

Aim: To Connect Flutter UI with fireBase database, ie. To Set Up Firebase with Flutter for iOS and Android Apps

Theory:

Firebase is a comprehensive platform provided by Google for building mobile and web applications. It offers a wide range of tools and services to help developers build high-quality apps quickly and efficiently. Here's an overview of Firebase and its key components:

Key Components:

1. **Realtime Database:** Firebase Realtime Database is a NoSQL cloud database that allows developers to store and sync data in real-time across multiple clients. It's suitable for applications requiring real-time updates, such as chat apps, collaborative tools, and live data feeds.
2. **Cloud Firestore:** Firestore is Firebase's newer database solution that offers more powerful querying capabilities, scalability, and real-time updates. It's a flexible, scalable database for mobile, web, and server development.
3. **Authentication:** Firebase Authentication provides a secure and easy-to-use authentication system that supports various authentication methods, including email/password, phone number, Google Sign-In, Facebook Login, and more.
4. **Cloud Storage:** Firebase Cloud Storage allows developers to store and serve user-generated content, such as images, videos, and audio files, directly from Google's infrastructure. It offers scalable, secure, and reliable storage solutions with powerful SDKs for easy integration.
5. **Analytics:** Firebase Analytics offers powerful analytics features to help developers understand user behavior, measure app performance, and track key metrics. It provides insights into user engagement, retention, and conversion rates.

Prerequisites:

- 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.

1. Create a flutter appflutter create storyverse

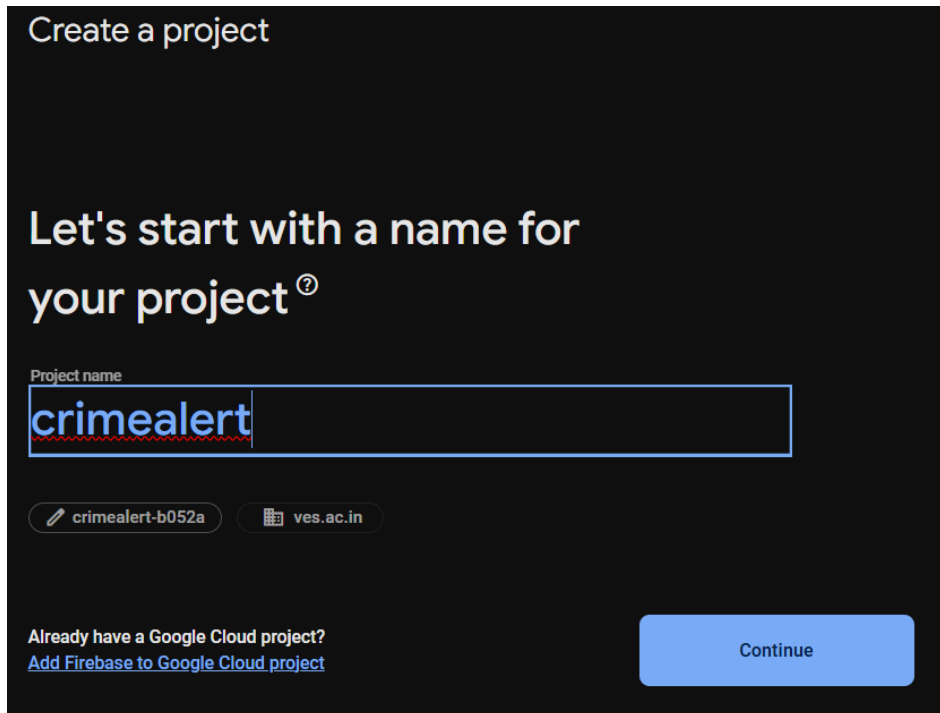
Navigate to the new project directory:

```
cd crimealert
```

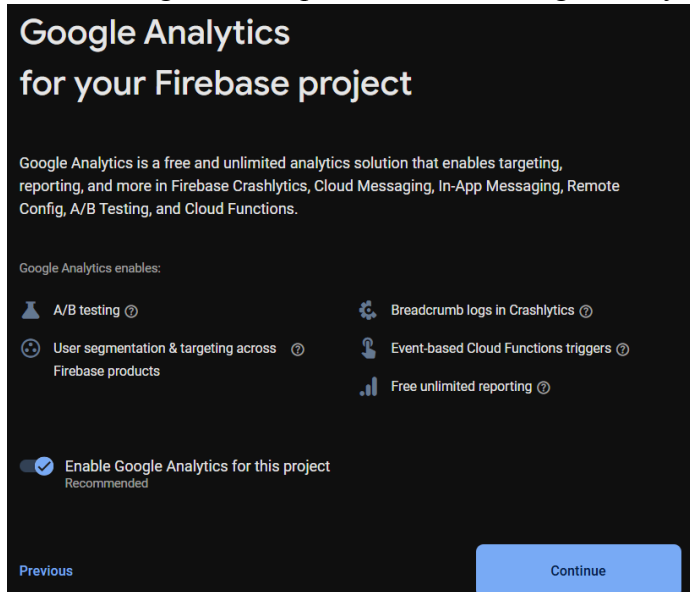
Now that we've got a Flutter project up and running, we can add Firebase.

