Name: Soham H. Nimbalkar **Expt No: 06** **Roll no: 49**

Experiment: 06

Aim: To apply navigation, routing and gestures in Flutter App.

Theory:

Firebase is a Backend-as-a-Service — BaaS — that started as a YC11 startup and grew up into a next-generation app-development platform on Google Cloud Platform.

Firebase frees developers to focus on crafting fantastic user experiences. You don't need to manage servers. You don't need to write APIs. Firebase is your server, your API and your datastore, all written so generically that you can modify it to suit most needs. Yeah, you'll occasionally need to use other bits of the Google Cloud for your advanced applications. Firebase can't be everything to everybody. But it gets pretty close.

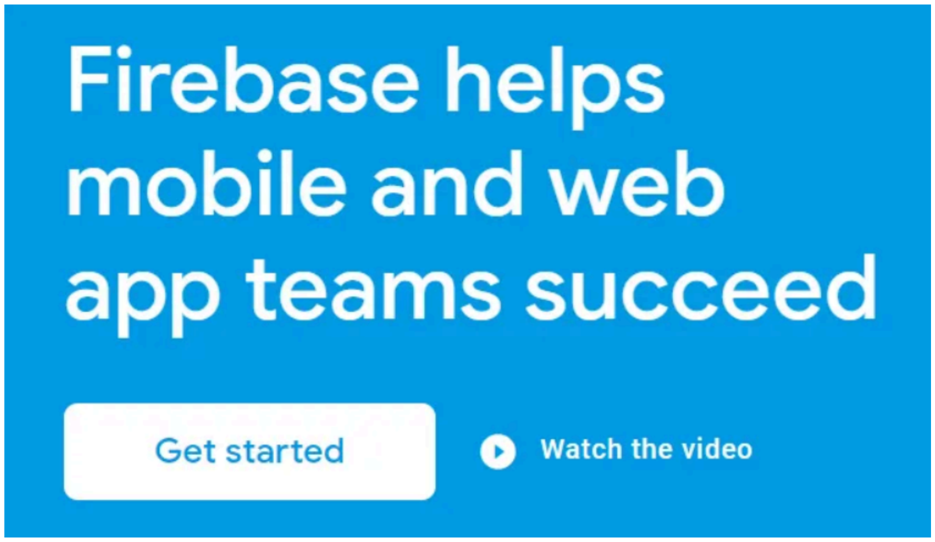
Top Features of Firebase:

- Build apps fast, without managing infrastructure
    - Firebase gives you functionality like analytics, databases, messaging and crash reporting so you can move quickly and focus on your users.
- Backed by Google, trusted by top apps
    - Firebase is built on Google infrastructure and scales automatically, for even the largest apps.
- One platform, with products that work better together
    - Firebase products work great individually but share data and insights, so they work even better together.

Steps to setup firebase:

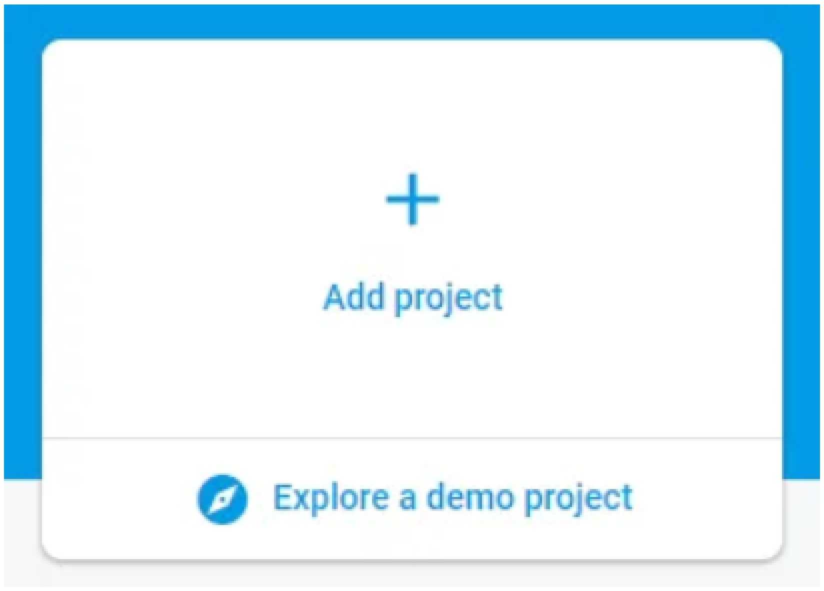
1. After setting up & creating a simple Flutter app, navigate to Firebase Official website.

Click on Get started.

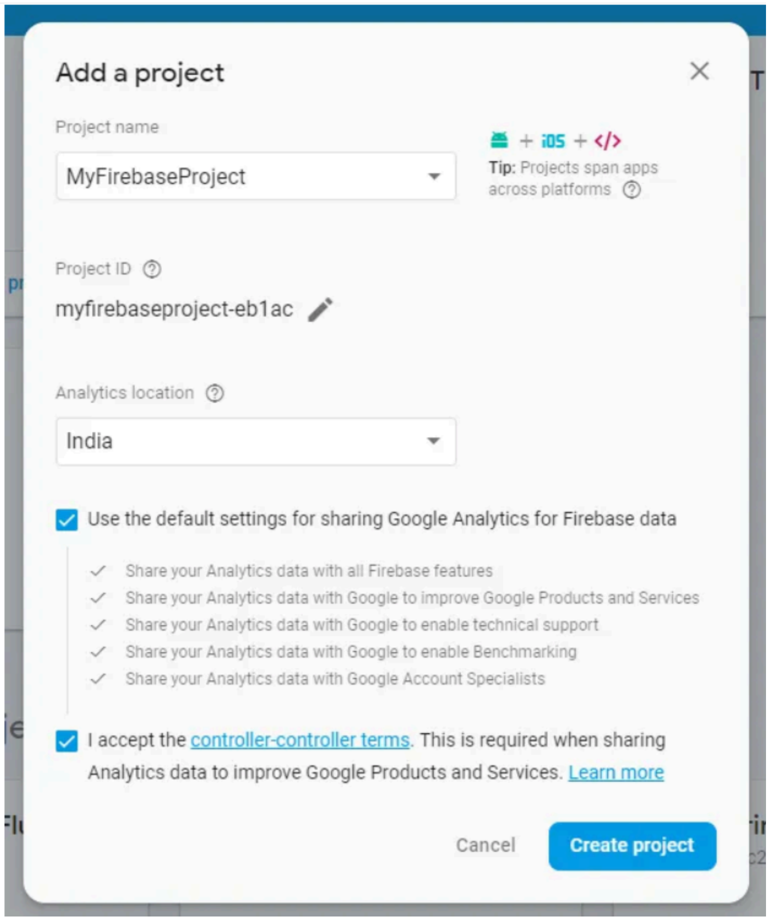


Name: Soham H. Nimbalkar **Expt No: 06** **Roll no: 49**

Create a new project by tapping on the Add project box.



Enter the Project name & select Analytics location.
Analytics location represents the country/region of your organization and sets the currency for revenue reporting.



