

## **Experiment No : 06**

**Aim:** To connect Flutter UI with firebase Database.

### **Theory:**

Flutter with Firebase together allows developers to create powerful, feature-rich applications efficiently. Here's a theoretical overview of how Flutter and Firebase work together:

### **Flutter:**

Flutter is an open-source UI software development kit created by Google. It's used for building natively compiled applications for mobile, web, and desktop from a single codebase. Some key points about Flutter:

Cross-Platform Development: Flutter allows developers to write code once and deploy it across multiple platforms like Android, iOS, web, and desktop.

Dart Programming Language: Flutter apps are written in the Dart programming language, which is also developed by Google. Dart is known for its simplicity and speed, making it suitable for mobile app development.

Widgets: Flutter uses a reactive-style framework, where UI components are widgets. Everything in Flutter is a widget, from buttons to layouts to animations. This widget-based architecture allows for fast UI rendering and customization.

Hot Reload: One of Flutter's most popular features is hot reload, which allows developers to instantly see the effects of code changes reflected in the app without losing the app state. This speeds up the development process significantly.

Rich UI: Flutter offers a rich set of pre-built widgets and tools for creating beautiful, responsive user interfaces. Developers can also customize these widgets or create their own from scratch.

### **Firebase:**

Firebase is a mobile and web application development platform developed by Firebase, Inc., which was acquired by Google in 2014. Firebase provides a suite of tools and services that help developers build, improve, and grow their apps. Some Firebase's key features are:

Realtime Database: Firebase offers a NoSQL cloud database that enables developers to store and sync data across multiple clients in real-time. It's suitable for applications requiring live data updates, such as chat apps or collaborative tools.

Authentication: Firebase Authentication provides easy-to-use SDKs and ready-to-use UI libraries for authenticating users using email/password, phone numbers, social media logins (Google, Facebook, Twitter), and more.

Cloud Firestore: Firestore is Firebase's flexible, scalable NoSQL cloud database that allows for more complex queries, hierarchical data structures, and better scalability compared to the Realtime Database.

Cloud Functions: Firebase Cloud Functions allow developers to run backend code in response to events triggered by Firebase features and HTTPS requests. It enables developers to extend the functionality of their Firebase apps without managing servers.

Machine Learning: Firebase ML provides easy-to-use APIs for integrating machine learning into mobile and web apps. Developers can use pre-trained models or deploy their custom models for tasks like image labeling, text recognition, and language translation.

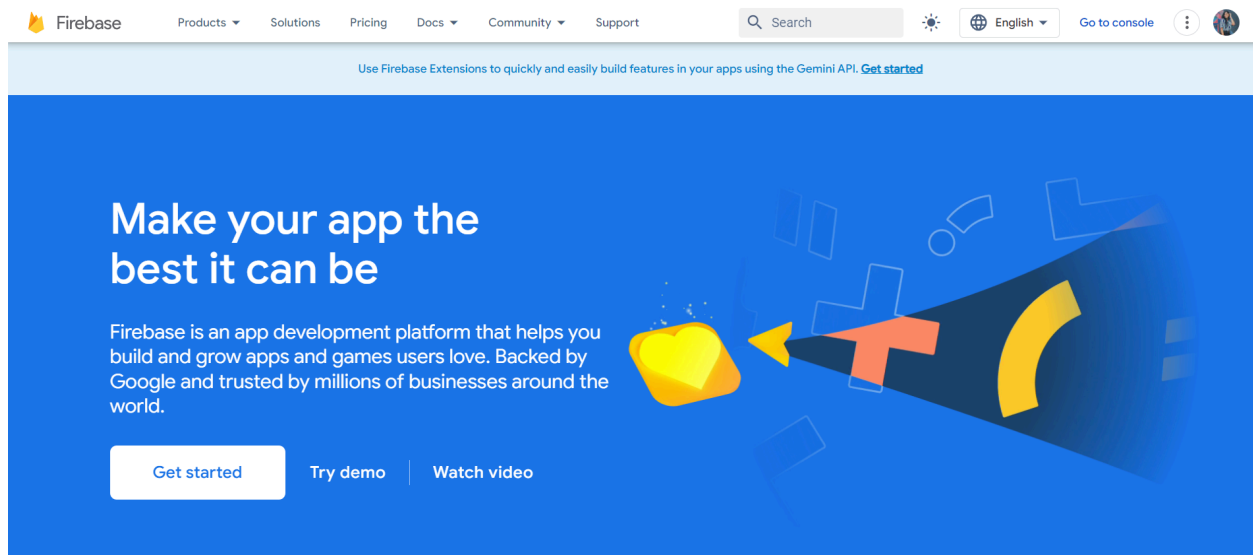
## Steps to connect your Flutter app with Firebase:

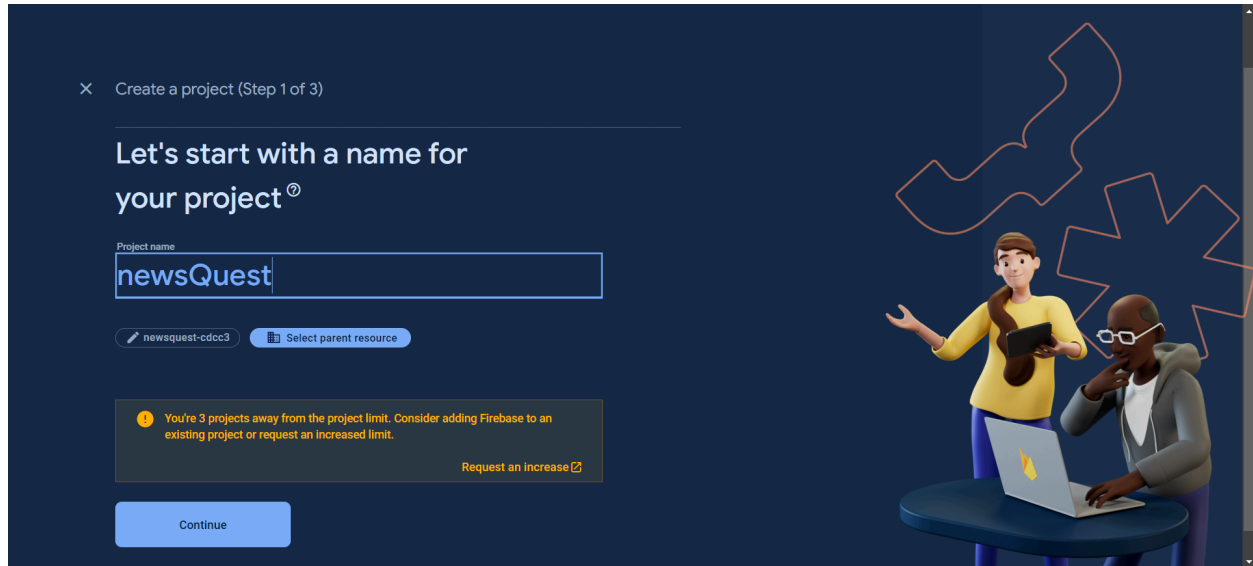
### Step 1: Create a Firebase Project

Go to the Firebase Console.

Click on "Add project" and follow the prompts to create a new project.

Once the project is created, you'll be redirected to the project dashboard.

