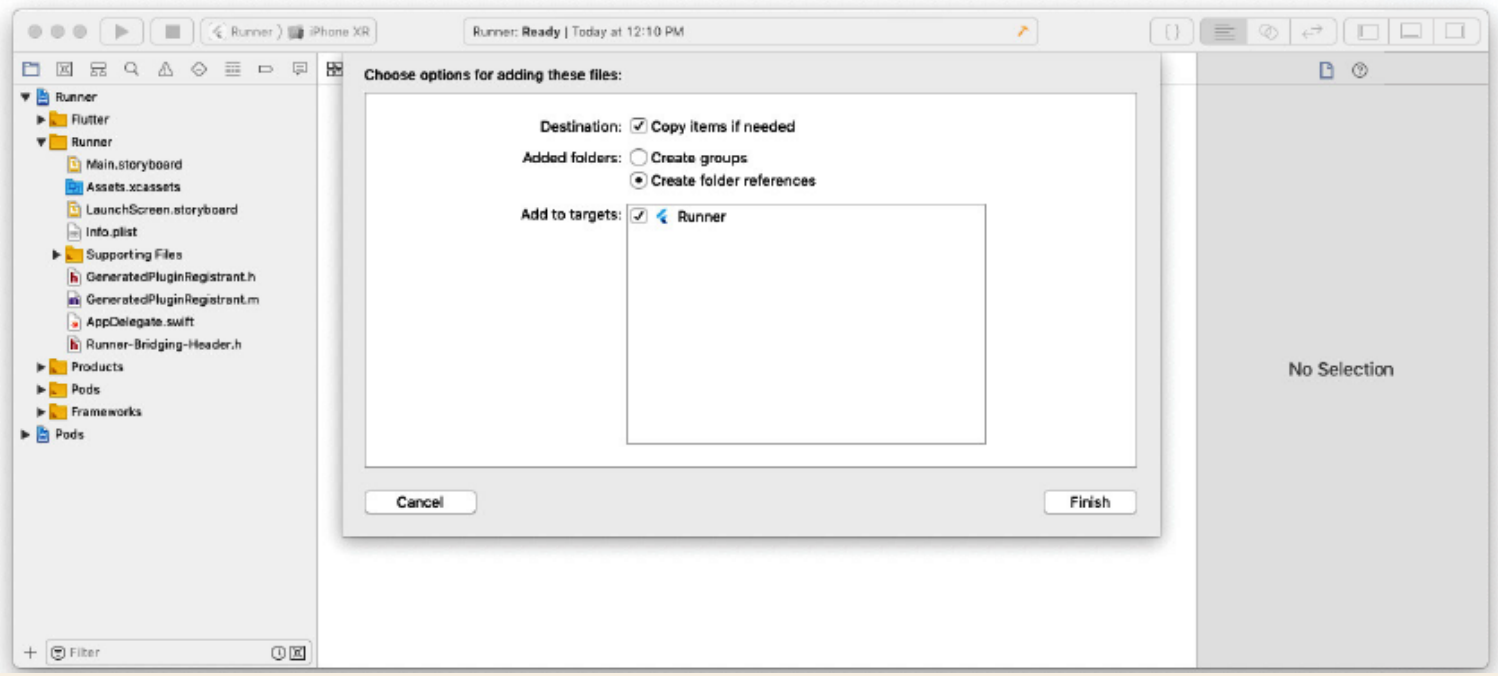


5. Drag the downloaded **GoogleService-Info.plist** file to the **Runner** folder in the **Xcode** project.

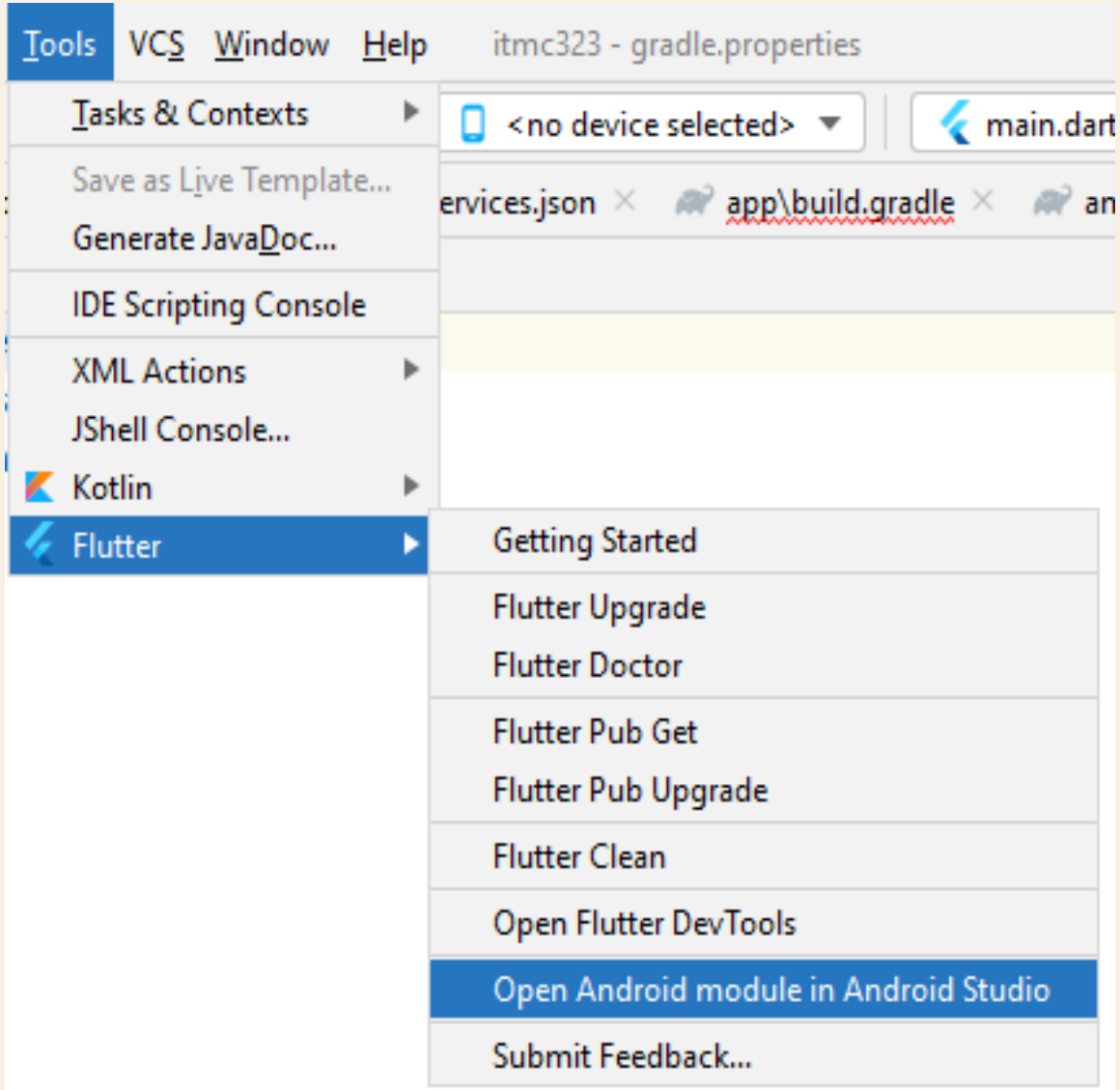


