EXPERIMENT NO:6

Aim: **-** How To 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 app, user authentication is handled using Firebase Authentication and user data is stored in MongoDB.

Login Functionality:

1. Users enter their email and password to log in.
2. Firebase Authentication verifies credentials.
3. If successful, the app retrieves the user's data from MongoDB and redirects them to the home page.

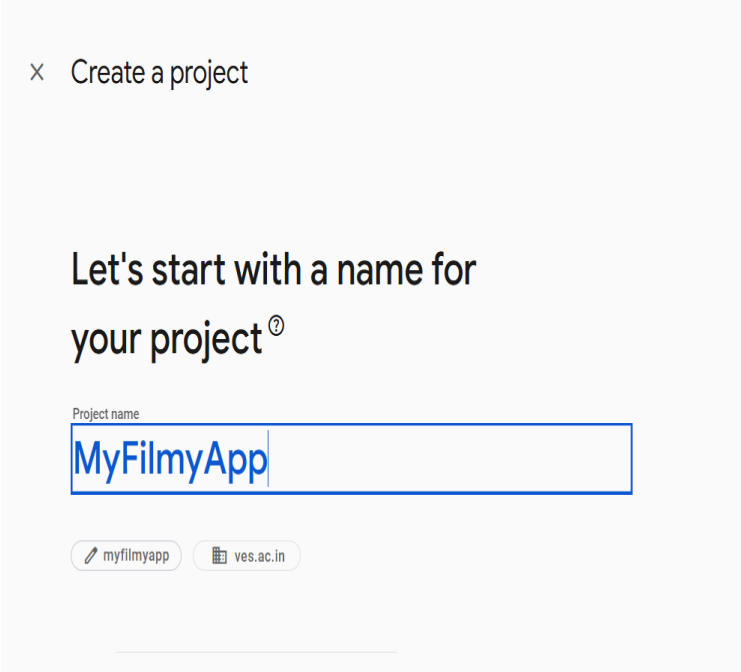
Signup Functionality:

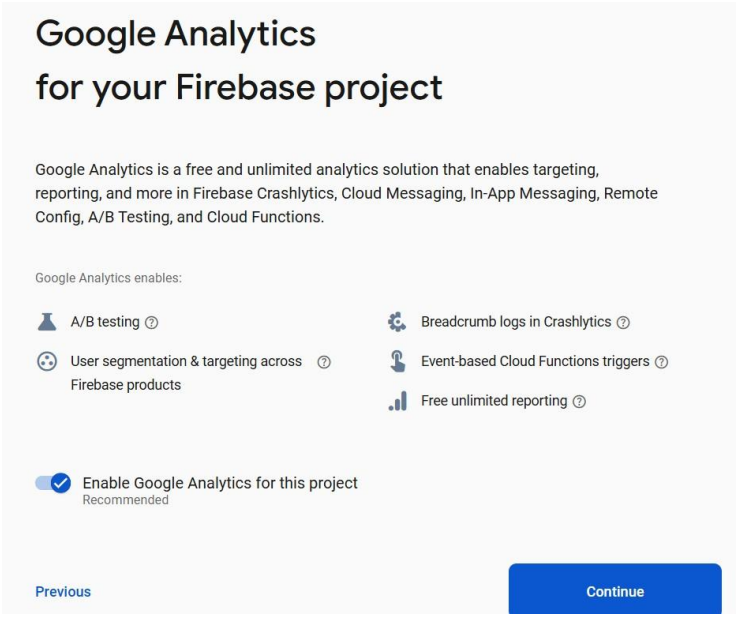
1. Users register with an email, password, and optionally other details.
2. Firebase Authentication creates the new user.
3. The user's details are then stored in MongoDB for future reference.
4. Once signed up, the user is redirected to the main page.

Steps:

1. Creating a New Firebase Project for WageTracker 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: WageTracker.
- 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.



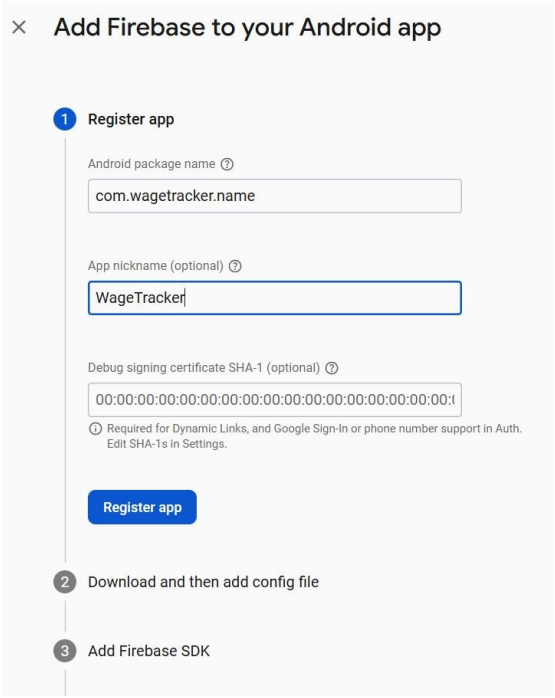


- If you choose to use Google Analytics, you will need to review and accept the terms and conditions prior to project creation.
- After pressing Continue, your project will be created and resources will be provisioned. You will then be directed to the dashboard for the new project.

1. Adding Android support

- **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:



- 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.flutterfirebaseexample**
- 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:

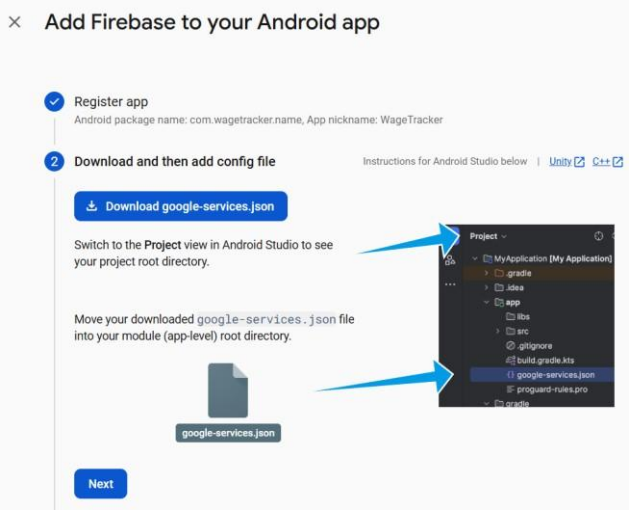
android/app/build.gradle

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

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:



- 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
    ...
  }
}
```

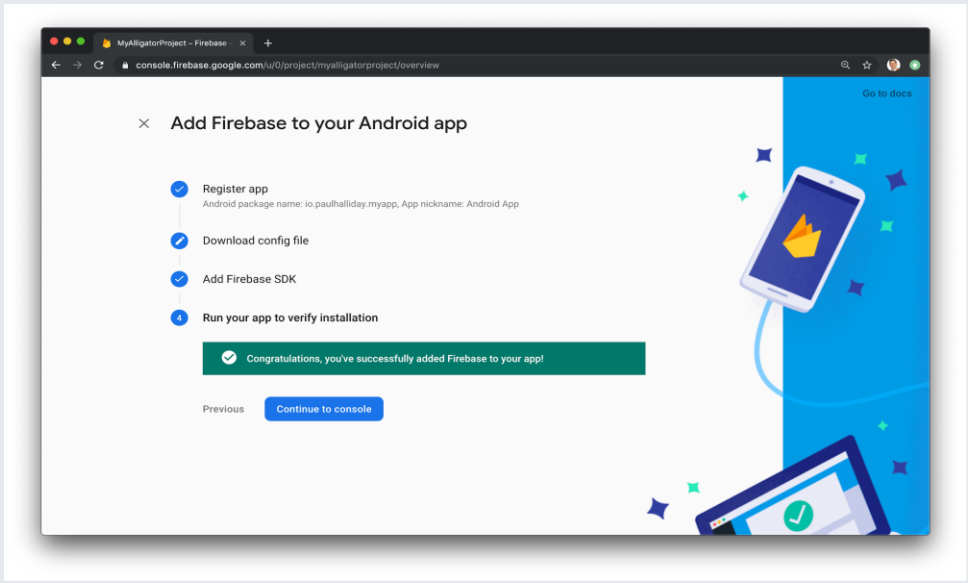
Finally, update the app level file at android/app/build.gradle to include the following:

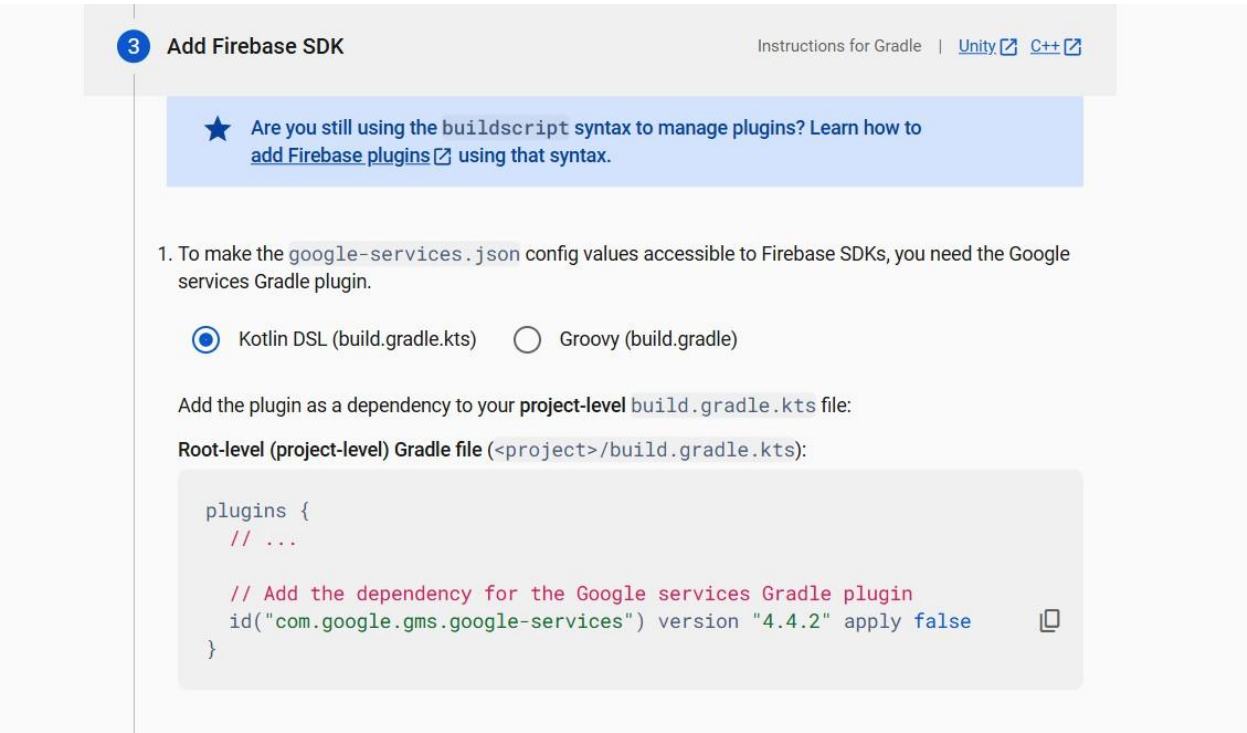
android/app/build.gradle

