

Name: Sujal Tekwani

Class: D15B

Roll No: 59

Setting Up Firebase with Flutter for iOS and Android Apps

This document provides a detailed step-by-step guide on how to integrate Firebase with a Flutter project for both iOS and Android platforms. Firebase offers a suite of tools for app development, including analytics, authentication, cloud storage, and more. By following this guide, you will be able to set up Firebase in your Flutter app and start using its features.

Prerequisites

Before starting, ensure you have the following:

Flutter SDK installed on your machine.

Android Studio or Xcode for Android and iOS development, respectively.

A Firebase account (create one at firebase.google.com).

A Flutter project created (`flutter create project_name`).

Step 1: Create a Firebase Project

Go to the Firebase Console.

Click Add Project.

Enter a project name and follow the prompts to create the project.

Once the project is created, you will be redirected to the Firebase project dashboard.

Step 2: Add Firebase to Your Flutter Project

For Android

In the Firebase Console, click the Android icon to add an Android app to your Firebase project.

Enter your app's details:

Android package name: Find this in your `android/app/build.gradle` file under `applicationId`.

App nickname (optional): Add a nickname for your app.

Debug signing certificate SHA-1 (optional): If you need Firebase Authentication or Dynamic Links, add your SHA-1 key.

Click Register App.

Download the google-services.json file and place it in the android/app directory of your Flutter project.

Add the following dependencies to your android/build.gradle file:

```
buildscript {  
    dependencies {  
        classpath 'com.google.gms:google-services:4.3.15' // Use the latest version  
    }  
}
```

Add the following to the bottom of your android/app/build.gradle file:

```
apply plugin: 'com.google.gms.google-services'
```

For iOS

In the Firebase Console, click the iOS icon to add an iOS app to your Firebase project.

Enter your app's details:

iOS bundle ID: Find this in your Xcode project under Bundle Identifier.

App nickname (optional): Add a nickname for your app.

Click Register App.

Download the GoogleService-Info.plist file.

Open your Flutter project in Xcode.

Drag and drop the GoogleService-Info.plist file into the Runner directory in Xcode.

Ensure the file is added to the Runner target.

Add the following to your ios/Podfile:

```
platform :ios, '11.0' # or higher
```

Run pod install in the ios directory to install Firebase dependencies.

Step 3: Add Firebase Dependencies to Flutter

Open your pubspec.yaml file in your Flutter project.

Add the following dependencies under dependencies:

dependencies:

flutter:

 sdk: flutter

 firebase_core: latest_version # Required for Firebase integration

 firebase_analytics: latest_version # Optional: For analytics

 firebase_auth: latest_version # Optional: For authentication

 cloud_firestore: latest_version # Optional: For Firestore database

 firebase_storage: latest_version # Optional: For cloud storage

Run flutter pub get to install the dependencies.

Step 4: Initialize Firebase in Your Flutter App

Open your lib/main.dart file.

Import the Firebase Core package:

```
import 'package:firebase_core/firebase_core.dart';
```

Initialize Firebase in the main function:

dart

Copy

```
void main() async {  
  WidgetsFlutterBinding.ensureInitialized();  
  await Firebase.initializeApp();  
  runApp(MyApp());  
}
```

Step 5: Test Firebase Integration

Run your app on an Android or iOS emulator/device.

Check the Firebase Console to ensure your app is connected and sending data (e.g., analytics events).

Step 6: Use Firebase Services

Now that Firebase is set up, you can start using its services in your Flutter app. For example:

Firebase Authentication: Add user authentication using email/password, Google Sign-In, etc.

Firebase Firestore: Store and retrieve data from a NoSQL database.

Firebase Storage: Upload and download files.