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.





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

Configure Google Analytics

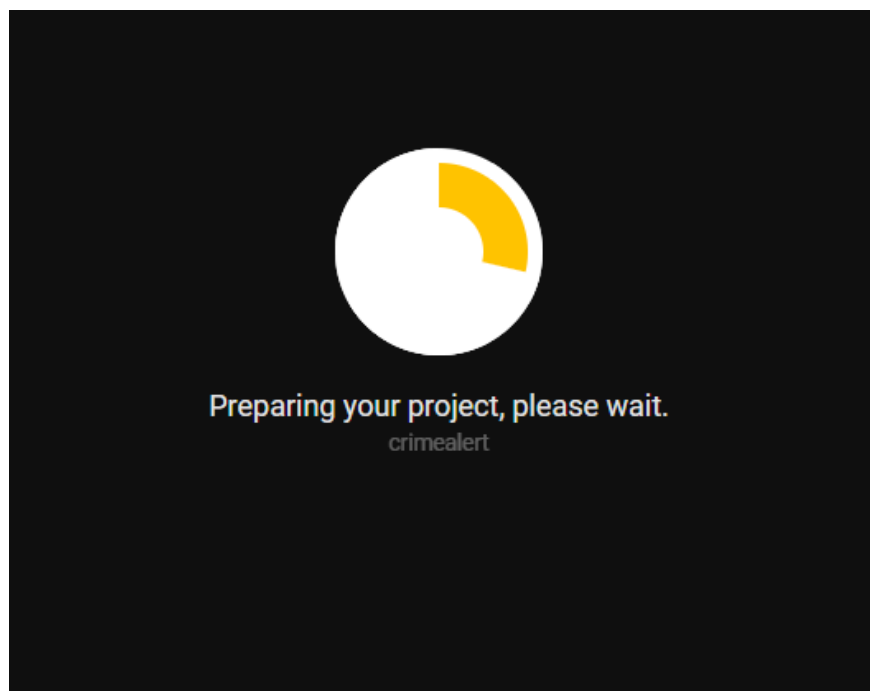
Choose or create a Google Analytics account ?

 Shetty Avan ▼

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](#) .

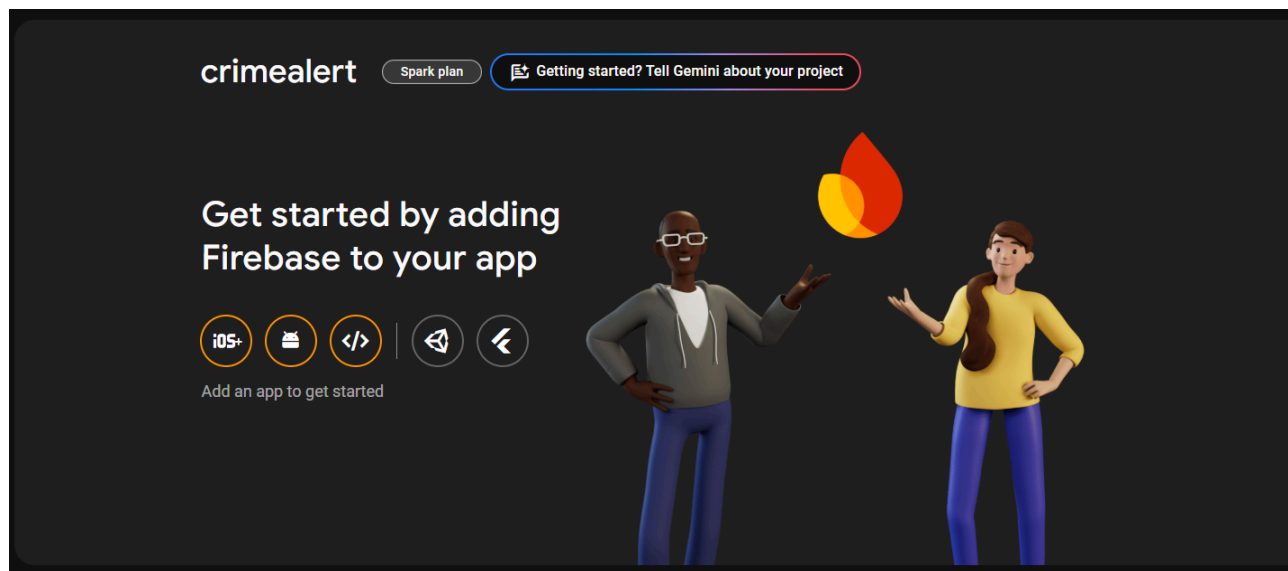
[Previous](#) [Create project](#)



