Name : Yajat Bhasin **Div:D15B** **Roll No:5**

Experiment No. 6

Aim:

To Set Up Firebase with Flutter for iOS and Android Apps

Theory:

What is Firebase?

Firebase is an app development platform that helps you build and grow apps and games users love.Alongside its flagship features like the real-time database, cloud storage, and authentication, Firebase provides additional functionalities such as Cloud Firestore, a scalable database for real-time updates and offline data persistence. Cloud Functions enable developers to extend app functionality with custom backend code triggered by events. Cloud Messaging facilitates cross-platform notifications to engage users effectively, while Authentication offers secure user management with support for various authentication methods

Firebase Features:

1. Realtime Database
2. Cloud Storage
3. Authentication
4. Cloud Firestore
5. Cloud Functions
6. Cloud Messaging (FCM)
7. Performance Monitoring
8. App Distribution
9. Dynamic Links
10. Test Lab
11. Predictions

Implementation:

Creating a New Flutter Project

This tutorial will require the creation of an example Flutter app.

Once you have your environment set up for Flutter, you can run the following to create a new application:

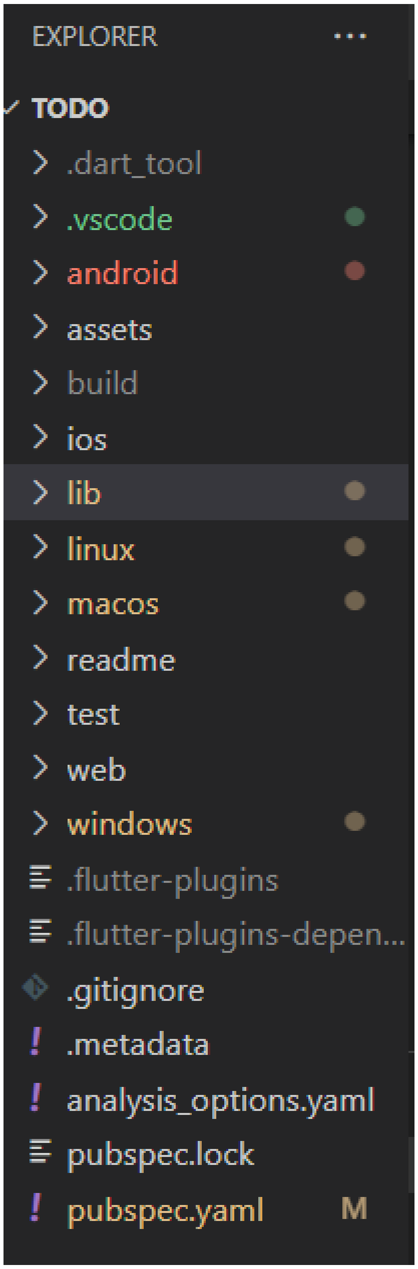
```
flutter create ToDo
```

Navigate to the new project directory:

```
cd ToDo
```

Using `flutter create` will produce a demo application that will display the number of times a button is clicked.

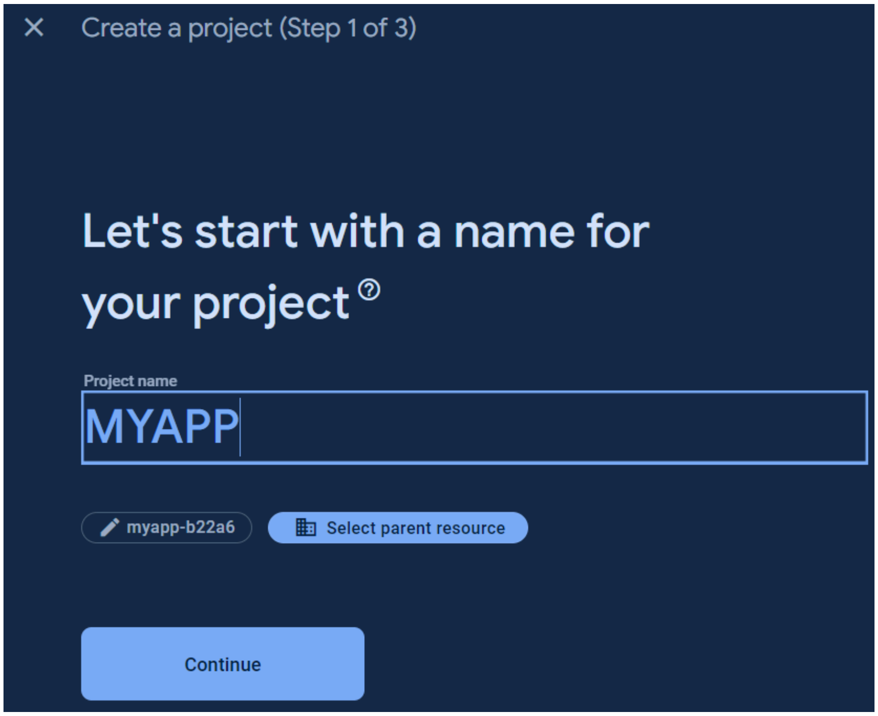
Now that we’ve got a Flutter project up and running, we can add



```
EXPLORER ···
✓ TODO
  > .dart_tool
  > .vscode            ●
  > android            ●
  > assets
  > build
  > ios
  > lib                ●
  > linux              ●
  > macos              ●
  > readme
  > test
  > web
  > windows            ●
  .flutter-plugins
  .flutter-plugins-depen...
  .gitignore
  ! .metadata
  ! analysis_options.yaml
  pubspec.lock
  ! pubspec.yaml       M
```

