Name: Atif Ansari Roll no: 04 Class: D15B

MPL Experiment - 6

Aim: To set up Firebase with Flutter for iOS and Android Apps.

Theory: Firebase is a Backend-as-a-Service (BaaS) platform by Google that provides serverless backend functionalities such as database management, authentication, cloud storage, and real-time analytics. It allows developers to build scalable and real-time applications without managing dedicated backend servers.

Key Features of Firebase for Flutter:

- Firebase Authentication: Supports login via Google, Email/Password, and Social media.
- Cloud Firestore: A NoSQL database that provides real-time data synchronization.
- Firebase Realtime Database: Syncs data across multiple users instantly.
- Firebase Cloud Messaging: Enables push notifications for mobile applications.
- Firebase Storage: Stores and retrieves user-generated media like images and videos.
- Firebase Analytics: Tracks user interactions

within the app for insights and improvements.

Other advantages of using firebase are:

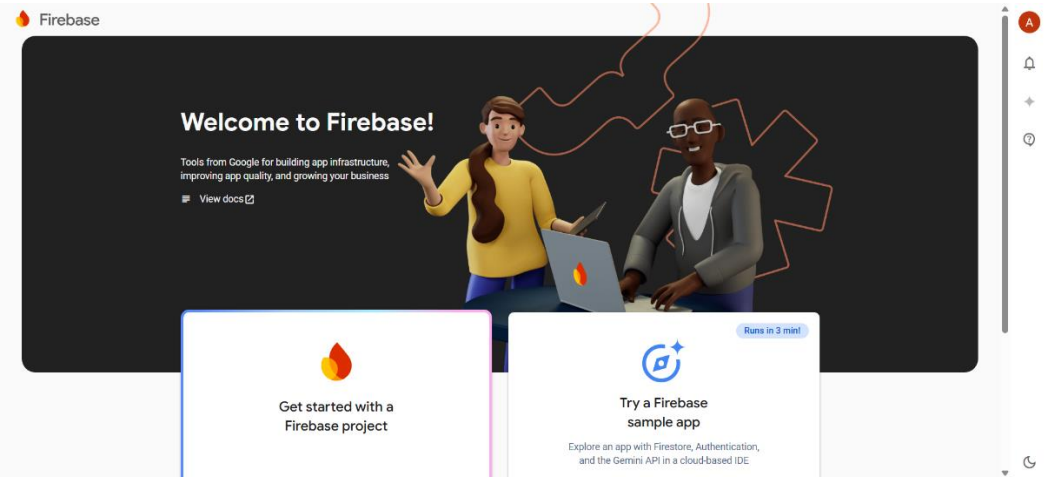
1. Cross-Platform Support.
2. Easy Integration.
3. Serverless Architecture.
4. Real-time Capabilities.

Conclusion: In this experiment, we successfully configured and integrated Firebase with a flutter application for both Android and iOS. This setup established a real-time backend connection, enabling services such as authentication, database management, and cloud storage.