After pressing Continue, the project will be created and resources will be provisioned. We 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.

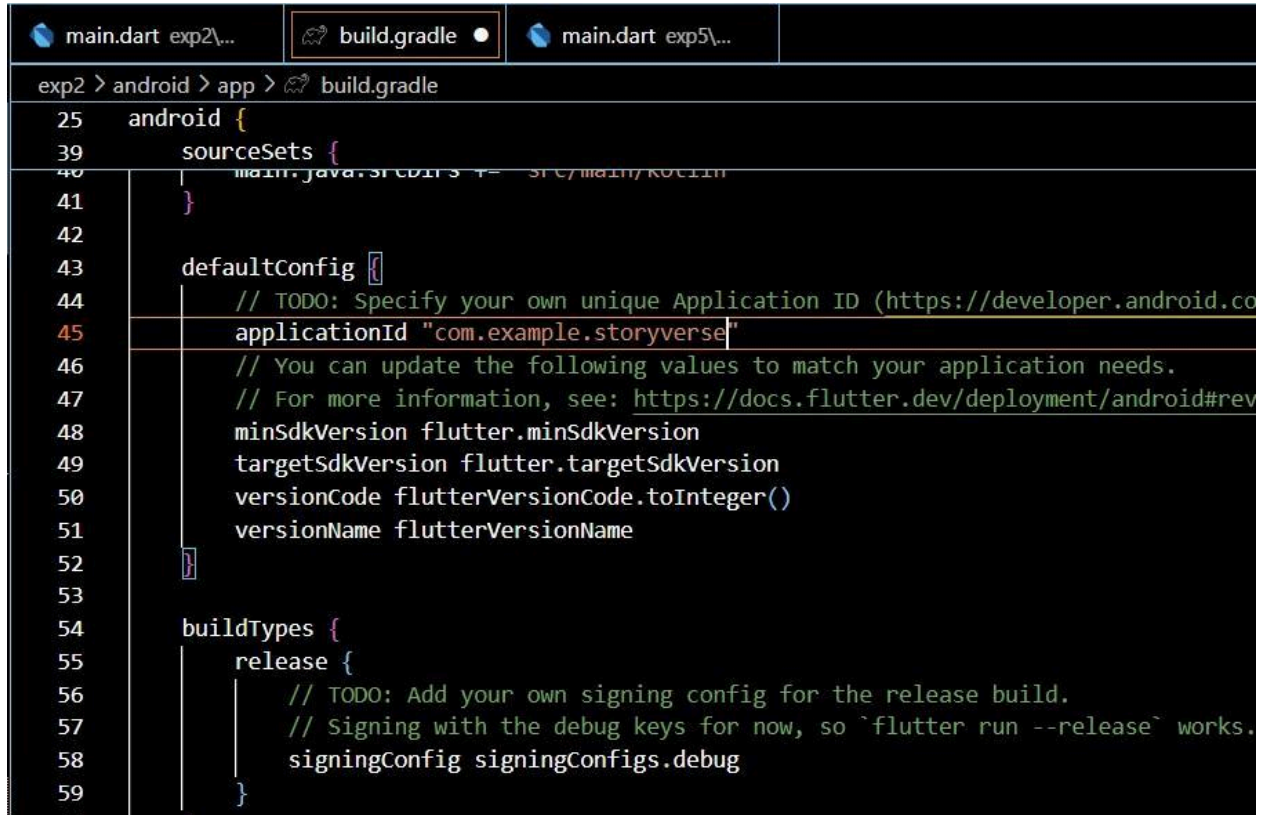


This brings us to the following screen:

The screenshot shows the 'Add Firebase to your Android app' screen, specifically the '1 Register app' step. It has a close button (X) in the top left. The form contains three input fields: 'Android package name' with the value 'com.example.crimealert', 'App nickname (optional)' with the value 'crimealert', and 'Debug signing certificate SHA-1 (optional)' with a long hexadecimal string. Below the third field is a note: 'Required for Dynamic Links, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.' A 'Register app' button is at the bottom of the form. On the left side, there is a vertical progress indicator with four steps: '1 Register app' (active), '2 Download and then add config file', '3 Add Firebase SDK', and '4 Next steps'.