6. In the next dialog, **finish adding** the **GoogleService-Info.plist** file and make sure:

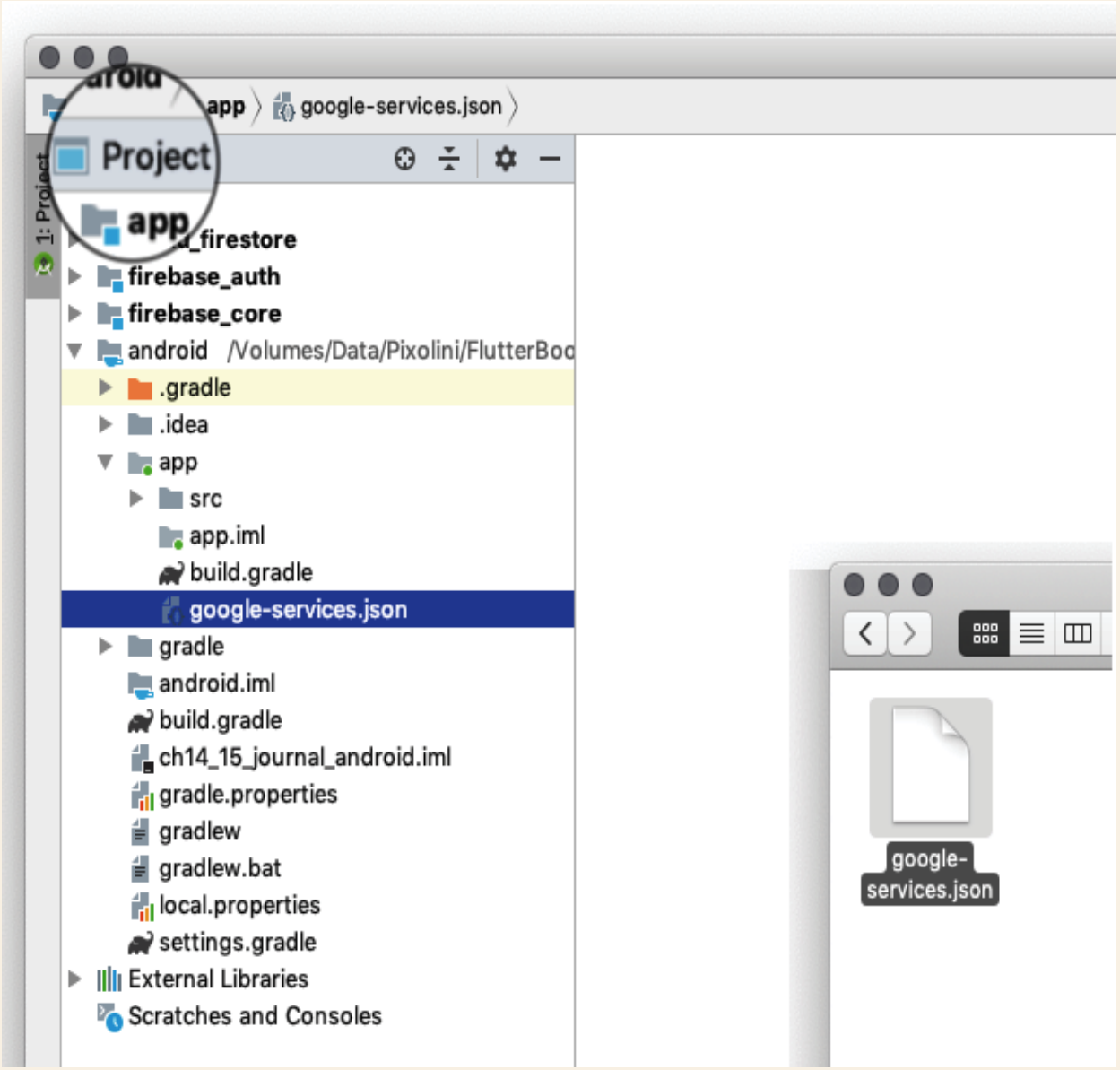
- **Copy Items If Needed** is **checked**,
- **Create Folder References radio button** is **selected**,
- the **Add To Targets ⇨Runner option** is **checked**.
- The iOS Xcode project is now configured to handle Firebase and Firestore.
- Once the file is copied, **close** Xcode.