```
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:





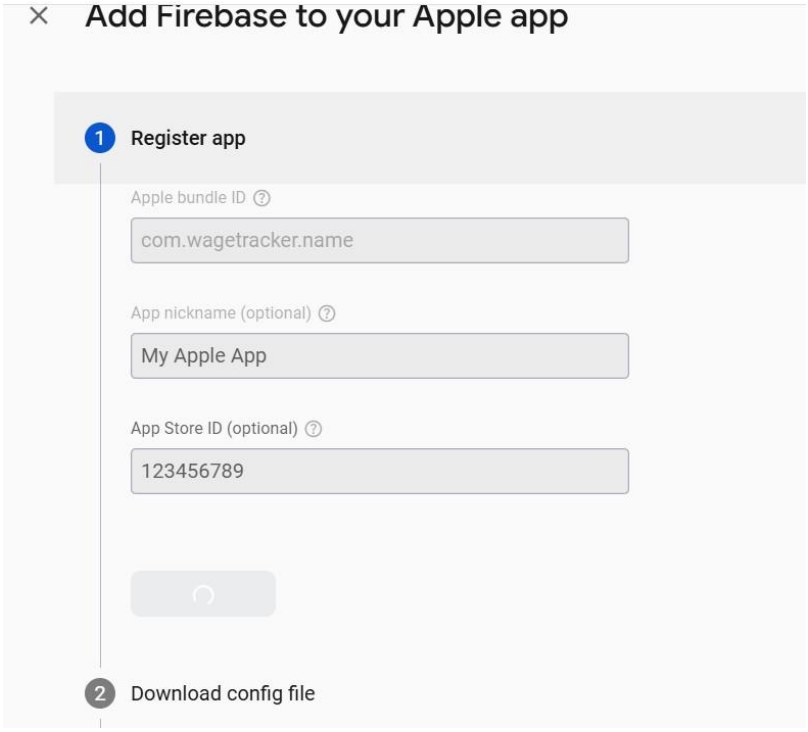
2. Adding iOS Support

In order to add Firebase support for iOS, we have to follow a similar set of instructions.

Head back over to the dashboard and select Add app and then iOS icon to be navigated to the setup process.