The most important thing here is to match up the Android package name that you choose here with the one inside of our application.

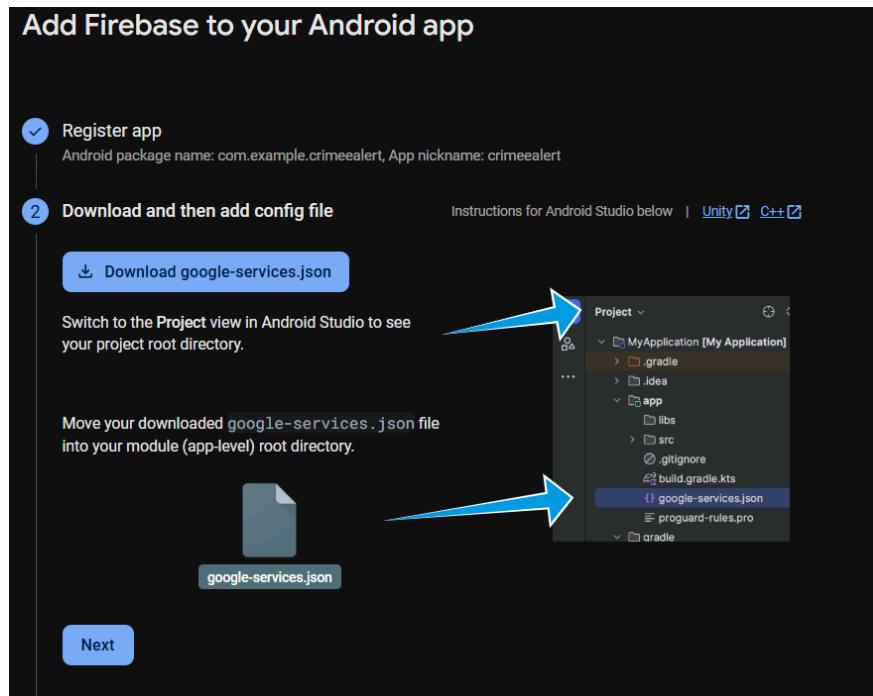
Once you've decided on a name, open android/app/build.gradle in your code editor and update the applicationId to match the Android package name:



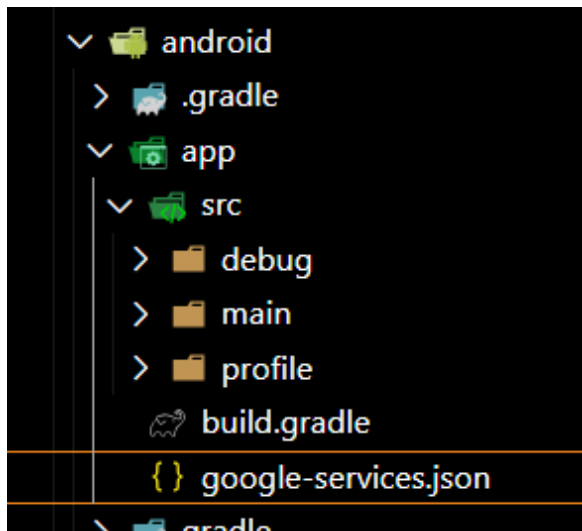
```
exp2 > android > app > build.gradle
25  android {
39      sourceSets {
40          // main.java.sourceSets += src/main/kotlin
41      }
42
43      defaultConfig {
44          // TODO: Specify your own unique Application ID (https://developer.android.co
45          applicationId "com.example.storyverse"
46          // You can update the following values to match your application needs.
47          // For more information, see: https://docs.flutter.dev/deployment/android#rev
48          minSdkVersion flutter.minSdkVersion
49          targetSdkVersion flutter.targetSdkVersion
50          versionCode flutterVersionCode.toInteger()
51          versionName flutterVersionName
52      }
53
54      buildTypes {
55          release {
56              // TODO: Add your own signing config for the release build.
57              // Signing with the debug keys for now, so `flutter run --release` works.
58              signingConfig signingConfigs.debug
59          }
60      }
61  }
```

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

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.



We'll now need to update our Gradle configuration to include the Google Services plugin. Follow these steps to add the firebase sdk in the source code

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:32.7.4"))  
  
    // 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.

