MAD & PWA LAB
EXP- 6

Name: Himanshu Lohote **Class: D15A**
Roll No. 32

Set Up Firebase with Flutter for iOS and Android Apps

Firebase is a great backend solution for anyone that wants to use authentication, databases, cloud functions, ads, and countless other features within an app.

In this article, you will create a Firebase project for iOS and Android platforms using Flutter.

Prerequisites

To complete this tutorial, you will need:

- A Google account to use Firebase.
- Developing for iOS will require XCode.
- To download and install Flutter.
- To download and install Android Studio and Visual Studio Code.
- It is recommended to install plugins for your code editor:
  - `Flutter` and `Dart` plugins installed for Android Studio.
  - `Flutter` extension installed for Visual Studio Code.

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 fooddeliveryapp
```

Navigate to the new project directory:

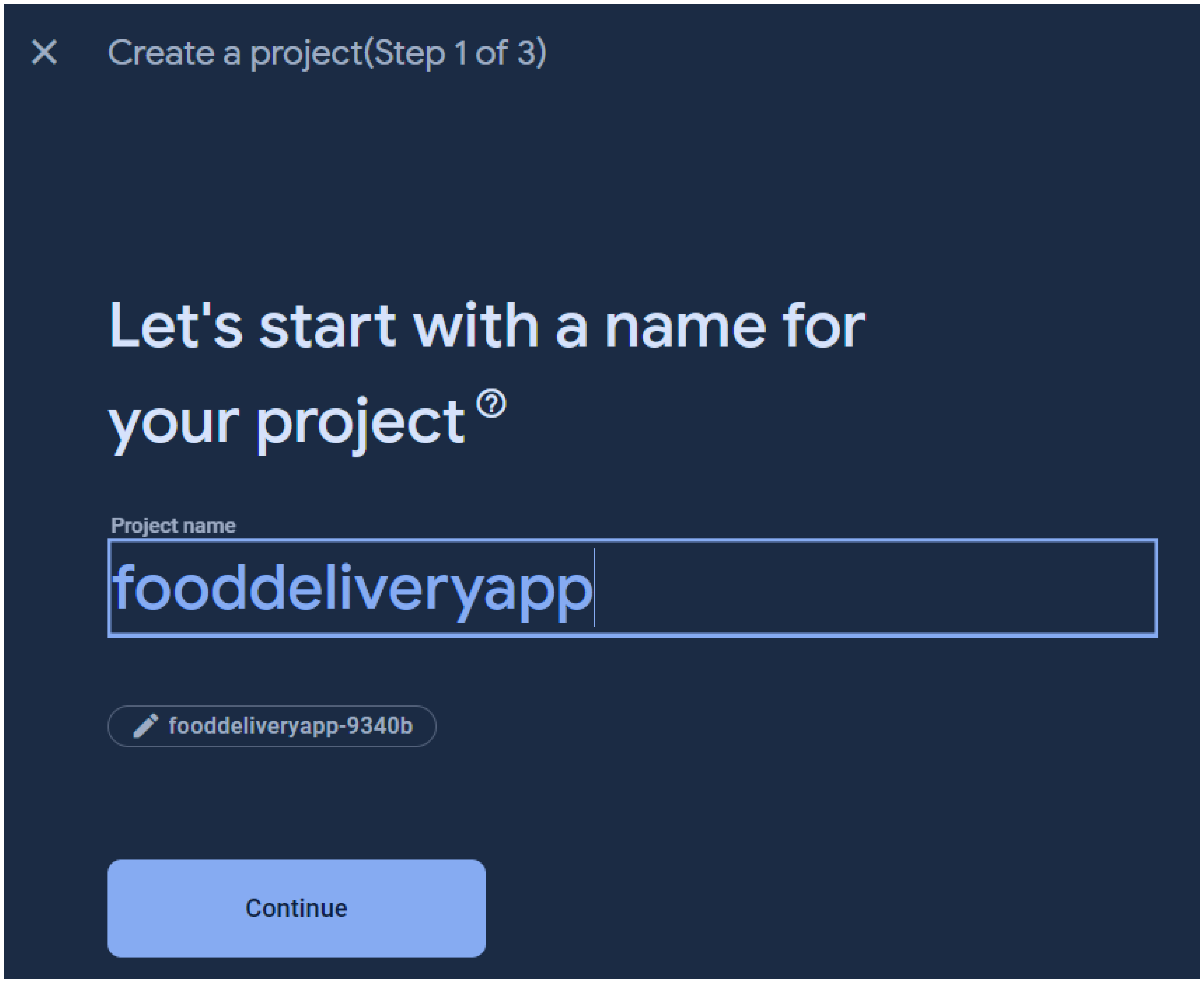
```
cd fooddeliveryapp
```

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 Firebase.

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:



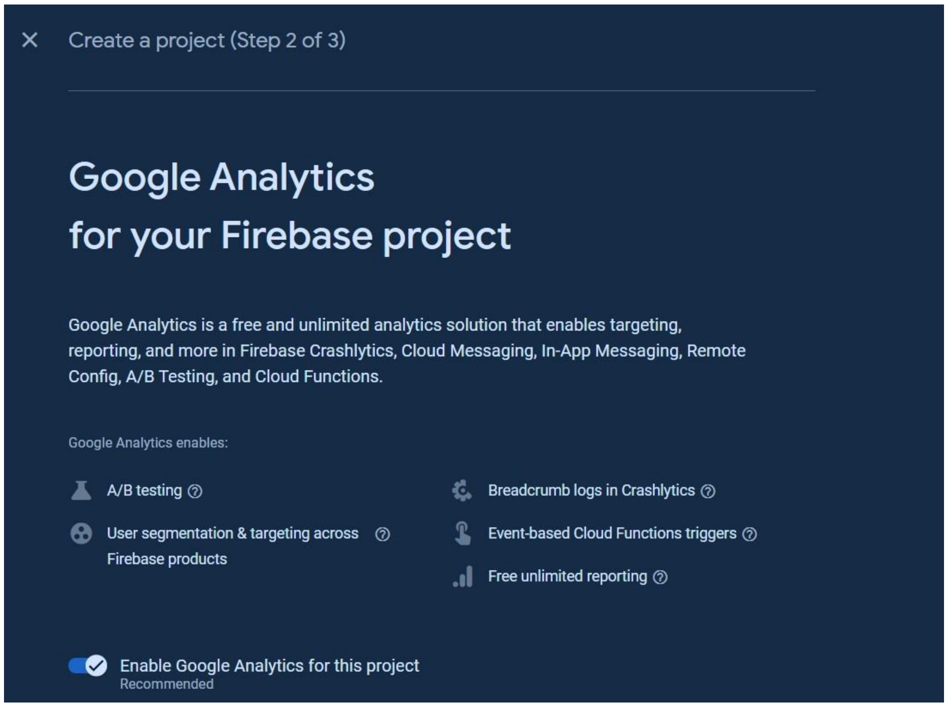
Create a project(Step 1 of 3)

Let's start with a name for your project

Project name: fooddeliveryapp

fooddeliveryapp-9340b

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 3)

Google Analytics
for your Firebase project

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.

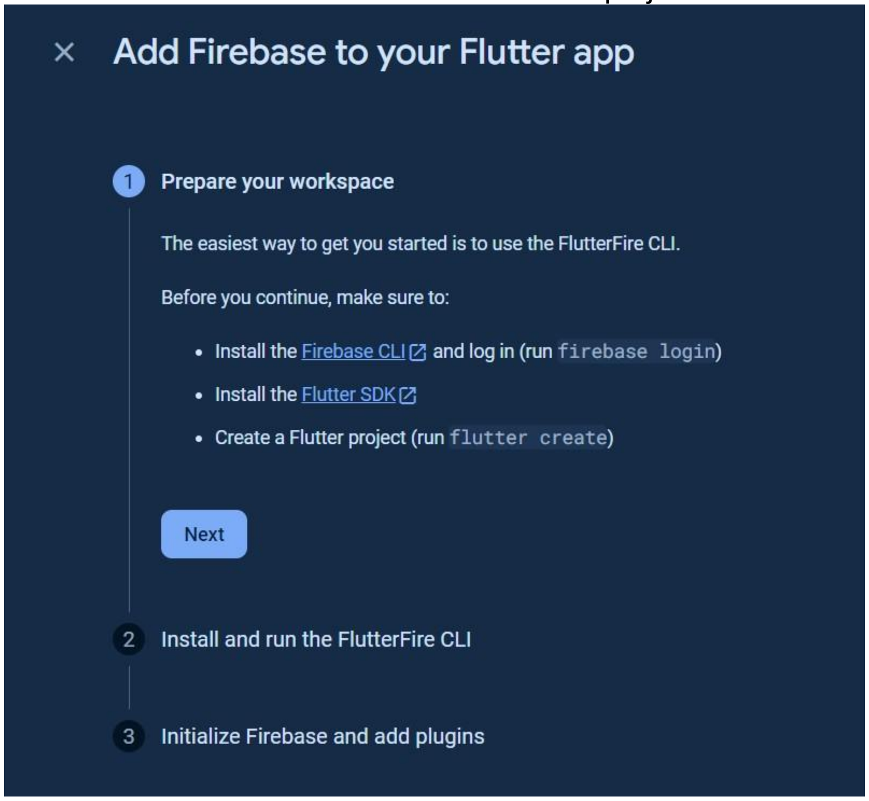
Google Analytics enables:

- 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

if you choose to use Google Analytics, you will need to review and accept the terms and conditions prior to project creation.

