ParkEasy Documentation

Overview

- The ParkEasy admin panel is allows you can manage registered users, settings and many more, you can see live user data

• Prerequisites

- Before setting up and running the ParkEasy app, make sure you have the following installed on your system: Flutter SDK: Follow the official Flutter installation guide for your operating system: https://flutter.dev/docs/get-started/install
- Code editor: Choose a code editor Android Studio.

• Firestore Database Collection Import Export

1) Setup NPM in your Computer URL: https://nodejs.org/en/download/

2) Extract Source Code of firestore-import-export-master.zip

3) We have to open command prompt on project folder and run command: npm install

4) Setup Firebase Project if you not created.

5) Configure appConfig.json file. you can get from firebase Project settings

- ✞ Go to > Service account > Select Node.js > Generate new private key
- ✞ Wait untill create key this will auto download. and replace with current appConfig.json

6) Run command on project folder bellow import commands

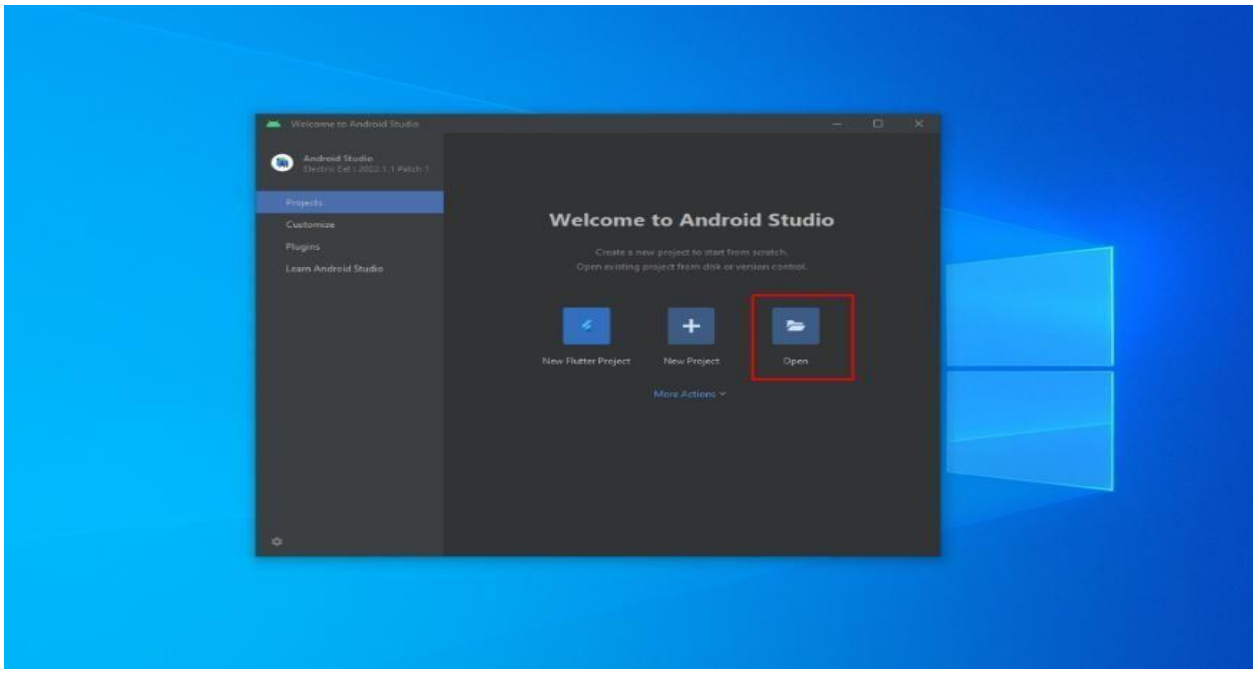
 # IMPORT Commands:

```
npx -p node-firestore-import-export firestore-import -a appConfig.json -b backup.json
```

 # Export commands:

```
npx -p node-firestore-import-export firestore-export -a appConfig.json -b backup.json
```

- ## Project Setup
  - Follow these steps to set up and run the **ParkEasy**
    1. Extract this Zip file.
    2. Open the Android Studio.
    3. Open extract file in the Android Studio.