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 "4.4.1" apply false  
}
```

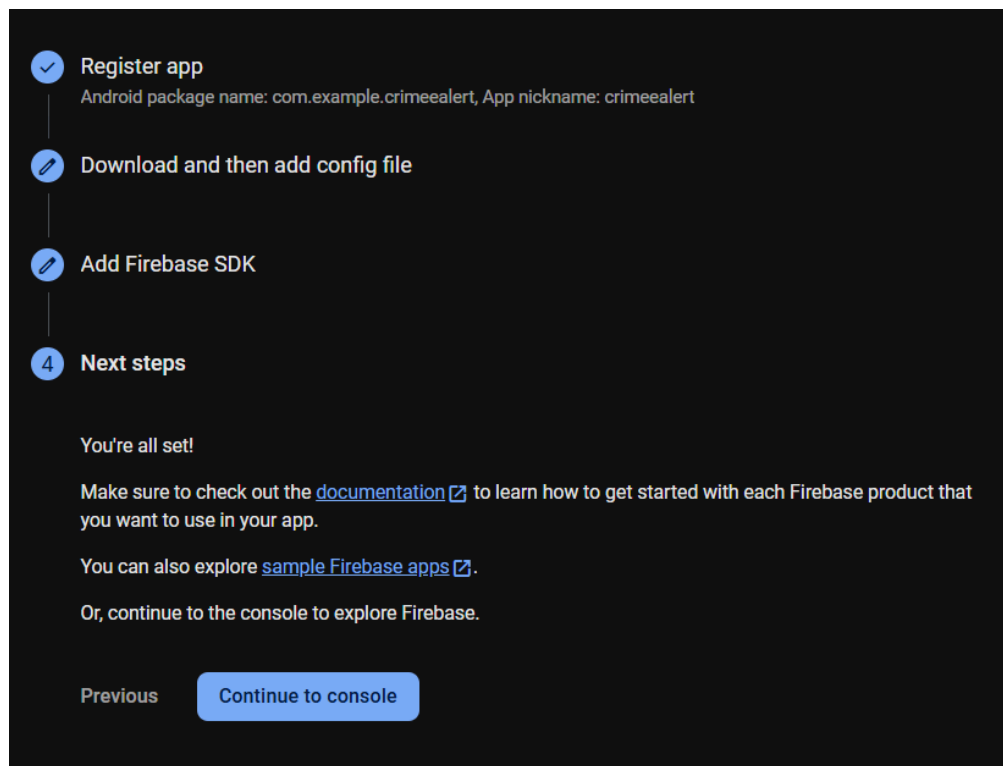
Add the respective code lines in the respective files

```
plugins {  
    id "com.android.application"  
    id "kotlin-android"  
    id "dev.flutter.flutter-gradle-plugin"  
    id("com.google.gms.google-services") version "4.4.1" apply false  
}
```

```
plugins {  
    id "com.android.application"  
    id "kotlin-android"  
    id "dev.flutter.flutter-gradle-plugin"  
    id("com.android.application")  
    id("com.google.gms.google-services")  
}
```

```
dependencies {  
    implementation(platform("com.google.firebase:firebase-bom:32.7.4"))  
    implementation("com.google.firebase:firebase-analytics")  
}
```

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.



The screenshot shows the 'Next steps' section of the Firebase console. It features a vertical list of steps: 'Register app' (completed with a checkmark), 'Download and then add config file' (with a download icon), 'Add Firebase SDK' (with a code icon), and 'Next steps' (highlighted with a blue circle and the number 4). Below the steps, a message says 'You're all set!' followed by links to 'documentation' and 'sample Firebase apps'. At the bottom, there are 'Previous' and 'Continue to console' buttons.

✓ Register app
Android package name: com.example.crimeealert, App nickname: crimeealert

Download and then add config file

Add Firebase SDK

4 Next steps

You're all set!

Make sure to check out 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.

Previous Continue to console

On pressing continue, we are finally done with setting up Firebase for Android. We follow similar steps for setting it up on iOS.

Conclusion:

Setting up Firebase in a Flutter project is a straightforward process that enables developers to leverage powerful backend services seamlessly. By integrating Firebase, developers gain access to features like authentication, real-time database, cloud storage, and more, enhancing the functionality and scalability of their Flutter apps. The process typically involves adding Firebase SDK dependencies to the Flutter project, configuring the Firebase project settings, and initializing Firebase services within the app. Once set up, developers can utilize Firebase APIs to streamline authentication, store and retrieve data, send notifications, and perform various other tasks, thereby accelerating app development and providing users with a robust and reliable experience.