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.



Add Firebase to your Flutter app

1 Prepare your workspace

The easiest way to get you started is to use the FlutterFire CLI.

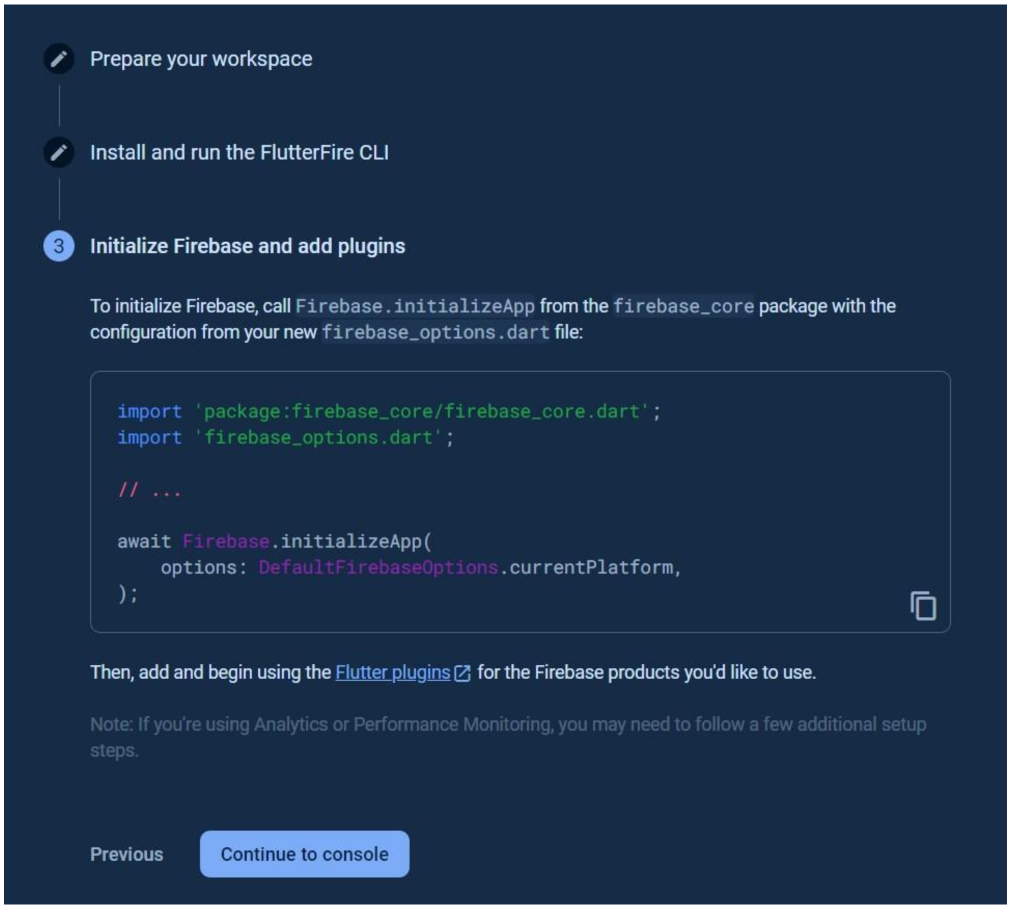
Before you continue, make sure to:

- Install the Firebase CLI and log in (run `firebase login`)
- Install the Flutter SDK
- Create a Flutter project (run `flutter create`)

Next

2 Install and run the FlutterFire CLI

3 Initialize Firebase and add plugins


Prepare your workspace
Install and run the FlutterFire CLI
3 Initialize Firebase and add plugins

To initialize Firebase, call `Firebase.initializeApp` from the `firebase_core` package with the configuration from your new `firebase_options.dart` file:

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

// ...

await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
);
```