- Open terminal in Android Studio and run command: **flutter pub get**

• Connect Project to Firebase

- Follow this video: https://youtu.be/fxDusoMcWj8
- For Firebase deployment use this article: https://medium.com/flutter/must-tryusehttps://medium.com/flutter/must-try-use-firebase-to-host- your-flutter-appon-the-web-852ee533a469firebase-to-host- your-flutter-app-on-the-web852ee533a469
- Follow this step for create project in firebase Step 1: Create the Firebase project

  Step 2: Install the firebase-tools CLI

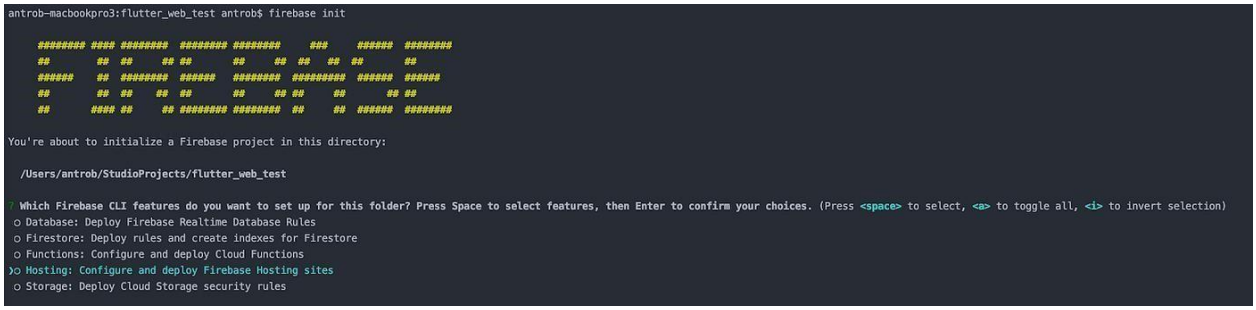
  Step 3: Initialize Firebase hosting for your Flutter project

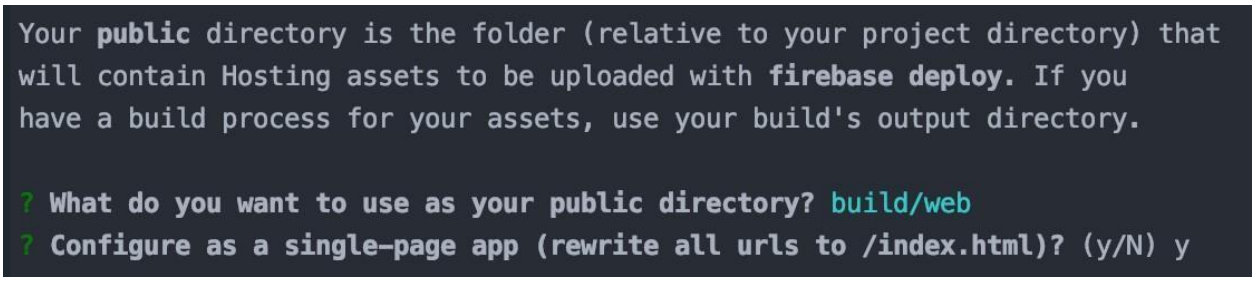
- After initializing follow these steps: -

1. Open a terminal and navigate to the root directory for your Flutter app and run this command to login: firebase login

   → If you are already logged in and want to re-login as different account, first logout with: firebase logout

2. Next, run this command from the root directory of your Flutter app to initialize your Firebase project: firebase init
3. After firebase init follow this picture





4. For Build your app run this command:

```
flutter build web --web-renderer html -release
```

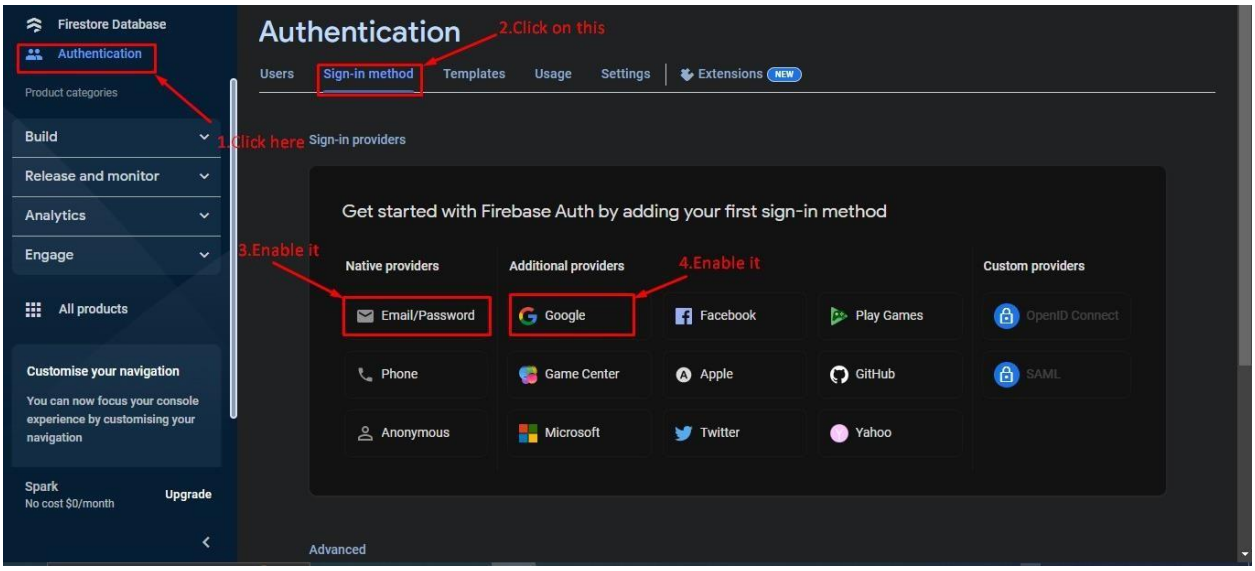
5. After build deploy for deploy your app run this command:

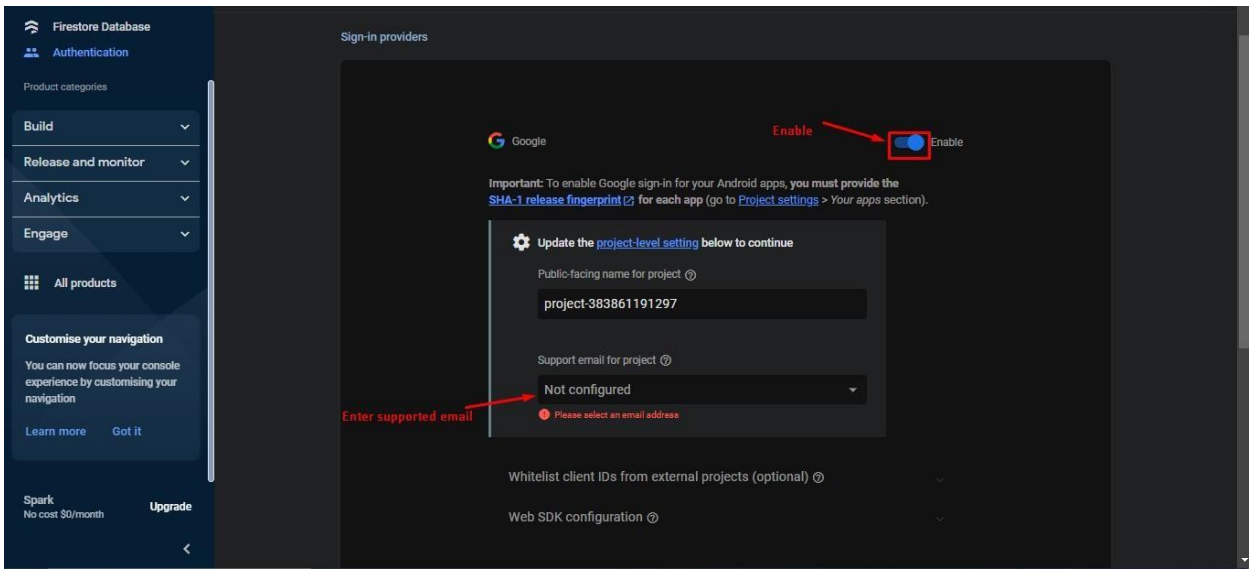
```
firebase deploy
```

6. After, follow this all-step check firebase console and then go to hosting

• Set up for Firebase Authentication

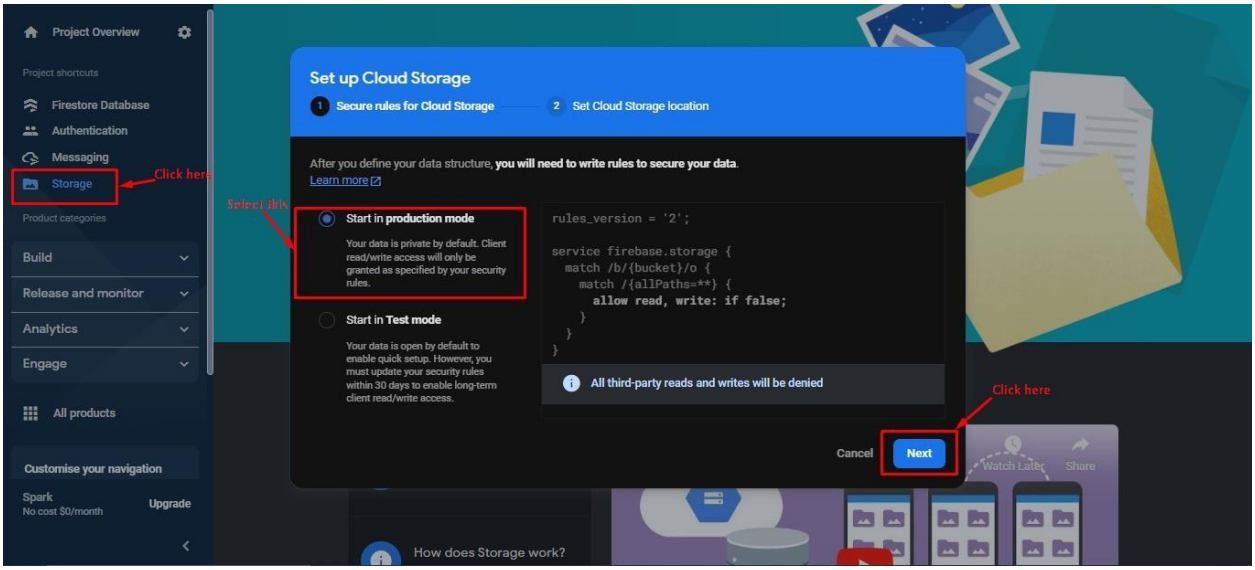
1. After hosting click on Authentication then click on get started button
2. After click on start button follow this step:

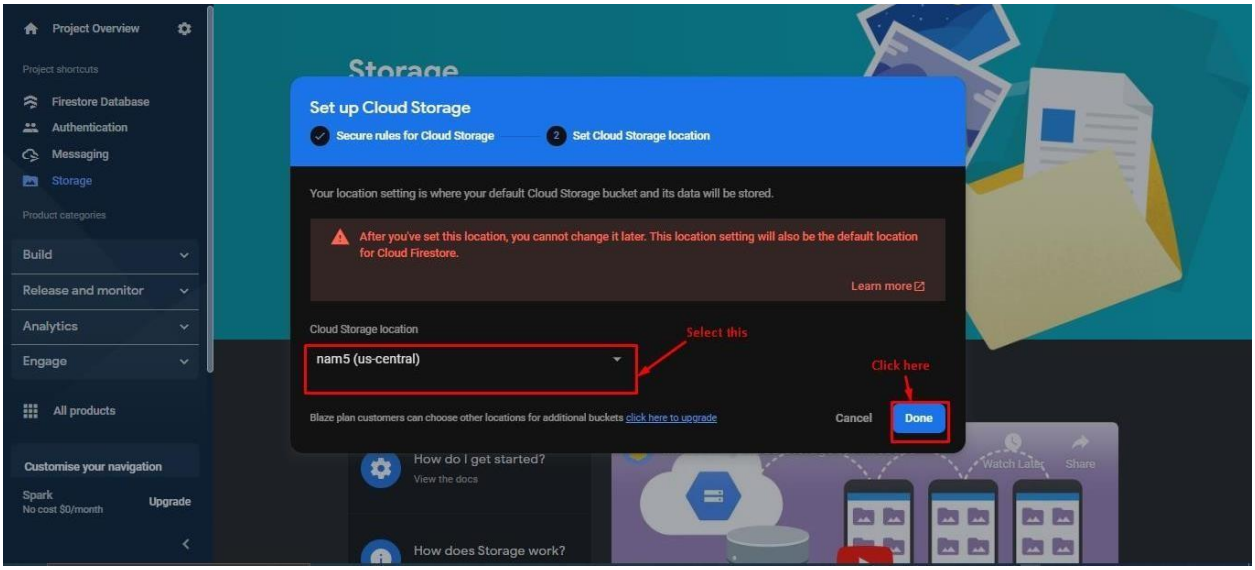


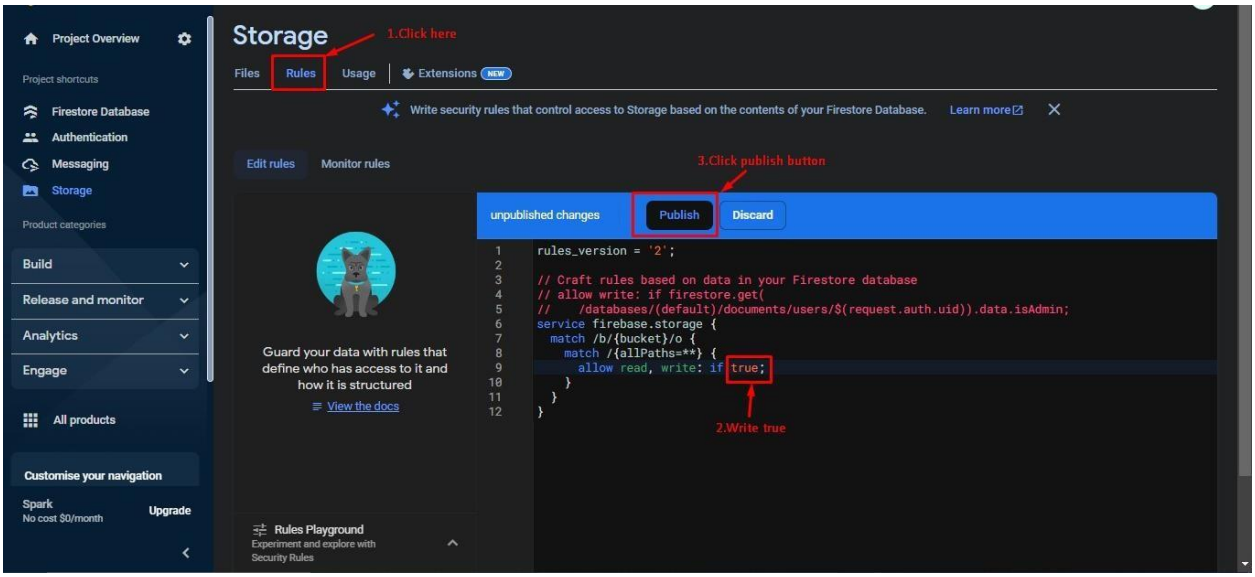


Excellent Your Firebase Authentication setup is complete

- **Firebase Storage**
  - For use firebase storage follow this step:

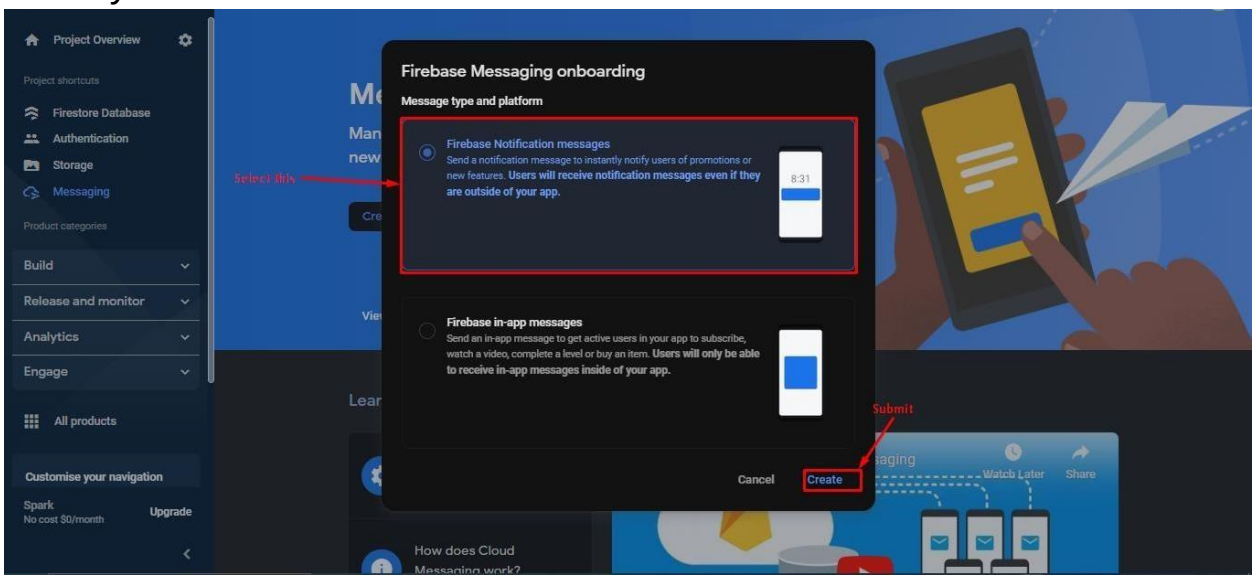




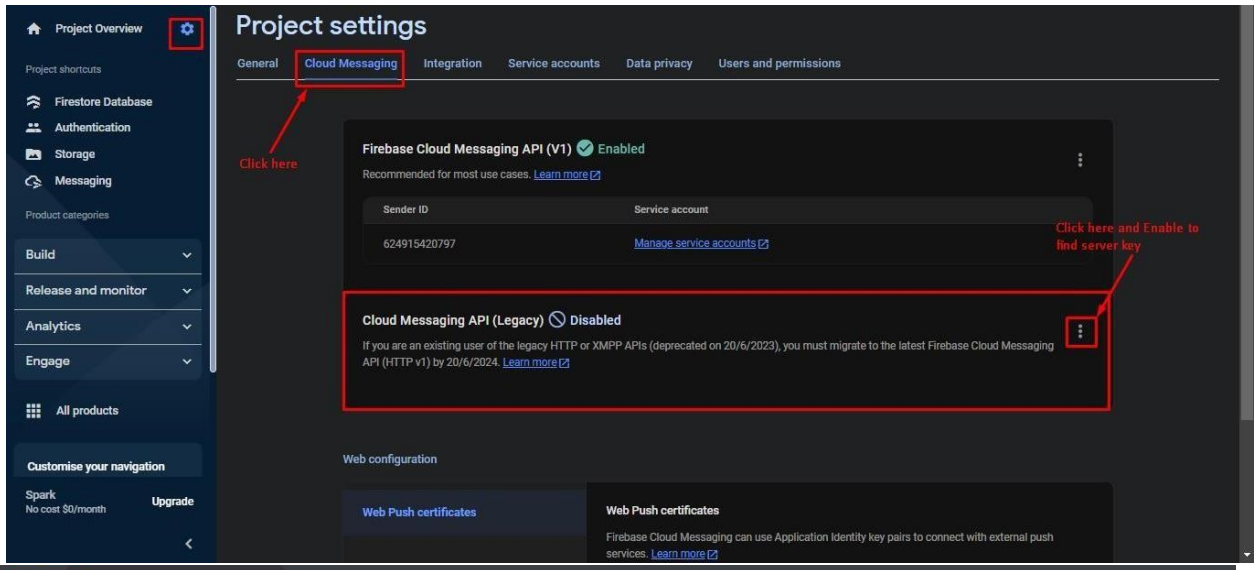


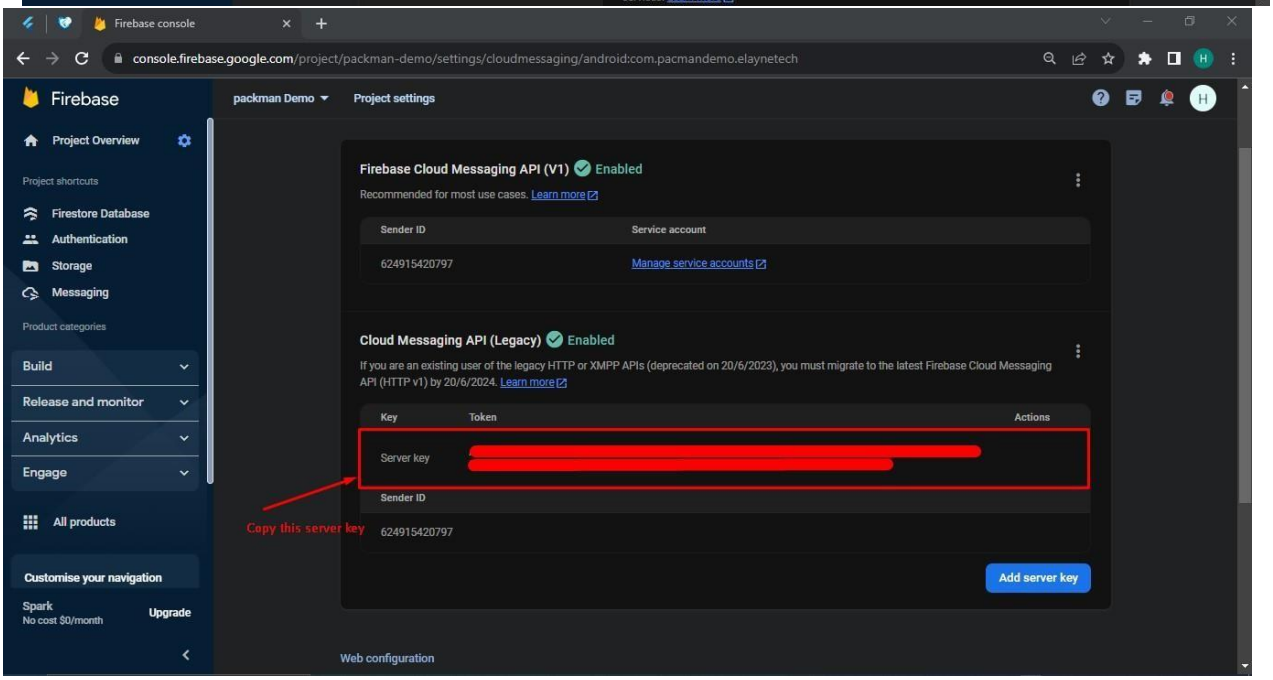
- Firebase Notification

- For enable notification to follow this step and get a notification server key



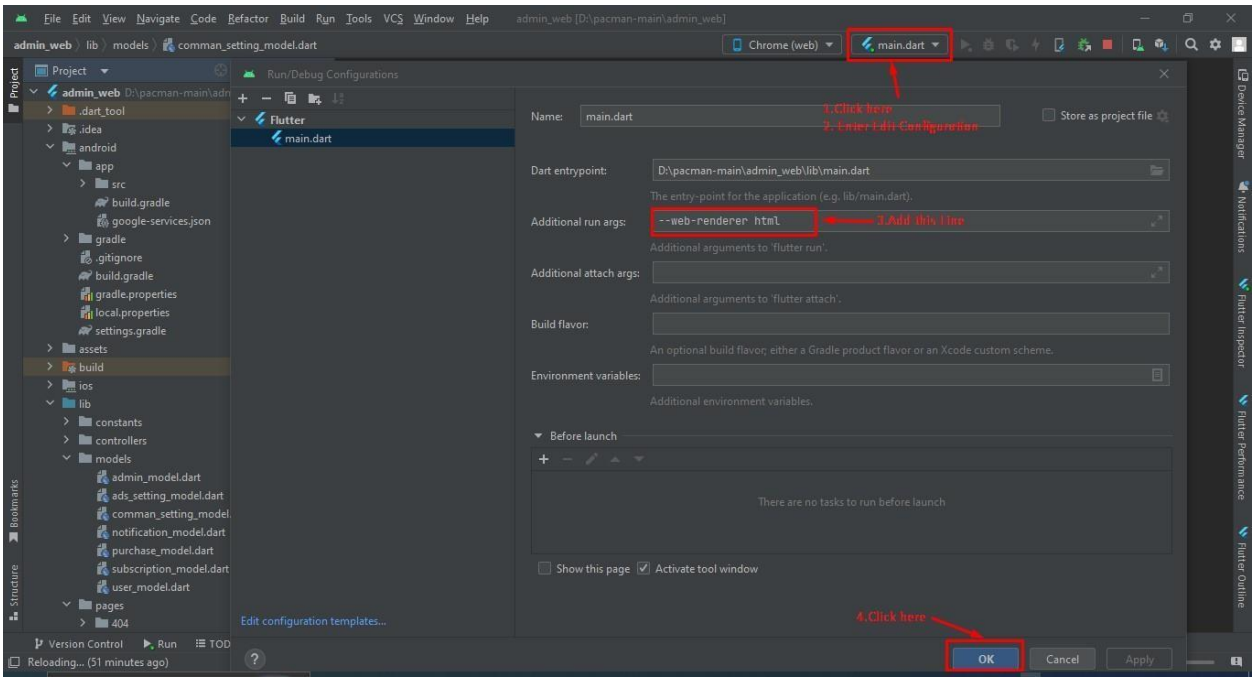
- Now you can go Project Setting Section and Select cloud messaging





- After getting a server key then place in admin panel general setting

✞ Before Run admin penal First Add this –wen-renderer html command in main Configuration.



- **Change app name in Android and IOS in flutter**

For change app name in android follow this step:

- Open the **android/app/src/main/AndroidManifest.xml** file.
- Look for the **<application>** tag and find the **android: label** attribute. This attribute specifies the app name

```
<application
    android:label="Your App Name">
```

For change app name in IOS follow this step:

- Open the **ios/Runner/Info.plist** file.
- Find the **CFBundleName** key and change its value to the desired app name. This key is typically located in a **<dict>** section under **<key>CFBundleName</key>**

```
<key>CFBundleName</key>