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:



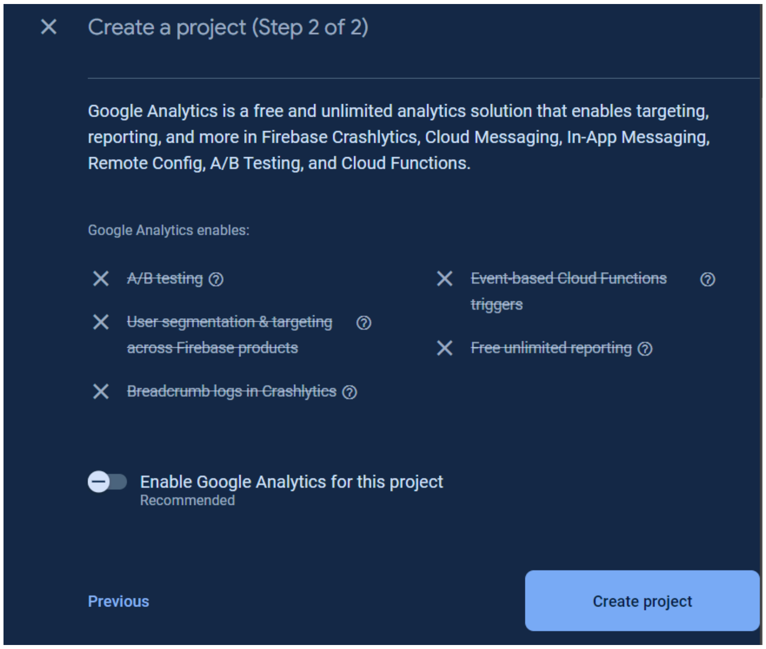
Create a project (Step 1 of 3)

Let's start with a name for your project

Project name

MYAPP

Next, we're given the option to enable Google Analytics. This tutorial will not require Google Analytics, but you can also choose to add it to your project.



Create a project (Step 2 of 2)

Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, and Cloud Functions.

Google Analytics enables:

- ✗ ~~A/B testing~~
- ✗ ~~User segmentation & targeting across Firebase products~~
- ✗ ~~Breadcrumb logs in Crashlytics~~
- ✗ ~~Event-based Cloud Functions triggers~~
- ✗ ~~Free unlimited reporting~~

○ Enable Google Analytics for this project
Recommended

After pressing Continue, your project will be created and resources will be provisioned. You will then be directed to the dashboard for the new project.