Then, add and begin using the Flutter plugins for the Firebase products you'd like to use.

Note: If you're using Analytics or Performance Monitoring, you may need to follow a few additional setup steps.

Adding Android support:

Registering the App

The most important thing here is to match up the Android package name that you choose here with the one inside of our application.

The structure consists of at least two segments. A common pattern is to use a domain name, a company name, and the application name:

`com.example.fooddeliveryapp`

Once you've decided on a name, open `android/app/build.gradle` in your code editor and update the `applicationId` to match the Android package name:

android/app/build.gradle

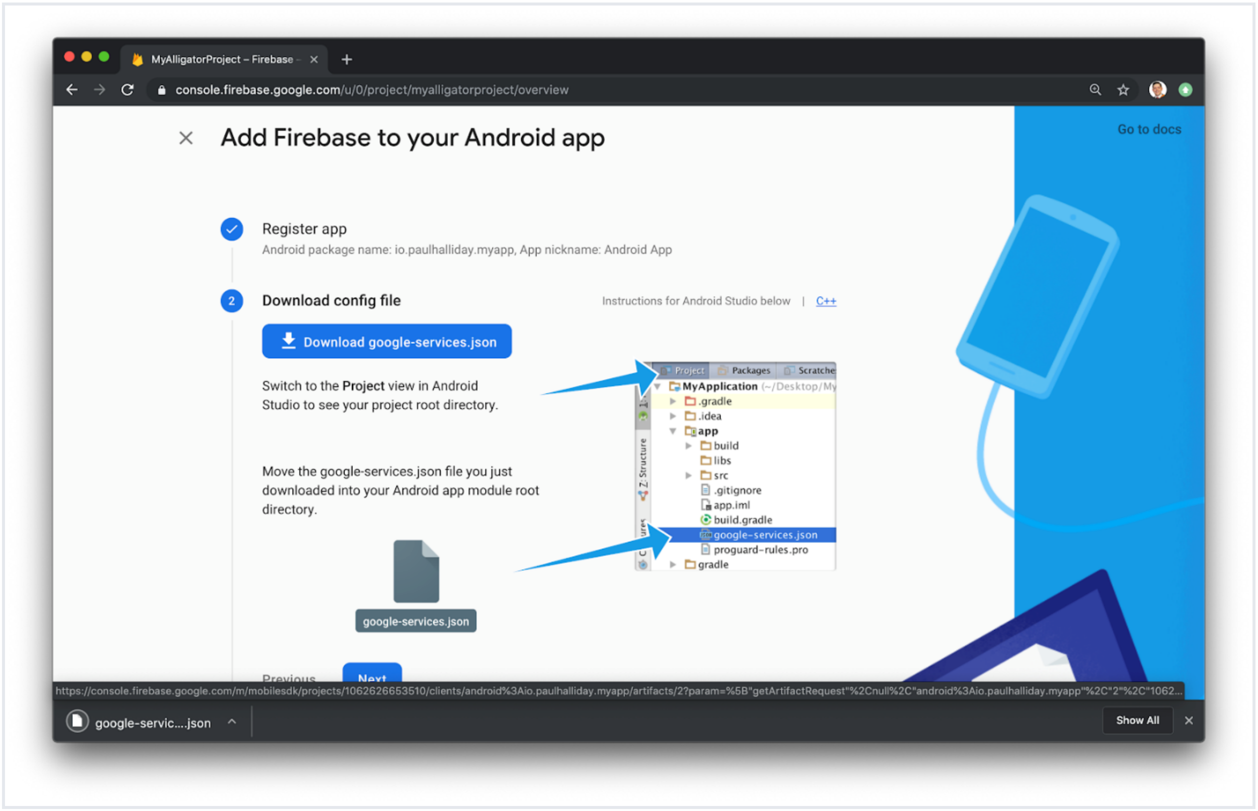
```
...
defaultConfig {
    // TODO: Specify your own unique Application ID
(https://developer.android.com/studio/build/application-id.html).
    applicationId 'com.example. fooddeliveryapp
'
    ...
}
...
```