<string>Your App Name</string>
```

- Update the string value with the desired app name.

- After making these changes, the app name should be updated for both Android and iOS. Keep in mind that you might need to rebuild and reinstall the app on your devices or emulators for the changes to take effect.
- Additionally, make sure to follow any naming conventions or guidelines specific to each platform, especially if you plan to publish your app-onapp stores.

Change app launcher logo in Android and IOS in flutter

- **In Android, you need to replace the launcher icon in the res directory. Follow these steps:**
  - Create a set of icons in different sizes for various screen densities (mdpi, hdpi, xhdpi, xxhdpi, xxxhdpi). You can use tools like Android Asset Studio (https://romannurik.github.io/AndroidAssetStudio/) to generate these icons. ○ Replace the existing icons in your Flutter project's **android/app/src/main/res** directory with your newly created icons.
    Make sure to replace the icons in all the density-specific folders.
  - After replacing the icons, rebuild your Flutter app using **flutter build** or **flutter run**. The new launcher icon should appear on the Android home screen.

- **In iOS, you need to replace the app icon images in the Assets.xcassets folder. Here's how:**
  - Create a set of app icon images in different sizes for various iOS devices. These icons should be in square dimensions (e.g., 60x60, 120x120, 180x180, etc.). ○ Open the Xcode project for your Flutter app, which is located in the **ios/Runner.xcodeproj** directory. ○ In Xcode, find the **Runner** project in the left sidebar and expand it. You will see an asset catalog named **Assets.xcassets**.
  - Inside the **Assets.xcassets**, locate the "AppIcon" asset set. Delete the existing icons and add your new icons to their respective slots by dragging and dropping. ○ After updating the icons in Xcode, rebuild and run your Flutter app.
    The new app icon should appear on the iOS device's home screen.
- Remember to follow the platform-specific guidelines for app icon sizes and naming conventions. Also, keep in mind that the app icon may have different sizes and shapes for different iOS devices and versions, so you should provide a complete set of icons to ensure your app looks good on all devices.