Registering the App

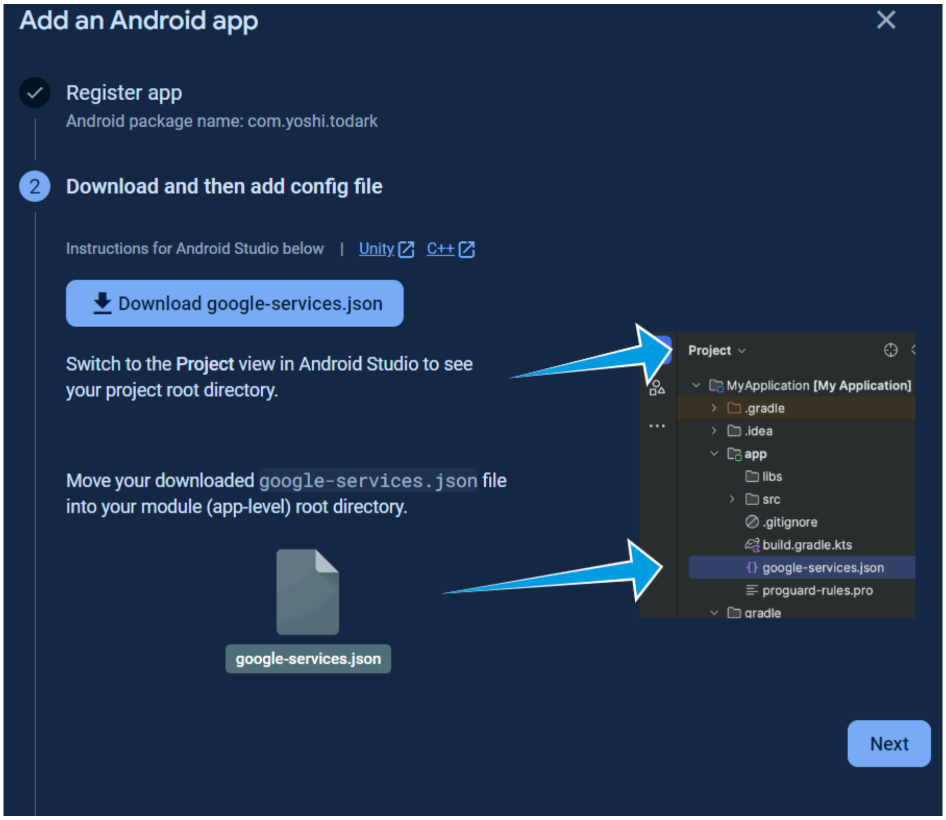
In order to add Android support to our Flutter application, select the Android logo from the dashboard. This brings us to the following screen:

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

Downloading the Config File

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:



Add an Android app

✓ Register app

Android package name: com.yoshi.todark

2 Download and then add config file

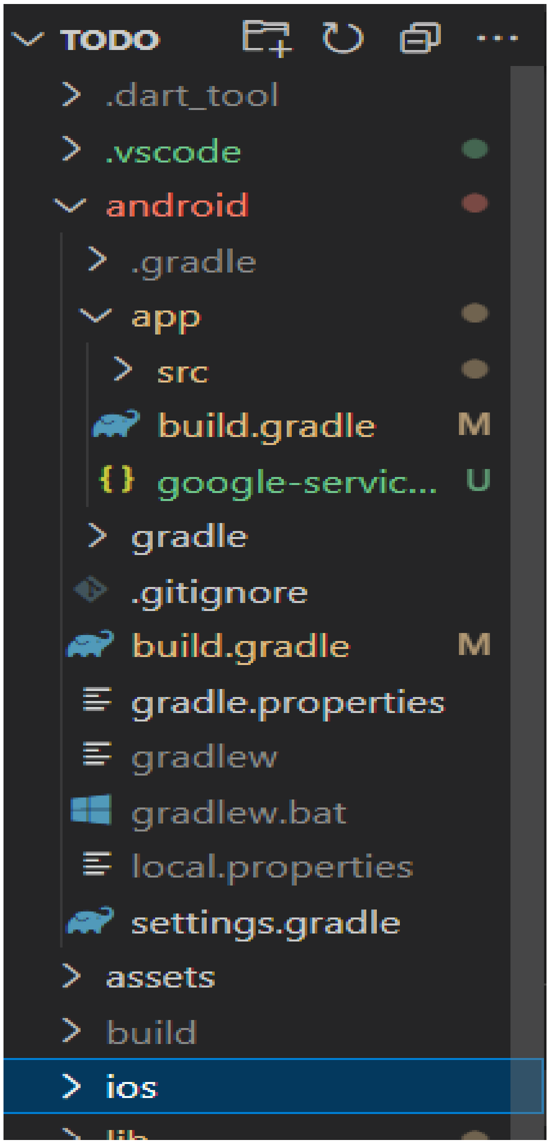
Instructions for Android Studio below | Unity C++

Download google-services.json

Switch to the **Project** view in Android Studio to see your project root directory.

Move your downloaded `google-services.json` file into your module (app-level) root directory.