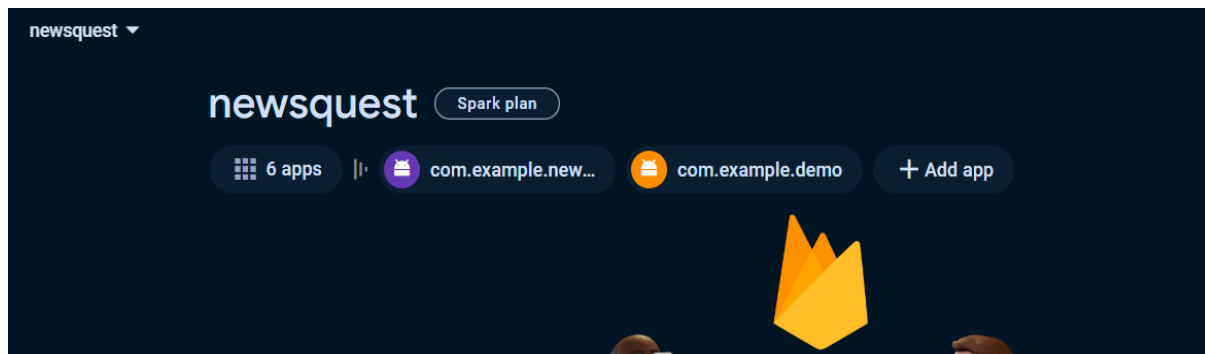


**Step 2:** Add your Flutter app to the Firebase project

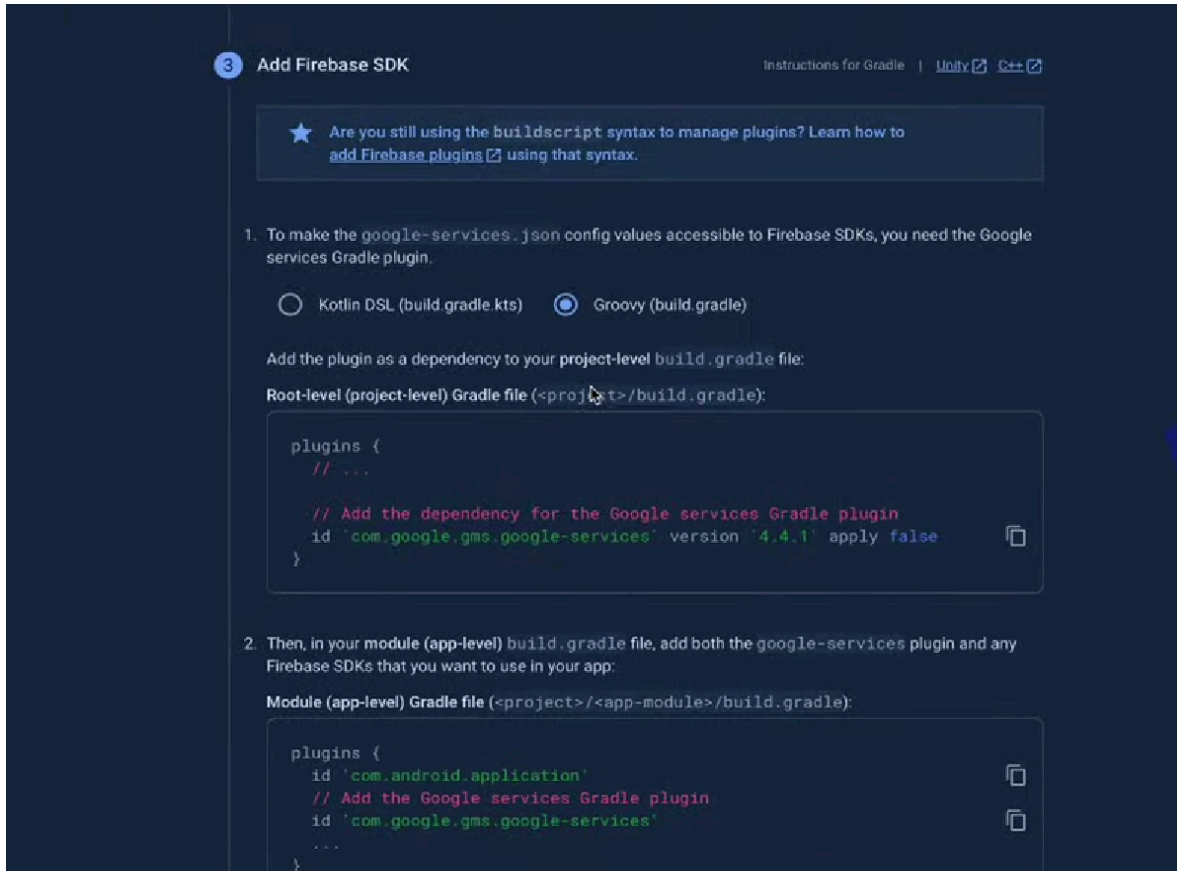
Click on the "Add app" button (iOS, Android, web) in the Firebase console.

Follow the instructions to register your app with Firebase. You'll need to provide your app's package name (for Android), bundle ID (for iOS), or other necessary information.

Download the google-services.json (for Android) or GoogleService-Info.plist (for iOS) configuration files and place them in your Flutter app's android/app or ios/Runner directories respectively.







**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` file:

Root-level (project-level) Gradle file (`<project>/build.gradle`):

```
plugins {
    // ...

    // Add the dependency for the Google services Gradle plugin
    id 'com.google.gms.google-services' version '4.4.1' apply false
}
```

2. Then, in your module (app-level) `build.gradle` 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`):

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

### Step 3: Set up Flutter Firebase dependencies

Open your Flutter project in your preferred code editor.

Add the Firebase dependencies to your `pubspec.yaml` file:

```
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: ^1.0.2
  firebase_core: 2.27.0
  cloud_firestore: 4.15.8
  firebase_auth: ^4.17.8
  google_sign_in: ^6.1.5
```

Run `flutter pub get` in the terminal to install the new dependencies.

### Step 4: Initialize Firebase in your Flutter app

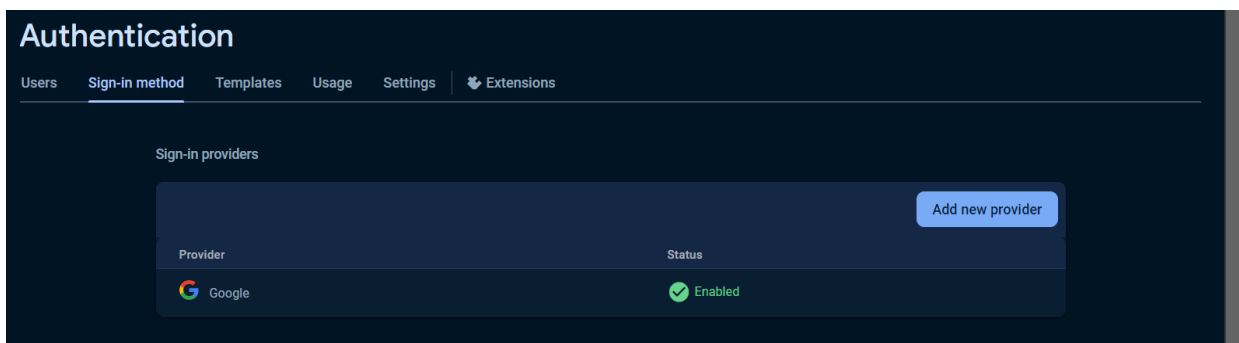
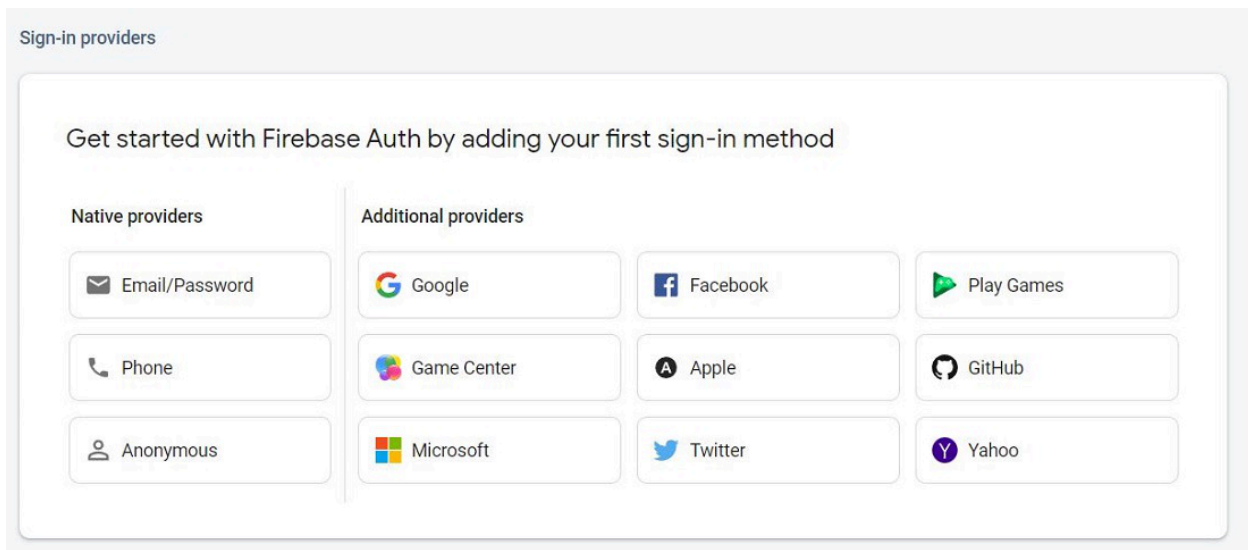
Import the Firebase packages at the top of your Dart file:

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

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    | options: DefaultFirebaseOptions.currentPlatform,
  );
}
```

**Step 5:** Set up Authentication or other Firebase services

Select the choice of authentication like email/password authentication or google auth authentication





```
32
33     try {
34         // check if both password and confirm password is same
35         if (passwordController.text == confirmPasswordController.text) {
36             await FirebaseAuth.instance.createUserWithEmailAndPassword(
37                 email: emailController.text,
38                 password: passwordController.text,
39             );
40         } else {
41             //show error password dont match
42             genericErrorMessage("Password don't match!");
43         }
44     }
```

**Conclusion:** We have now connected our flutter app to Firebase, developers often leverage Firebase's backend services for features like authentication, database storage, cloud functions, and analytics, while using Flutter for the frontend UI and business logic. This combination allows for rapid development of cross-platform apps with powerful backend functionality.