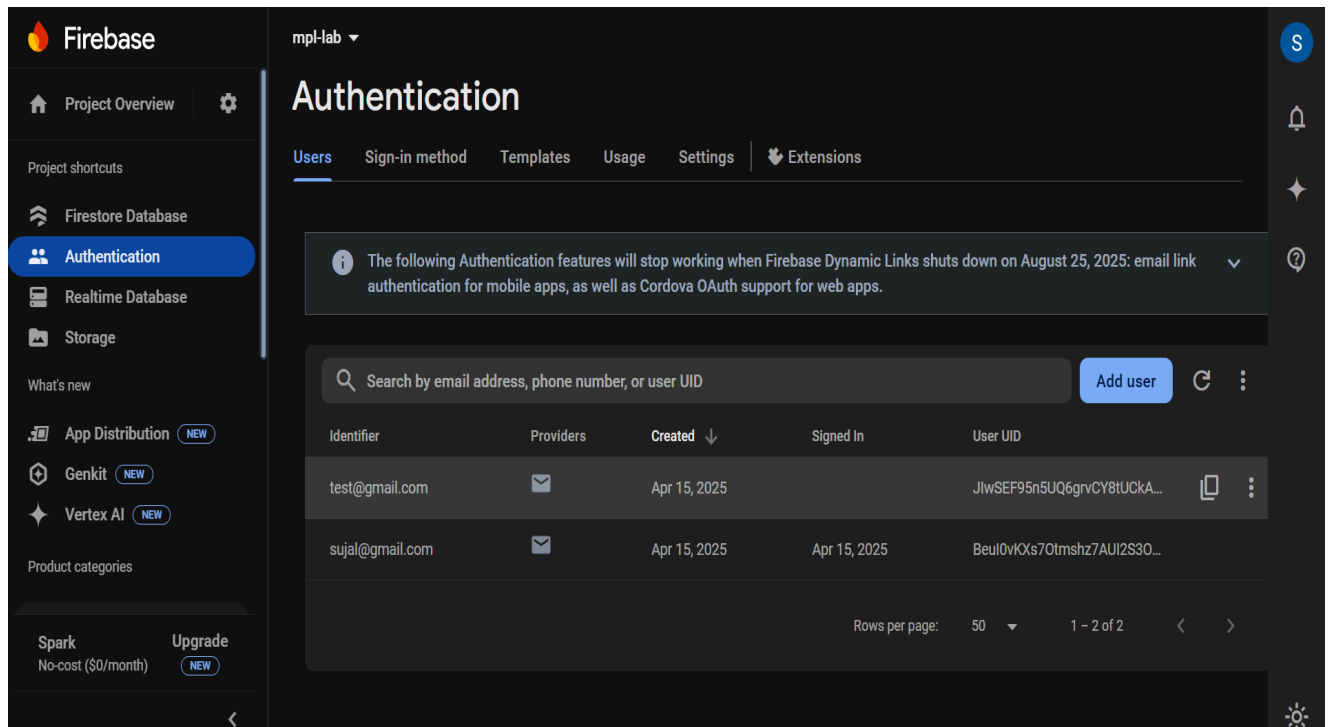
Firebase Analytics: Track user behavior and app usage.

Troubleshooting

Android: If you encounter issues with the google-services.json file, ensure it is placed in the correct directory (android/app).

iOS: If the app crashes on launch, ensure the GoogleService-Info.plist file is added to the Xcode project and the Runner target.