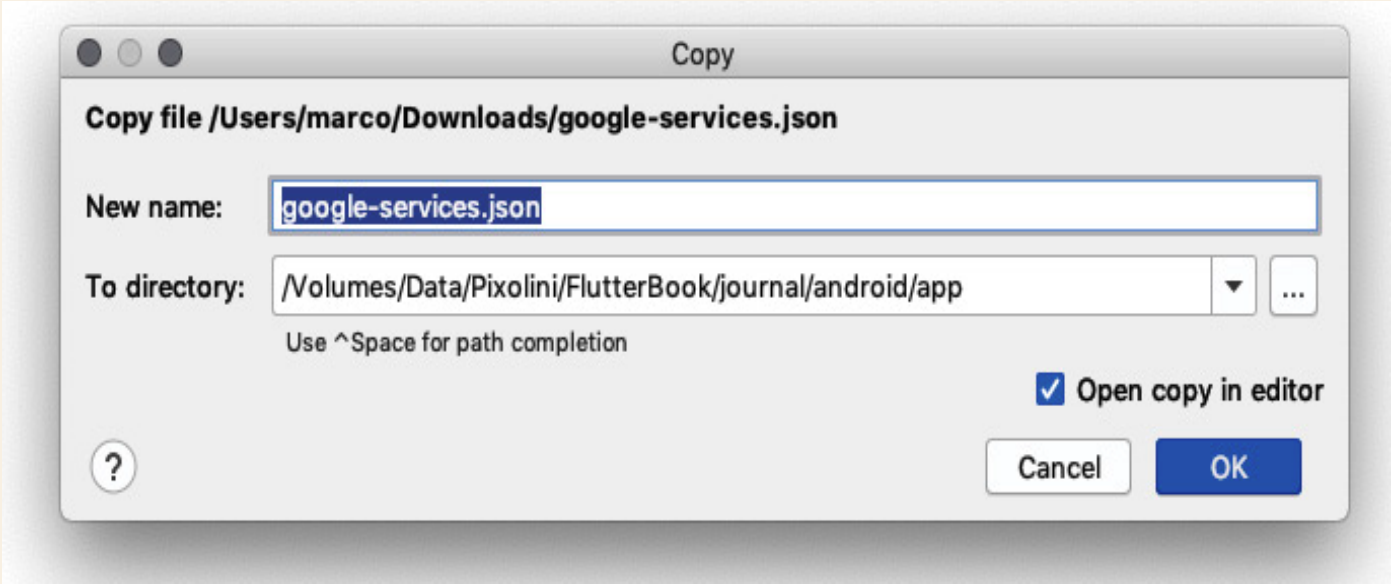
7. From the Flutter project, **open** the **Android Studio** project to **add** Firebase. From Android Studio, **click** the **menu bar** and **select Tools ➪ Flutter ➪ Open For Editing In Android Studio- open android module in android studio**.



8. Drag the **downloaded google-services.json** file to the **App folder** in the Android project.



9. **Finish adding** the **google-services.json** file and click the **OK-Refactor** button.



10. For the Android project, you need to edit **two files manually**.

- For the **first file**, **open** the **app level build.gradle** file located at **android/app/build.gradle**.
- **Add** to the **bottom** of the file the **google-services gradle plugin** by specifying **apply plugin: 'com.google.gms.googleservices'**.
- In this **app-level build.gradle** file, make sure you are using **compileSdkVersion 29, minSdkVersion 16**, and **targetSdkVersion 29** and **save**.

11. To **avoid** receiving **".dex file cannot exceed 64 error"** when you try to run the Flutter app for Android, you need to **add** the **multiDexEnabled true** in the **defaultConfig** section and **save**.

```
android {
        compileSdkVersion 28
        sourceSets {...}
        lintOptions {...}
        defaultConfig {
              applicationId "com.domainname.journal"
              minSdkVersion 16
              targetSdkVersion 28
              // ...
              // Enable if you get error in Flutter app - .dex file cannot exceed 64K
  multiDexEnabled true // Enable          }
         buildTypes {...} }
flutter {...}
dependencies {
// Add at the bottom of the file
apply plugin:'com.google.gms.google-services'
```