Add a project

Project name: MyFirebaseProject

Tip: Projects span apps across platforms

Project ID: myfirebaseproject-eb1ac

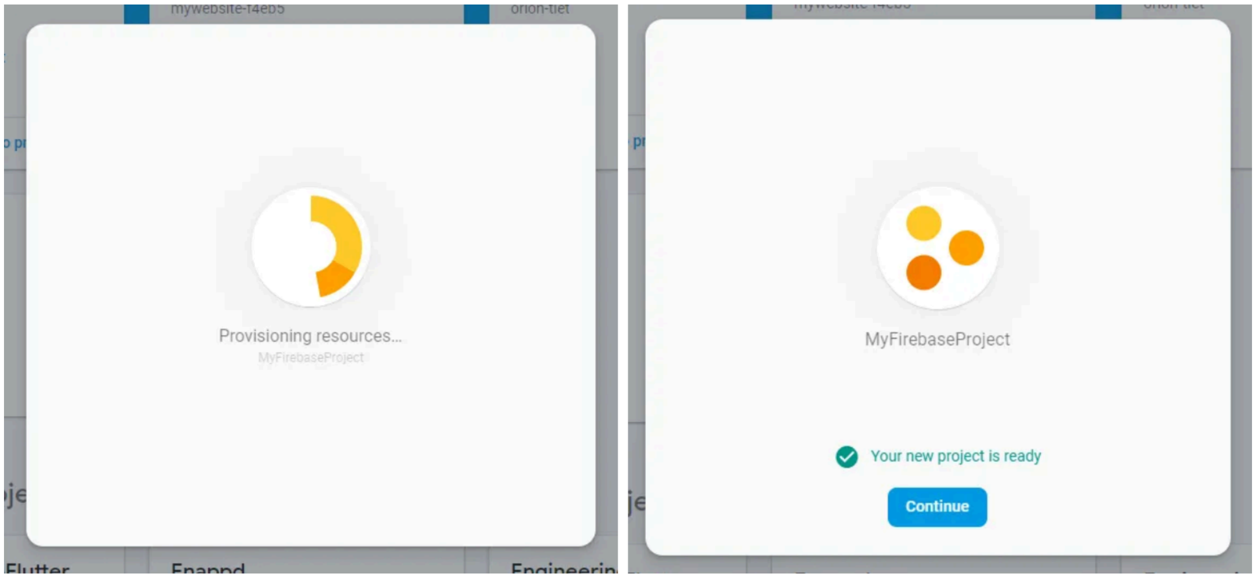
Analytics location: India

✓ Use the default settings for sharing Google Analytics for Firebase data

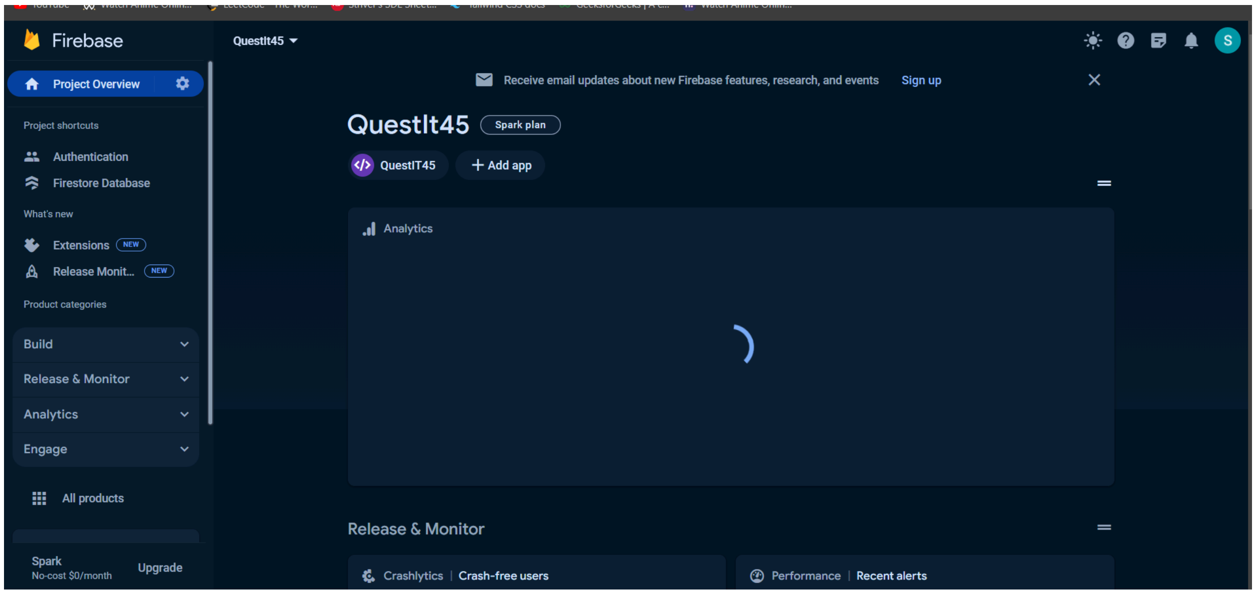
- ✓ Share your Analytics data with all Firebase features
- ✓ Share your Analytics data with Google to improve Google Products and Services
- ✓ Share your Analytics data with Google to enable technical support
- ✓ Share your Analytics data with Google to enable Benchmarking
- ✓ Share your Analytics data with Google Account Specialists

✓ I accept the controller-controller terms. This is required when sharing Analytics data to improve Google Products and Services. Learn more

Tap on Create project. Following this, Firebase will set up a new project for you. This will take just a minute.



After the project is created successfully, Firebase will show a prompt saying Your new project is ready. Click on Continue to complete the flow.



Add FlutterFire plugins

Flutter uses plugins to provide access to a wide range of platform-specific services, such as Firebase APIs. Plugins include platform-specific code to access services and APIs on each platform.

Firebase is accessed through a number of different libraries, one for each Firebase product (for example, database, authentication, analytics, or storage). Flutter provides a set of Firebase plugins, which is collectively called FlutterFire.

Since Flutter is a multi-platform SDK, each FlutterFire plugin is applicable for both iOS and Android. So, if you add any FlutterFire plugin to your Flutter app, it will be used by both the iOS and Android versions of your Firebase app.

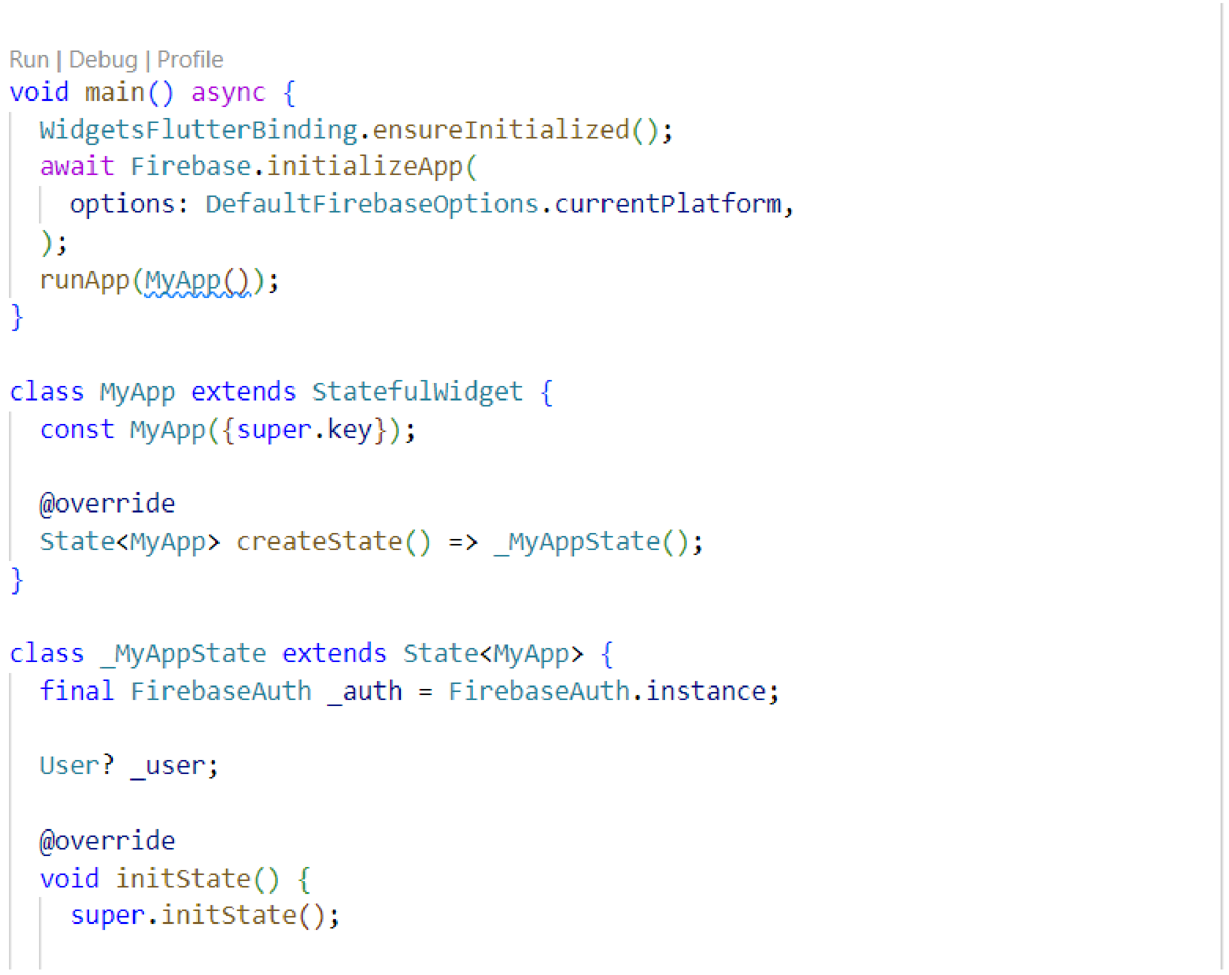
Available FlutterFire plugins
cloud_firestore
cloud_functions
firebase_admob
firebase_analytics
firebase_auth
firebase_core
firebase_crashlytics
firebase_database
firebase_dynamin_links
firebase_messaging
firebase_ml_vision
firebase_performance
firebase_remote_config
firebase_storage
Steps to add FlutterFire plugins to your app
From the root directory of your Flutter app, open your pubspec.yaml file.
Add the FlutterFire plugin for the Firebase Core SDK. All Flutter apps, both iOS and Android versions, require the firebase_core plugin.

```
dependencies:
  flutter:
    sdk: flutter
  firebase_core: ^0.2.5  # add dependency for Firebase Core
```