Change app id in Android and IOS in flutter

- To change the application ID for Android, you need to update the **android/app/build.gradle** file in your Flutter project:
  - Open the **android/app/build.gradle** file. ○ Locate the **defaultConfig** block and change the **applicationId** to your desired new package name. For example:

```
android {
    defaultConfig {
        applicationId "com.example.newpackage"
        // ...
    }
}
```

  - Save the changes. ○ You may need to update the **MainActivity** and other references throughout your Android codebase with the new package name.
  - Rebuild your Flutter app for Android by running **flutter build** or **flutter run**. The app will now use the new package name.
  -
- **To change the bundle identifier (application ID) for iOS, you need to update the Xcode project settings:**
  - Open your Flutter project's **ios/Runner.xcodeproj** using Xcode. ○ In Xcode, select the "Runner" target. ○ In the "General" tab, locate the "Bundle Identifier" field and change it to your desired new package name (e.g., **com.example.newpackage**).
  - Save the changes. ○ You may also need to update any other references to the old bundle identifier in your iOS code. ○ Rebuild your Flutter app for iOS by running **flutter build** or **flutter run**. The app will now use the new bundle identifier.
- Remember to keep your new package name or bundle identifier consistent across both platforms and update any external services or

configurations that may depend on the application ID or bundle identifier. Also, make sure to follow any naming guidelines or requirements set by the respective app stores when publishing your app.

Change google map key in Android and IOS in flutter

- **For Android:**
  - **Update AndroidManifest.xml**: In Android, you'll need to change the API key in your **AndroidManifest.xml** file. Follow these steps: ○ Open the **android/app/src/main/AndroidManifest.xml** file.
  - Locate the **<meta-data>** element within the **<application>** section where you specify the API key. It will look like this:

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="YOUR_API_KEY_HERE"/>
```

  - Replace **"YOUR_API_KEY_HERE"** with your new Google Maps API key. ○ **Rebuild the App**: After making this change, rebuild your Flutter app for Android using **flutter build** or **flutter run**. Your app should now use the updated API key.

- **For iOS:**
  - **Update Info.plist**: In iOS, you need to change the API key in the **Info.plist** file. Follow these steps: ○ Open the **ios/Runner/Info.plist** file.
  - Locate the key-value pair under **<key>NSLocationWhenInUseUsageDescription</key>**. This key is used for the Google Maps API key and should look like this:

```
<key>NSLocationWhenInUseUsageDescription</key>
<string>Your API key goes here</string>
```

-
    - Replace **"Your API key goes here"** with your new Google Maps API key. ○ **Rebuild the App**: After making this change, rebuild your Flutter app for iOS using **flutter build** or **flutter run**. Your app should now use the updated API key.
- Remember to ensure that your new API key has the necessary permissions and restrictions set up in your Google Cloud Console

project, such as allowing usage from Android and iOS apps and specifying the correct package names and bundle identifiers. Also, consider securing your API key to prevent unauthorized use.

• Firebase setup in project

1. **Create a Firebase Project:** ○ Go to the Firebase Console. ○ Click on "Add project" and follow the setup wizard to create a new project.
2. **Register Your App with Firebase:** ○ In the Firebase Console, select your project. ○ Click on the "Add app" button and select the appropriate platform (iOS or Android). ○ Follow the on-screen instructions to register your app. This will involve providing a package name for your Android app and a bundle identifier for your iOS app.
3. **Download Configuration Files:** ○ After registering your app, you'll be prompted to download configuration files for both Android and iOS. These files contain the necessary settings to connect your app to Firebase.
4. **Add Configuration Files to Your Project:** ○ For Android, place the **google-services.json** file in the **android/app** directory of your Flutter project. ○ For iOS, place the **GoogleService-Info.plist** file in the Runner folder of your iOS project.
5. **Add Dependencies to Your Flutter Project:** ○ Open your **pubspec.yaml** file and add the Firebase plugins you want to use. For example, to use Firebase Authentication and Firestore, add these dependencies:

```
dependencies:
  firebase_core: ^latest_version
  firebase_auth: ^latest_version
  cloud_firestore: ^latest_version
```

   - Replace **latest_version** with the actual version numbers of the packages.
6. **Initialize Firebase in Your App:** ○ In your Flutter app, you should initialize Firebase in the **main.dart** file or wherever your app is initialized. Import the necessary packages and call **Firebase.initializeApp()**:

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

  void main() async {
    WidgetsFlutterBinding.ensureInitialized();
    await Firebase.initializeApp();
    runApp(MyApp());
  }
  ```

7. **Use Firebase Services:** ○ Now you can use Firebase services like Authentication, Firestore, Realtime Database, and more in your Flutter app. Import the appropriate Firebase packages and follow the documentation for each service to set it up and use it.
8. **Testing and Verification:** ○ Test your app to ensure Firebase services are working correctly. You can use the Firebase Console to monitor and manage your Firebase project.

That's it! You've successfully set up Firebase in your Flutter project. Remember to refer to the official documentation for each Firebase service you intend to use for more detailed setup and usage instructions.
  ```