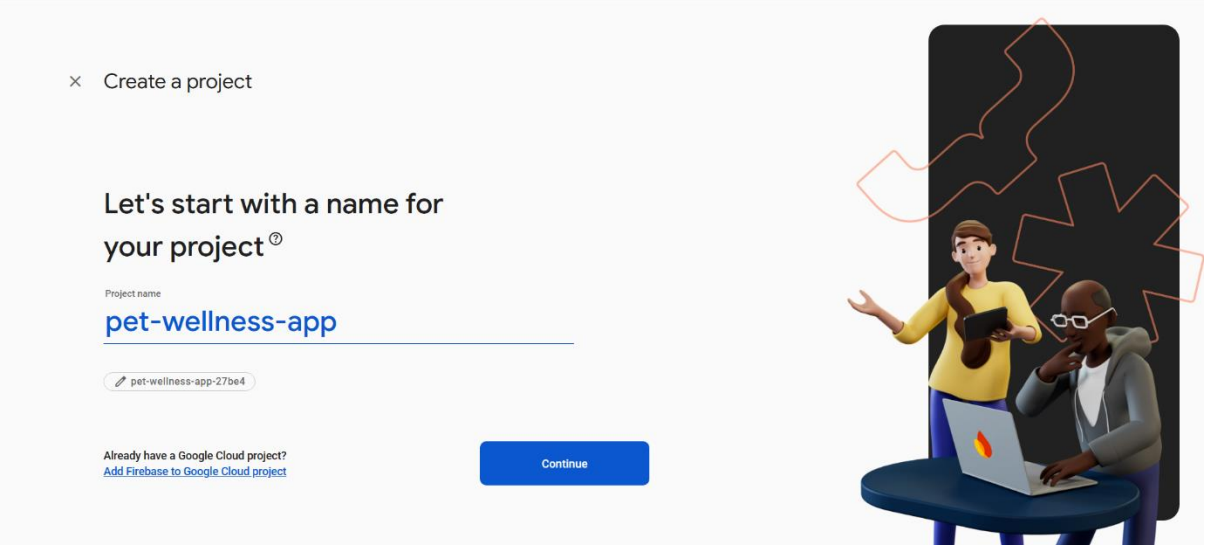
With firebase integrated, the app can now:

- Support user authentication.
- Store and manage data using firestore or Realtime Database.
- Send push notifications using firebase Cloud Messaging.
- Upload and retrieve media files using Firebase Storage.

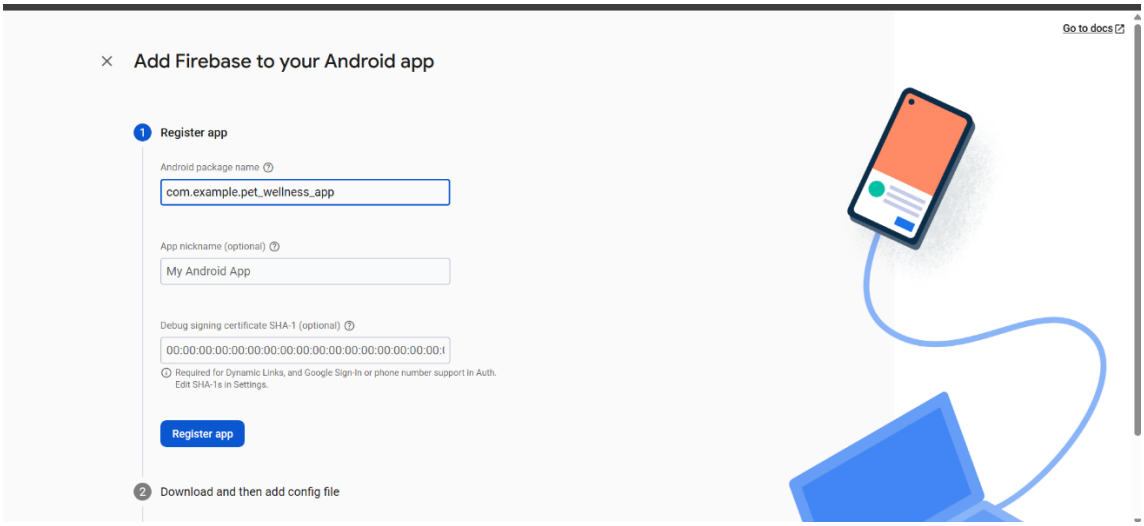
Step 1: First, log in with your Google account to manage your Firebase projects.