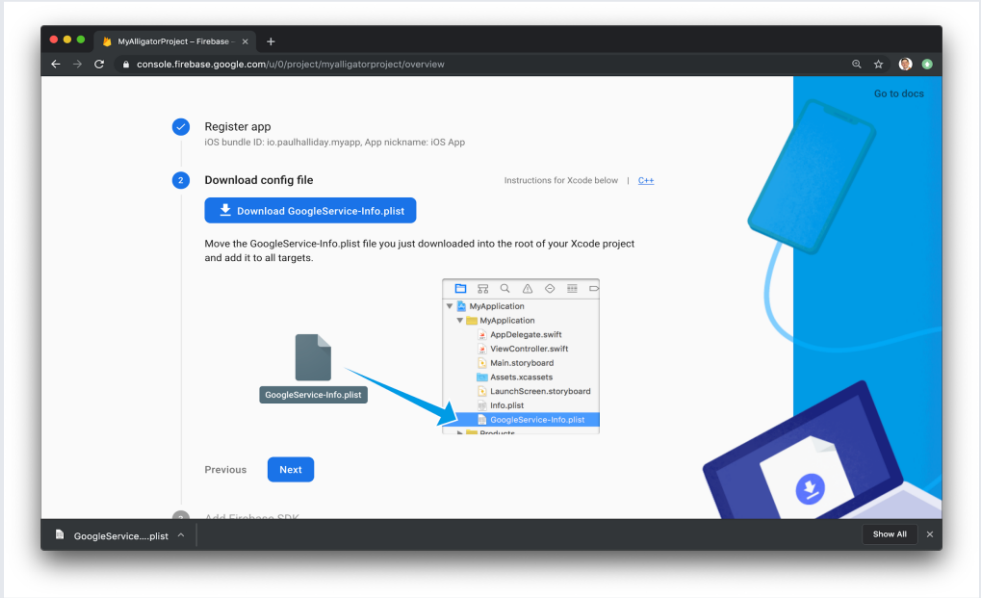
- **Registering an App**

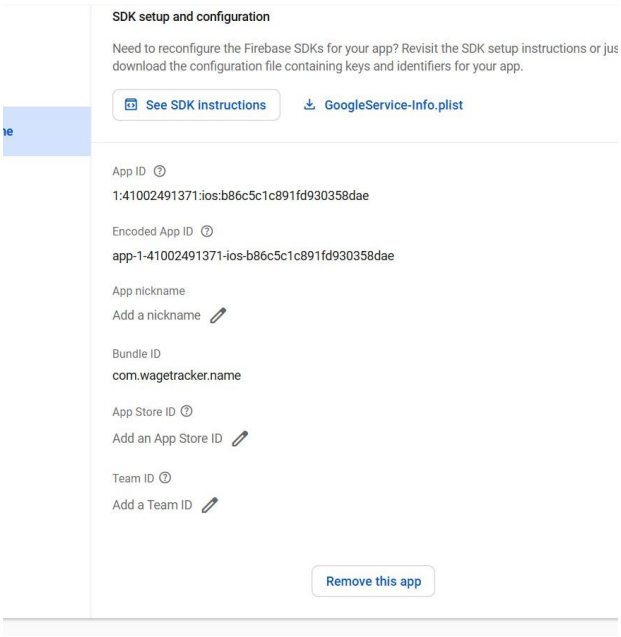
  Once again, we'll need to add an "iOS Bundle ID". It is possible to use the "Android package name" for consistency:



Click Register app to move to the next screen.

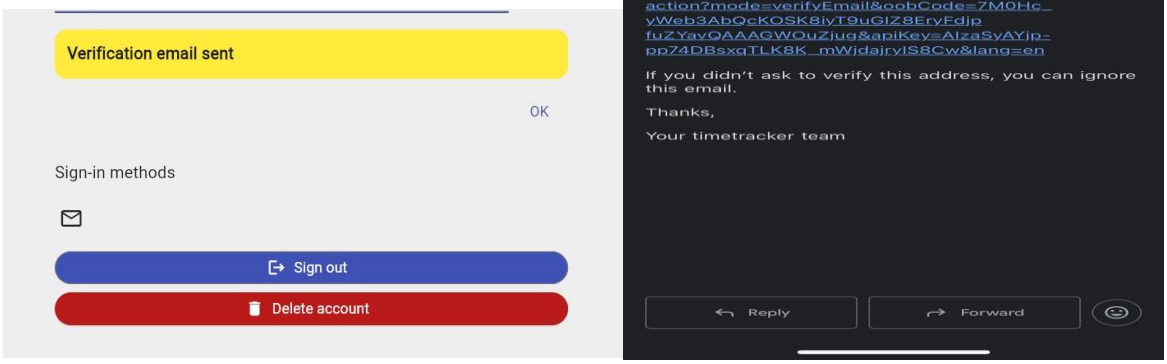
Downloading the Config File





Download GoogleService-Info.plist and move this into the root of your Xcode project within Runner.

Github Link: https://github.com/pleasecodekomal/wagetracker Output:



Conclusion: This experiment successfully demonstrated how to connect a Flutter UI to a

Firebase database for real-time data management. By integrating Firebase Authentication and Firestore, user authentication and data storage were implemented efficiently. The experiment also covered CRUD operations, ensuring smooth interaction with the database. Proper authentication and security measures were applied to protect user data. Overall, this experiment highlights the seamless integration of Flutter with Firebase, enabling the development of dynamic and secure applications.