Project

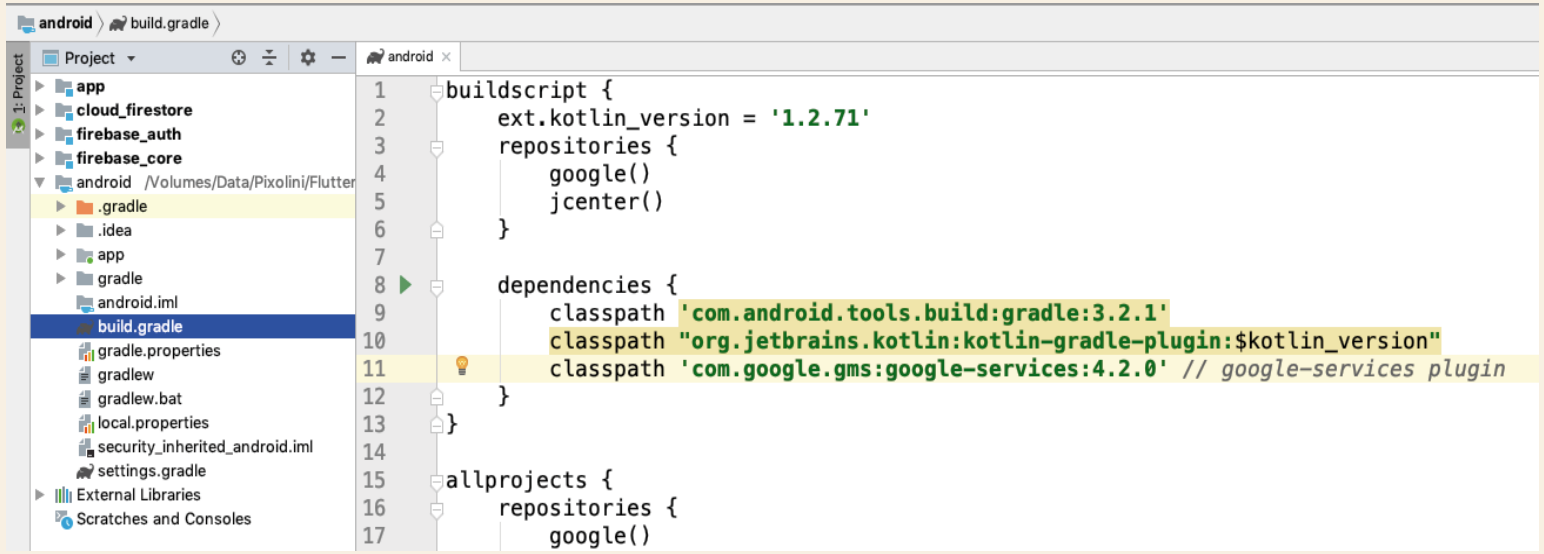
- android /Volumes/Data/Pixolini/FlutterBoo
  - .gradle
  - .idea
  - app
    - src
    - app.iml
    - build.gradle
    - google-services.json
  - gradle
  - android.iml
  - build.gradle
  - gradle.properties
  - gradlew
  - gradlew.bat
  - journal_android.iml
  - local.properties
  - settings.gradle
- android [cloud_firestore] ~/Development/
- android [firebase_auth] ~/Development/fl
- android [firebase_core] ~/Development/fl
- External Libraries
- Scratches and Consoles

android | app | gradle.properties | google-services.json

```
24  apply plugin: 'com.android.application'
25  apply plugin: 'kotlin-android'
26  apply from: "$flutterRoot/packages/flutter_tools/gradle/flutter.gradle"
27
28  android {
29      compileSdkVersion 28
30
31      sourceSets {
32          main.java.srcDirs += 'src/main/kotlin'
33      }
34
35      lintOptions {
36          disable 'InvalidPackage'
37      }
38
39      defaultConfig {
40          // TODO: Specify your own unique Application ID (https://developer.android.com
41          applicationId "com.domainname.journal"
42          minSdkVersion 16
43          targetSdkVersion 28
44          versionCode flutterVersionCode.toInteger()
45          versionName flutterVersionName
46          testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
47          // Enable if you get error in Flutter app - .dex file cannot exceed 64K
48          multiDexEnabled true // Enable
49      }
50
51      buildTypes {...}
58  }
59
60  flutter {source '../..'}
63
64  dependencies {
65      implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
66      testImplementation 'junit:junit:4.12'
67      androidTestImplementation 'com.android.support.test:runner:1.0.2'
68      androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
69      // Add if you get error in Flutter app - .dex file cannot exceed 64K
70      // The multidex library as a dependency
71      implementation 'com.android.support:multidex:1.0.3'
72  }
73
74  apply plugin: 'com.google.gms.google-services'
```

12. For the **second file**, open the **project-level build.gradle** file located at **android/build.gradle**. **Add** to the **dependencies** the **classpath** of the **google-services** plugin and **save**.

```
buildscript {
      // ...
      dependencies {
            // ...
            // Add the following line:
            classpath 'com.google.gms:google-services:4.2.0' //googleservices plugin
      }
}
```



13. You will see a **yellow bar** with a notice that the **gradle files have changed**. **Click** the **Sync Now button**, and once the process is done, **close** the Android project.