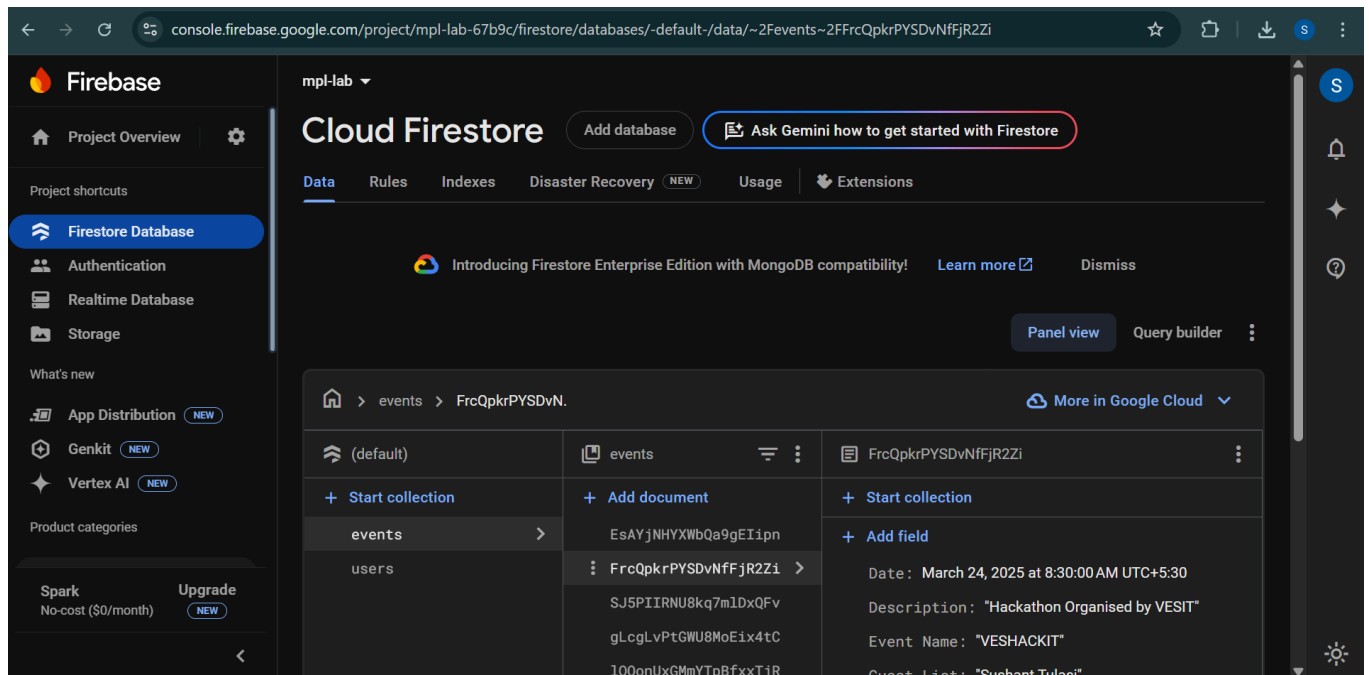
Dependencies: Always use the latest versions of Firebase plugins and ensure there are no version conflicts.



The screenshot displays the Firebase Authentication console for a project named 'mpl-lab'. The left sidebar contains navigation links for Project Overview, Firestore Database, Authentication (selected), Realtime Database, Storage, App Distribution, Genkit, and Vertex AI. The main content area is titled 'Authentication' and includes tabs for Users, Sign-in method, Templates, Usage, Settings, and Extensions. A warning message at the top states: 'The following Authentication features will stop working when Firebase Dynamic Links shuts down on August 25, 2025: email link authentication for mobile apps, as well as Cordova OAuth support for web apps.' Below this is a search bar and an 'Add user' button. A table lists two users:

Identifier	Providers	Created ↓	Signed In	User UID
test@gmail.com		Apr 15, 2025		JlwSEF95n5UQ6grvCY8tUCkA...
sujal@gmail.com		Apr 15, 2025	Apr 15, 2025	Beul0vKXs70tmshz7AUl2S30...

At the bottom right, there is a pagination control showing 'Rows per page: 50' and '1 - 2 of 2'.



Conclusion

You have successfully set up Firebase in your Flutter app for both iOS and Android platforms. You can now leverage Firebase's powerful features to enhance your app's functionality. For more details, refer to the official Firebase Flutter documentation. Replace latest_version with the actual version numbers of the Firebase plugins you are using. You can find the latest versions on pub.dev.