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 Firebase to your Android app

Register app
Android package name: io.paulhalliday.myapp, App nickname: Android App

Download config file

Instructions for Android Studio below | C++

Download google-services.json

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

Move the google-services.json file you just downloaded into your Android app module root directory.

https://console.firebase.google.com/m/mobilesdk/projects/1062626653510/clients/android%3Aio.paulhalliday.myapp/artifacts/2?param=%5B"getArtifactRequest"%2Cnull%2C"android%3Aio.paulhalliday.myapp"%2C"2"%2C"1062...

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:4.3.6'
  }
}

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

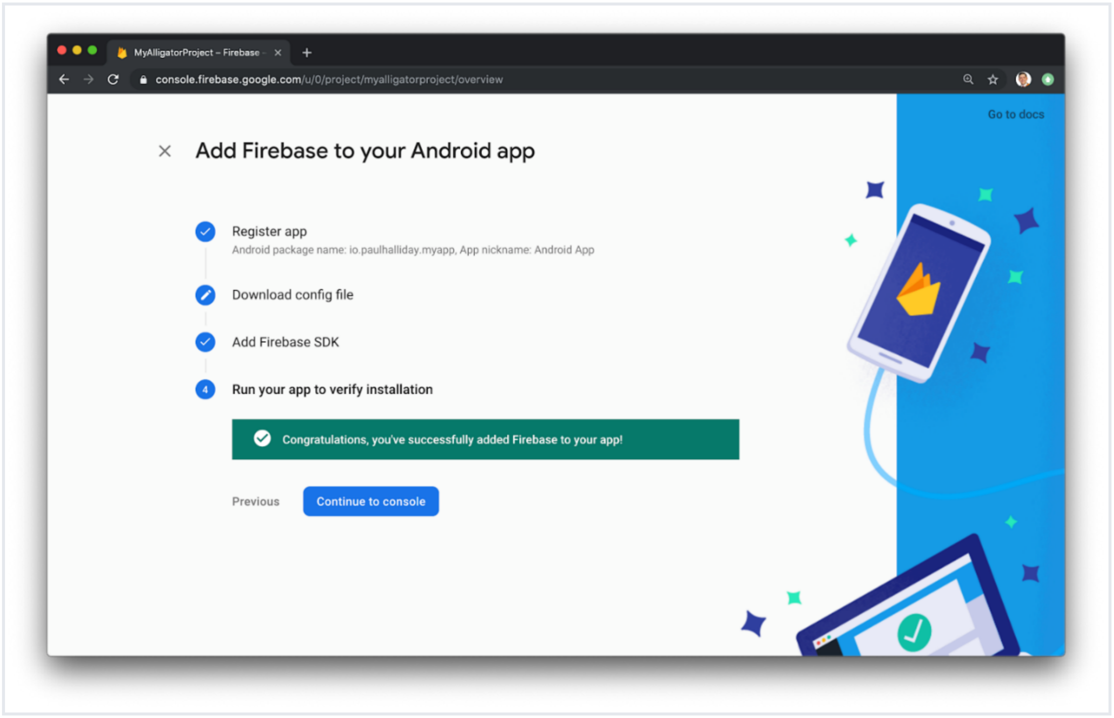
```
apply plugin: 'com.android.application'
// Add this line
apply plugin: 'com.google.gms.google-services'

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
}
```