Added google-services. Json



```
TODO
  .dart_tool
  .vscode
  android
    .gradle
    app
      src
      build.gradle          M
      google-servic...      U
    gradle
    .gitignore
    build.gradle            M
    gradle.properties
    gradlew
    gradlew.bat
    local.properties
    settings.gradle
  assets
  build
  ios
```

Next, move the `google-services.json` file to the `android/app` directory within the Flutter project. **Adding the Firebase SDK**

We’ll now need to update our Gradle configuration to include the Google Services plugin. Open `android/build.gradle` in your code editor and modify it to include the following:

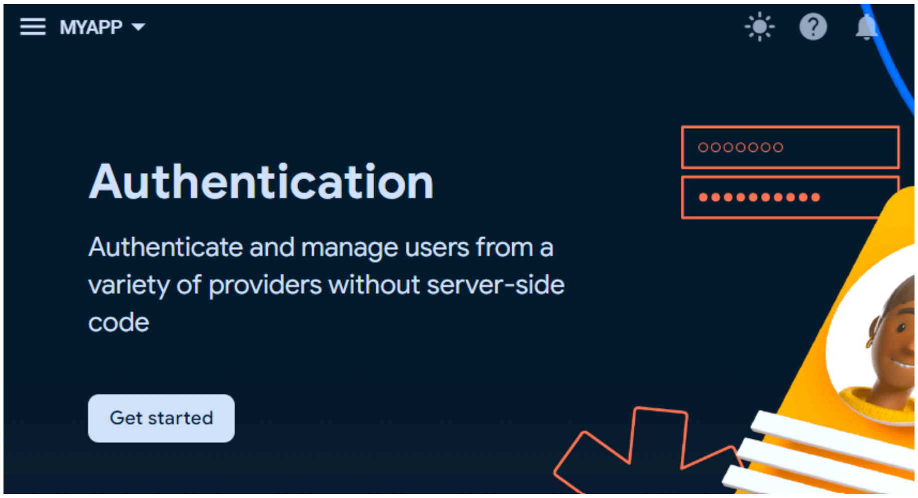
android/buiild.gradle

```
buildscript {
  repositories {
    // Check that you have the following line (if not, add it):
    google() // Google's Maven repository
  }
  dependencies {
    ...
    // Add this line
    classpath 'com.google.gms:google-services:4.4.1'
  }
}

allprojects {
  ...
  repositories {
    // Check that you have the following line (if not, add it):


    google() // Google's Maven repository
    ...
  }
}
```

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

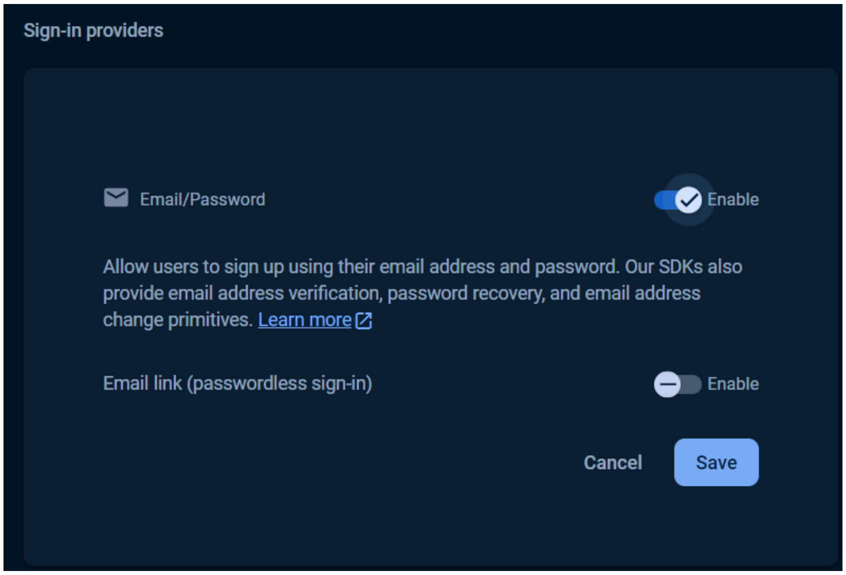
```
android/app/build.gradle
apply plugin: 'com.android.application'
// Add this line
apply plugin: 'com.google.gms.google-services'

dependencies {
 // Import the Firebase BoM
 implementation platform('com.google.firebase:firebase-bom:28.0.0')

 // Add the dependencies for any other desired Firebase products
 // https://firebase.google.com/docs/android/setup#available-libraries
}
```

Authentication



Authentication

Authenticate and manage users from a variety of providers without server-side code



Sign-in providers

Email/Password ☑ Enable

Allow users to sign up using their email address and password. Our SDKs also provide email address verification, password recovery, and email address change primitives. Learn more

Email link (passwordless sign-in) □ Enable

Conclusion:

Flutter offers seamless integration with Firebase through the official FlutterFire set of libraries. The combination of Flutter and Firebase streamlines development, enabling a focus on user experiences while Firebase manages backend infrastructure. This partnership facilitates the creation of high-quality cross-platform apps, making it a top choice for modern development projects.