Add additional FlutterFire plugins to use specific Firebase products.

Run flutter packages get.
Running the application
Connect your Emulator or physical Android device to test the application.
Click on Build & Run.

Code:



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

class MyApp extends StatefulWidget {
  const MyApp({super.key});

  @override
  State<MyApp> createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  final FirebaseAuth _auth = FirebaseAuth.instance;

  User? _user;

  @override
  void initState() {
    super.initState();
```

Firebase_options.dart:

```
// File generated by FlutterFire CLI.
// ignore_for_file: lines_longer_than_80_chars,
avoid_classes_with_only_static_members
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
import 'package:flutter/foundation.dart'
    show defaultTargetPlatform, kIsWeb, TargetPlatform;
```

```dart
/// Default [FirebaseOptions] for use with your Firebase apps.
///
/// Example:
/// ```dart
/// import 'firebase_options.dart';
/// // ...
/// await Firebase.initializeApp(
///   options: DefaultFirebaseOptions.currentPlatform,
/// );
/// ```
class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    if (kIsWeb) {
      throw UnsupportedError(
        'DefaultFirebaseOptions have not been configured for web - '
        'you can reconfigure this by running the FlutterFire CLI again.',
      );
    }
    switch (defaultTargetPlatform) {
      case TargetPlatform.android:
        return android;
      case TargetPlatform.iOS:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for ios - '
          'you can reconfigure this by running the FlutterFire CLI
again.',
        );
      case TargetPlatform.macOS:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for macos - '
          'you can reconfigure this by running the FlutterFire CLI
again.',
        );
      case TargetPlatform.windows:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for windows - '
          'you can reconfigure this by running the FlutterFire CLI
again.',
        );
```

```
      case TargetPlatform.linux:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for linux - '
          'you can reconfigure this by running the FlutterFire CLI
again.',
        );
      default:
        throw UnsupportedError(
          'DefaultFirebaseOptions are not supported for this platform.',
        );
    }
  }

  static const FirebaseOptions android = FirebaseOptions(
    apiKey: 'AIzaSyDiO4RHMie4Ty1_SyHmRe3x1sGMqm6vw7M',
    appId: '1:979696473409:android:1cfc7b8900813aee8a3dc0',
    messagingSenderId: '979696473409',
    projectId: 'agrifarm-7d57b',
    storageBucket: 'agrifarm-7d57b.appspot.com',
  );
}
```

Conclusion:
Hence we have successfully configured firebase project and integrated it with our flutter UI to make the app dynamic and to store data.