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:


MyAlligatorProject – Firebase
console.firebase.google.com/u/0/project/myalligatorproject/overview
Go to docs

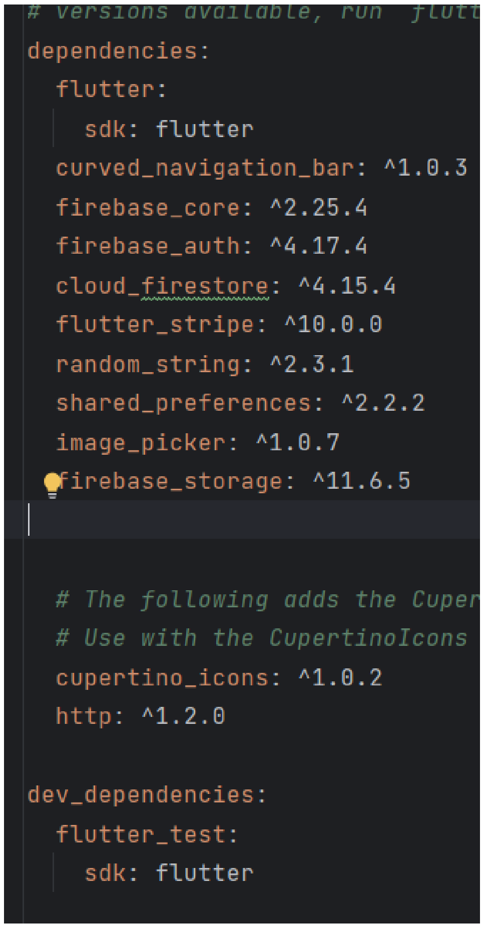
Add Firebase to your Android app

- Register app
  Android package name: io.paulhalliday.myapp, App nickname: Android App
- Download config file
- Add Firebase SDK
- Run your app to verify installation

Congratulations, you've successfully added Firebase to your app!

Previous
Continue to console

My Dependencies used in Food Delivery app using Firebase in Pubspec.yaml



```
# versions available, run  flutt
dependencies:
  flutter:
    sdk: flutter
  curved_navigation_bar: ^1.0.3
  firebase_core: ^2.25.4
  firebase_auth: ^4.17.4
  cloud_firestore: ^4.15.4
  flutter_stripe: ^10.0.0
  random_string: ^2.3.1
  shared_preferences: ^2.2.2
  image_picker: ^1.0.7
  firebase_storage: ^11.6.5


  # The following adds the Cuper
  # Use with the CupertinoIcons
  cupertino_icons: ^1.0.2
  http: ^1.2.0

dev_dependencies:
  flutter_test:
    sdk: flutter
```

Conclusion

In this experiment, we learned how to set up and ready our Flutter applications to be used with Firebase.

Flutter has official support for Firebase with the FlutterFire set of libraries.