Step 2: From within the Firebase dashboard, select the Create new project button and give it a name:

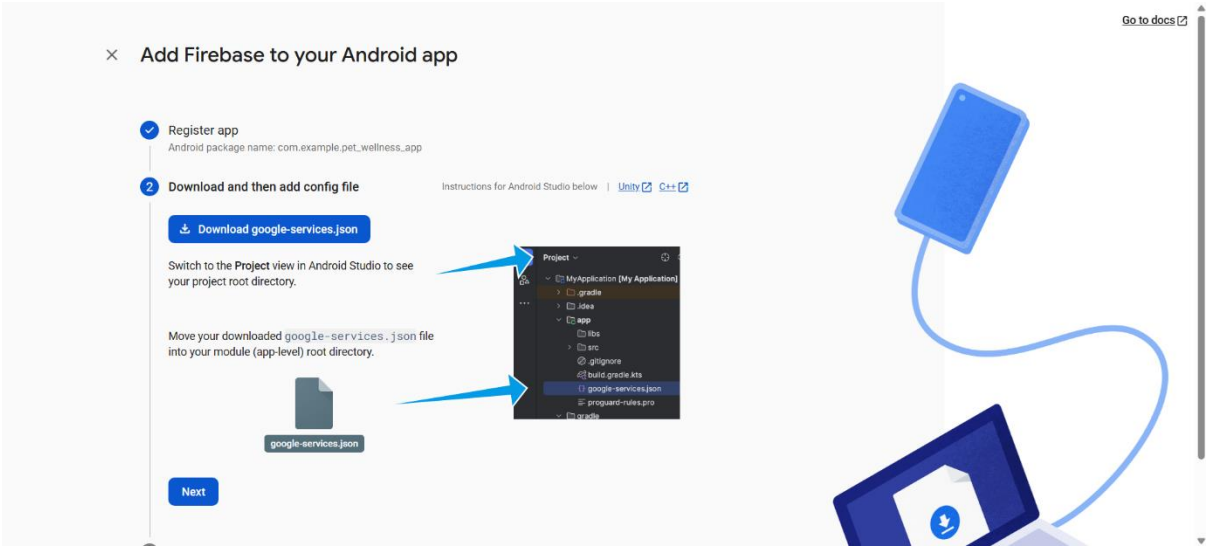


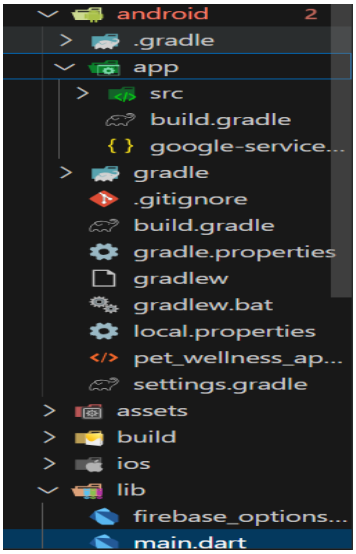
Step 3: In order to add Android support to our Flutter application, select the Android logo from the dashboard. This brings us to the following screen:



Step 4: 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:





Step 5: Open android/build.gradle in your code editor and modify it to include the following:

```
buildscript {
    repositories {
        google()
        mavenCentral()
    }
    dependencies {
        // START: FlutterFire Configuration
        classpath 'com.google.gms:google-services:4.3.15'
        // END: FlutterFire Configuration
        // Use a version of the Android Gradle Plugin that supports your setup.
        // Check https://developer.android.com/studio/releases/gradle-plugin for the latest stable version.
        classpath 'com.android.tools.build:gradle:8.3.0'
        id 'com.google.gms.google-services' version '4.4.2' apply false
    }
}
```

Step 6: Finally, update the app level file at android/app/build.gradle to include the following:

```
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply plugin: 'com.google.gms.google-services'

android {
    namespace "com.example.pet_wellness_app"
    compileSdkVersion flutter.compileSdkVersion

    defaultConfig {
        applicationId "com.example.pet_wellness_app"
        minSdkVersion 23
        targetSdkVersion flutter.targetSdkVersion
        versionCode flutter.versionCode
        versionName flutter.versionName
    }
```

Step 7: 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.

From here, run your application on an Android device or simulator. If everything has worked correctly, you should get the following message in the dashboard: