



Introduction To POS Saas- Flutter POS Inventory Software App + Web + Super Admin

Pos Saas - POS SaaS is a cutting-edge Point of Sale (POS) solution delivered as a Software as a Service (SaaS) application. Developed using the Flutter framework for a seamless cross-platform experience, it integrates seamlessly with Firebase to provide a robust and scalable backend. Utilizes Flutter for a responsive and consistent user experience across iOS and Android devices. Real-time data synchronization ensures that your data is always up-to-date. Access real-time analytics and reports for a holistic view of business performance. Multiple admin roles with customizable permissions for each user. Efficiently manage users, roles, and access levels for optimal security. Easily add, edit, and organize products with a user-friendly interface. Categorize products for efficient inventory management. Real-time tracking of stock levels to prevent overstock or stockouts. Automated alerts for low inventory.

Getting Started

This guide will get you started with the setup of the application on your local machine. Now that you have set up your server, it's time to set up the mobile application to consume the data from your website. This guide will help you to connect the application with your server.

Please make sure that you follow the steps mentioned in this section very carefully.
If you miss some steps it may lead to unwanted behavior or errors.

Prerequisite:

- **Android Studio: 2024.1.1**
- **Flutter version: 3.22.3 (min)**

Install Flutter

Before you can start testing your application, you will need to install Flutter on your machine.

Please install Flutter version +3 to avoid any dependency issues.

Setting up an Android Emulator or an iOS Simulator which will run your application on your local machine is part of the flutter installation.

After you've installed flutter then please follow the steps from the following link to setup your dev machine: <https://flutter.dev/docs/get-started/install>

Please make sure that you follow each and every step that is mentioned in the flutter installation guide properly

Set Up an editor

The recommended editor is Android Studio. Follow the guide from the link to setup an editor: <https://flutter.dev/docs/get-started/editor>

Flutter Doctor

After you've followed through all the steps from the Flutter installation guide above, you need to check your installation for any issues that might cause some problem. You can do this by following the command below:

```
flutter doctor
```

```
Last login: Sat Aug 31 11:39:07 on console
[tahmidtarongo@Tahmids-MacBook-Air ~ % flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.24.1, on macOS 14.6.1 23G93 darwin-arm64, locale en-BD)
[✓] Android toolchain - develop for Android devices (Android SDK version 35.0.0)
[✓] Xcode - develop for iOS and macOS (Xcode 15.4)
[✓] Chrome - develop for the web
[✓] Android Studio (version 2024.1)
[✓] Connected device (5 available)
[✓] Network resources

• No issues found!
tahmidtarongo@Tahmids-MacBook-Air ~ %
```

Note that the summary above has all tick marks. If you see a cross (X) in-front of an option then follow the steps mentioned below that option to complete your installation. If you see [!] No Connected device (0 available) do not worry, you will open an iOS Simulator or an Android Emulator later in the setup process which will resolve this.

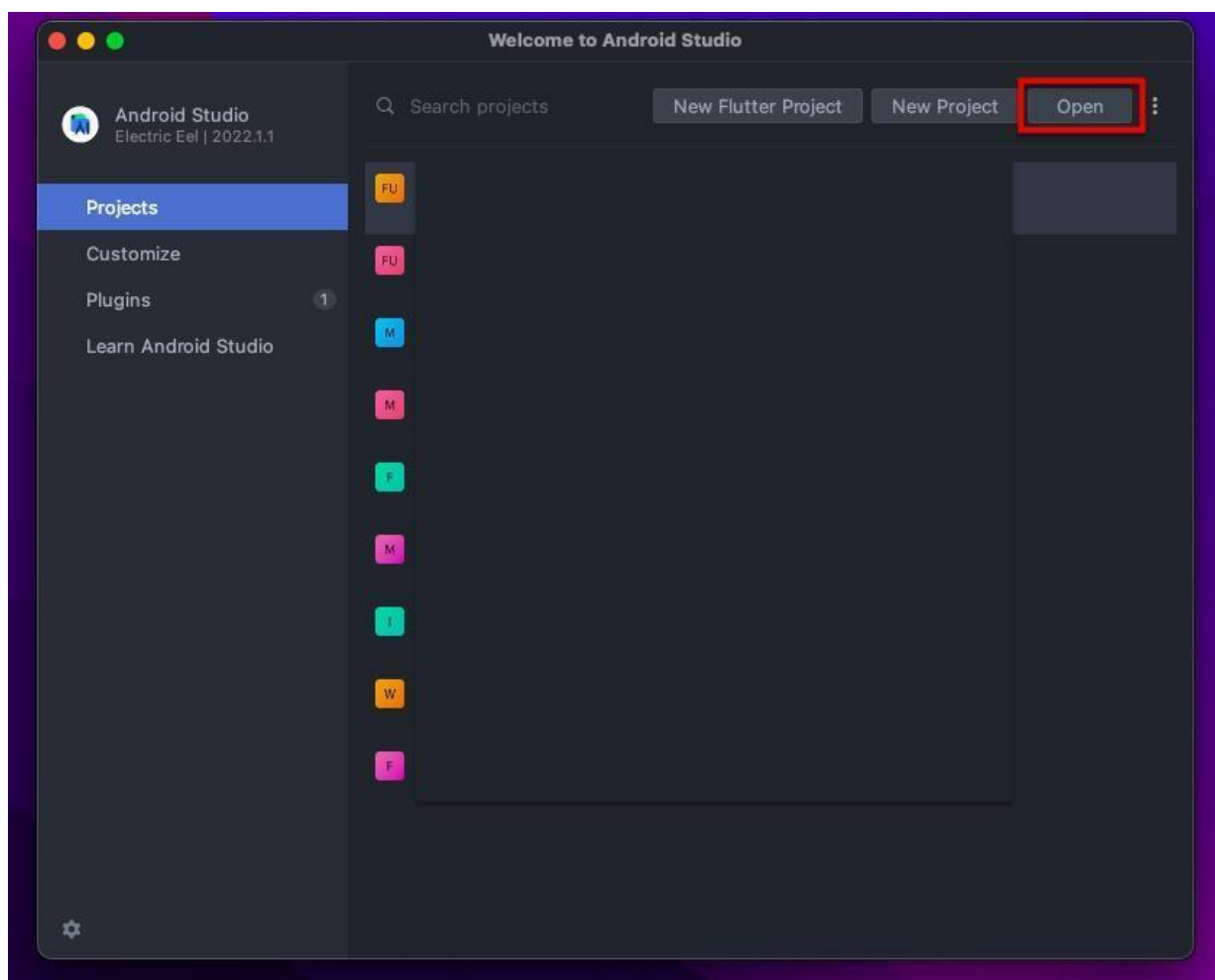
Modify App

Once you installed Android Studio on your machine, you can start changing the required files for App from android studio by following this guide.

Open app in Android Studio

To modify the required settings for your POS SaaS applications, you need to open it in Android Studio. Follow the steps below to open the application in android studio.

1. Open Android Studio on your machine.
2. From the Welcome to Android Studio screen, click on Open.



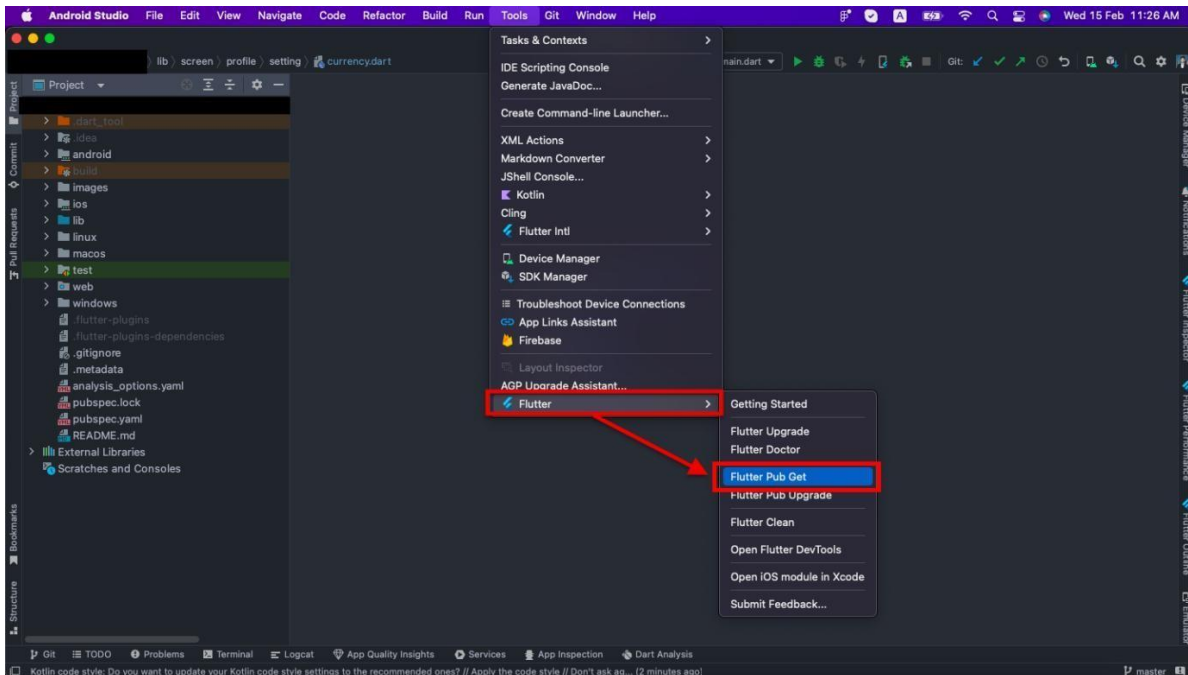
3. Choose the Pos SaaS - App / Pos SaaS Web/ Pos SaaS super admin directory and open it. Note that you have to set the web first. It is located at download or the folder you selected when it was downloaded from the marketplace. This will open the application source code in android studio.

Now you are ready to make the necessary changes in the application files by following the guide below.

Get Flutter Packages

Now you have to open tool option from the topbar and from the opening menu you have to choose Flutter and then Flutter Pub Get & press or select that.

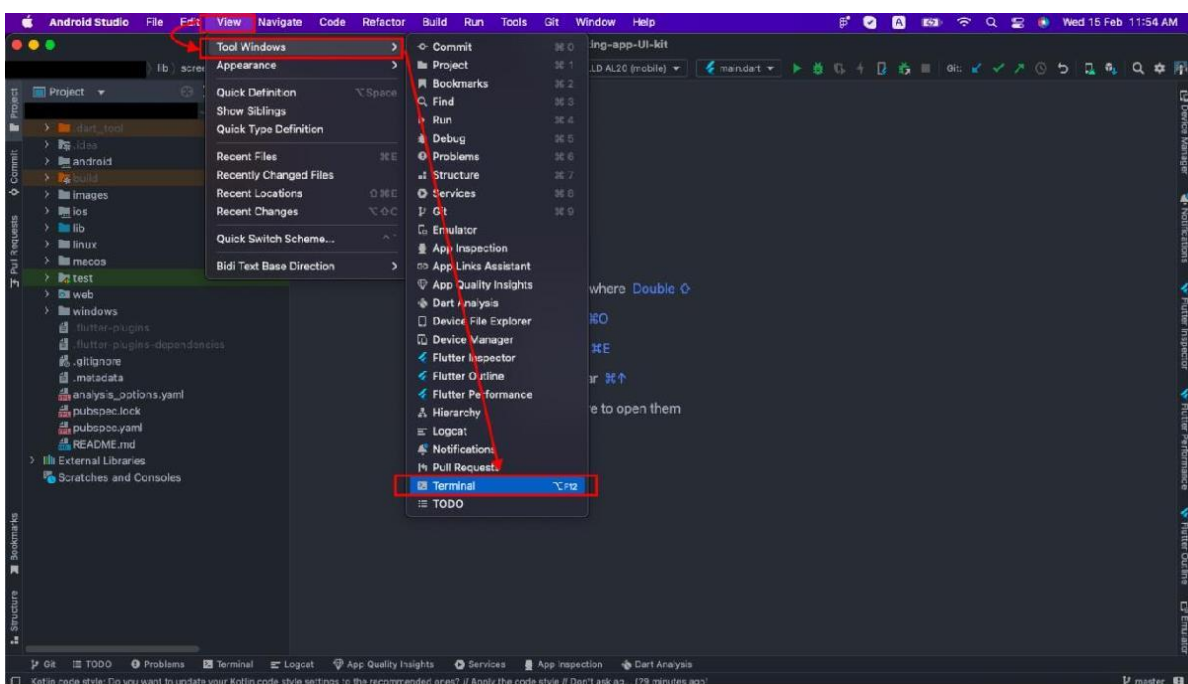
Also if required, you have to give the sdk path, & allow for the configuration.



1. You can do it from your terminal also. For that you have to open the terminal from the topbar.

Select:

View → Tool Windows → Terminal as shown in the Next figure.



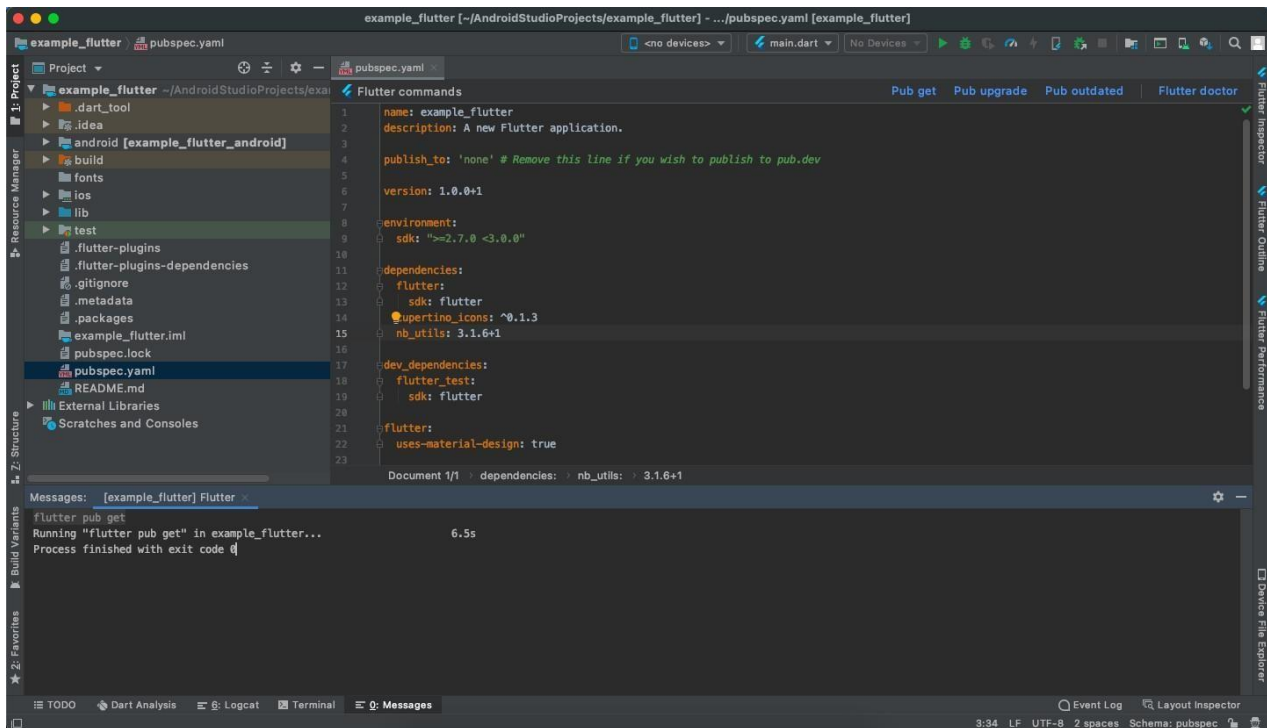
Now type flutter **pub get** on terminal and press enter.

This will get the dependencies for the application.

Make sure that you are connected to the internet while performing this task

Build and Run App

All below steps must be followed to build and run applications.

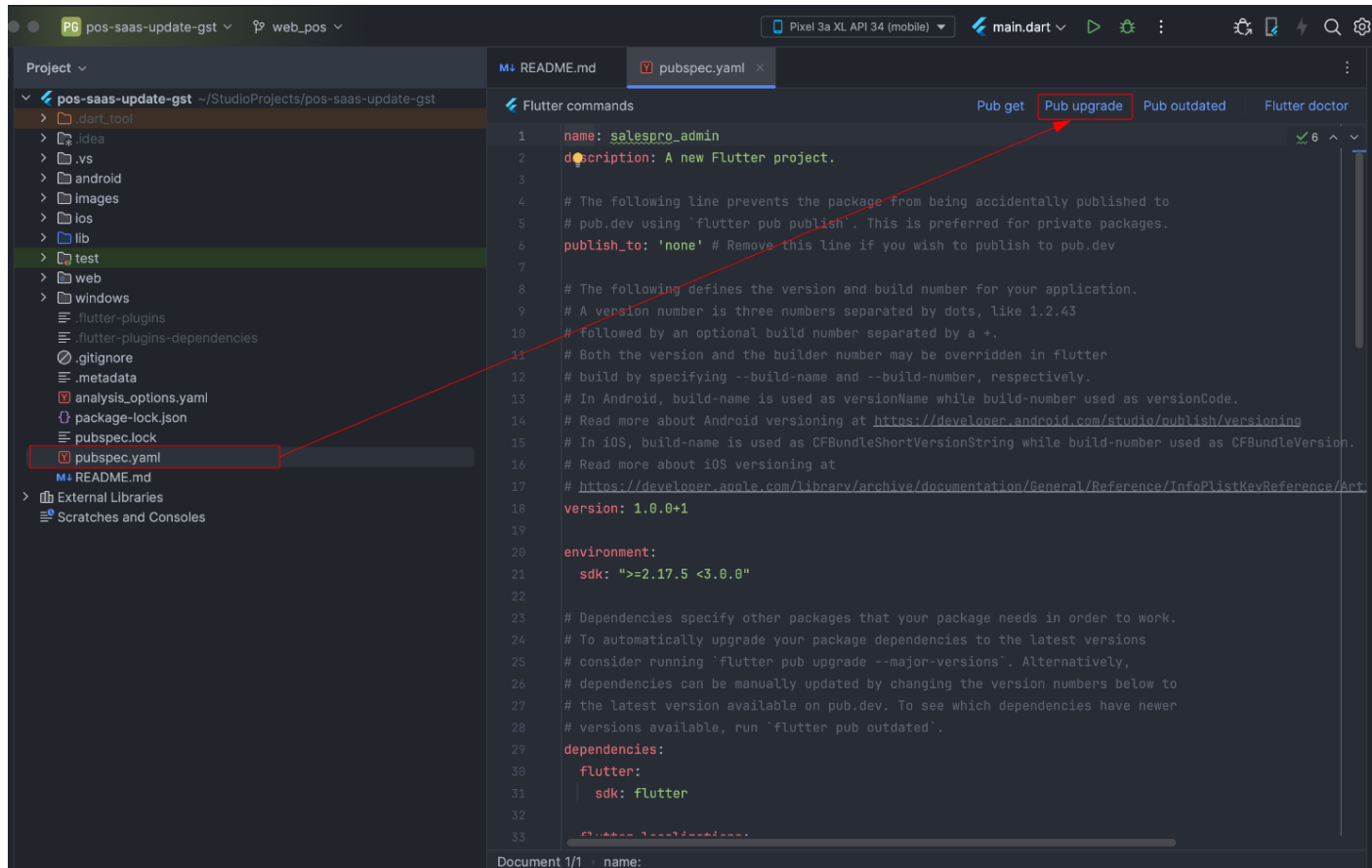


1. Locate the main Android Studio toolbar.
2. In the target selector (<no device selected>), select an Android device for running the app. Or if you want to run the web app select the available browser from the dropdown menu. If none are listed as available, select Tools > Android > AVD Manager and create one there. For details, see Managing AVDs.
3. Click the run icon in the toolbar, or invoke the menu item Run > Run.



After the app build completes, you'll see the app on your device.

How to update this Project Version



To update your project version, go to the **pubspec.yaml** file. And click on the **Pub upgrade** button at the top. It will automatically update all the packages version. Make sure your pc is connected with the internet.

To update your project version and dependencies in a Flutter project, follow these steps:

1. Open the **pubspec.yaml** File:

- Locate the **pubspec.yaml** file in the root directory of your Flutter project.
- Open it in your preferred text editor or IDE.

2. Update the Project Version:

- Find the **version** field at the top of the **pubspec.yaml** file.
- Change the version number to your desired version. For example: `version: 1.1.0+2`
- The version format is **major.minor.patch+build**.

Run **pub upgrade**:

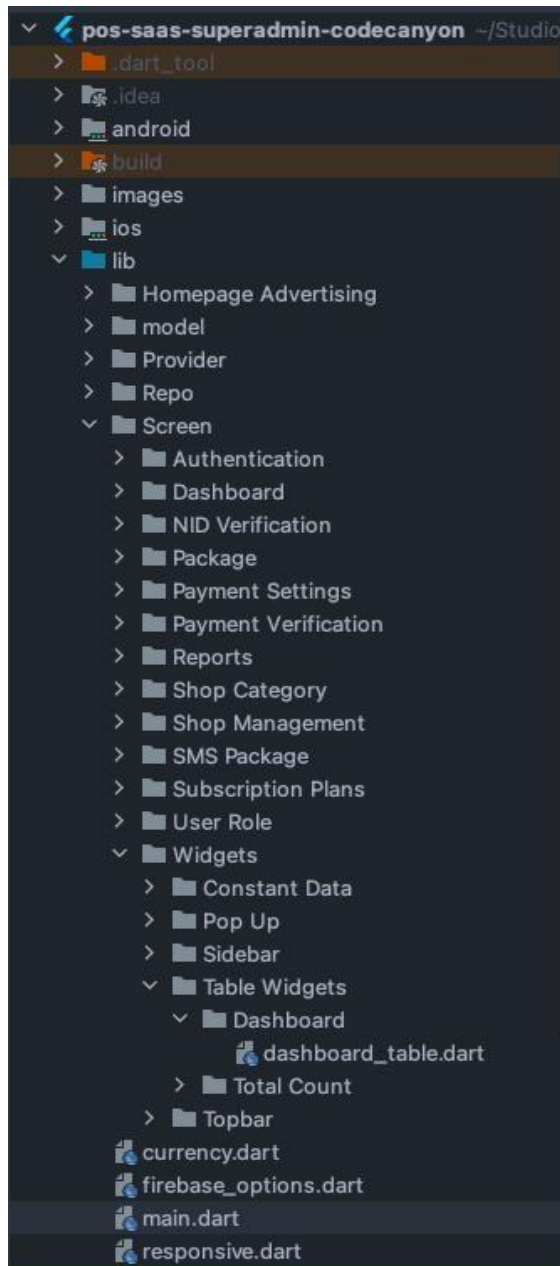
- Ensure your computer is connected to the internet.
- In your IDE, locate and click the **Pub upgrade** button. This button is typically found at the top of the editor or in the **pubspec.yaml** tab.
- Alternatively, you can run the following command in your terminal from the project directory:

```
flutter pub upgrade
```

This command updates all packages to the latest versions allowed by the constraints in **pubspec.yaml**.

POS Saas Super Admin

Project Structure:

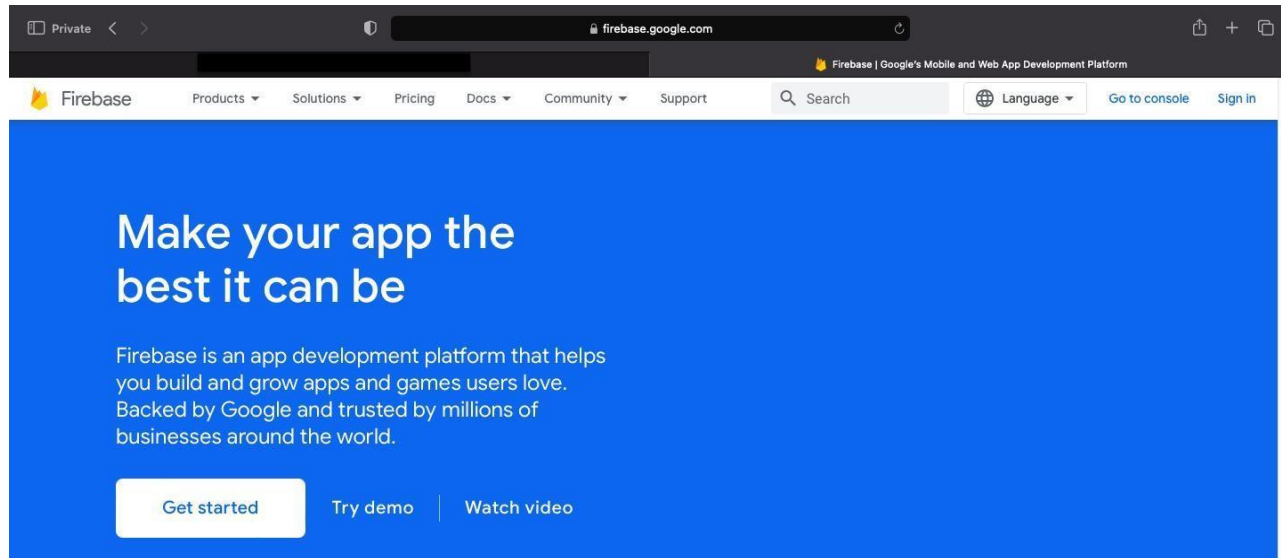


- **Images:** Apps/widgets/snippets images as well as appicon are stored here.
- **Lib:** Applications main files and directories are located here.
 - **Screen:** Full app's UI codes are stored here.
 - **Widgets:** All the reusable widgets are stored here, like(Top bar, Side bar).Also the constant file located in the folder. Where the all the colors are stored which is used to design the app.
 - In **firebase_option.dart** all the code stored which is related with the firebase.From here you can easily change the firebase credentials.
 - **main.dart:** All the routes have been declared here.

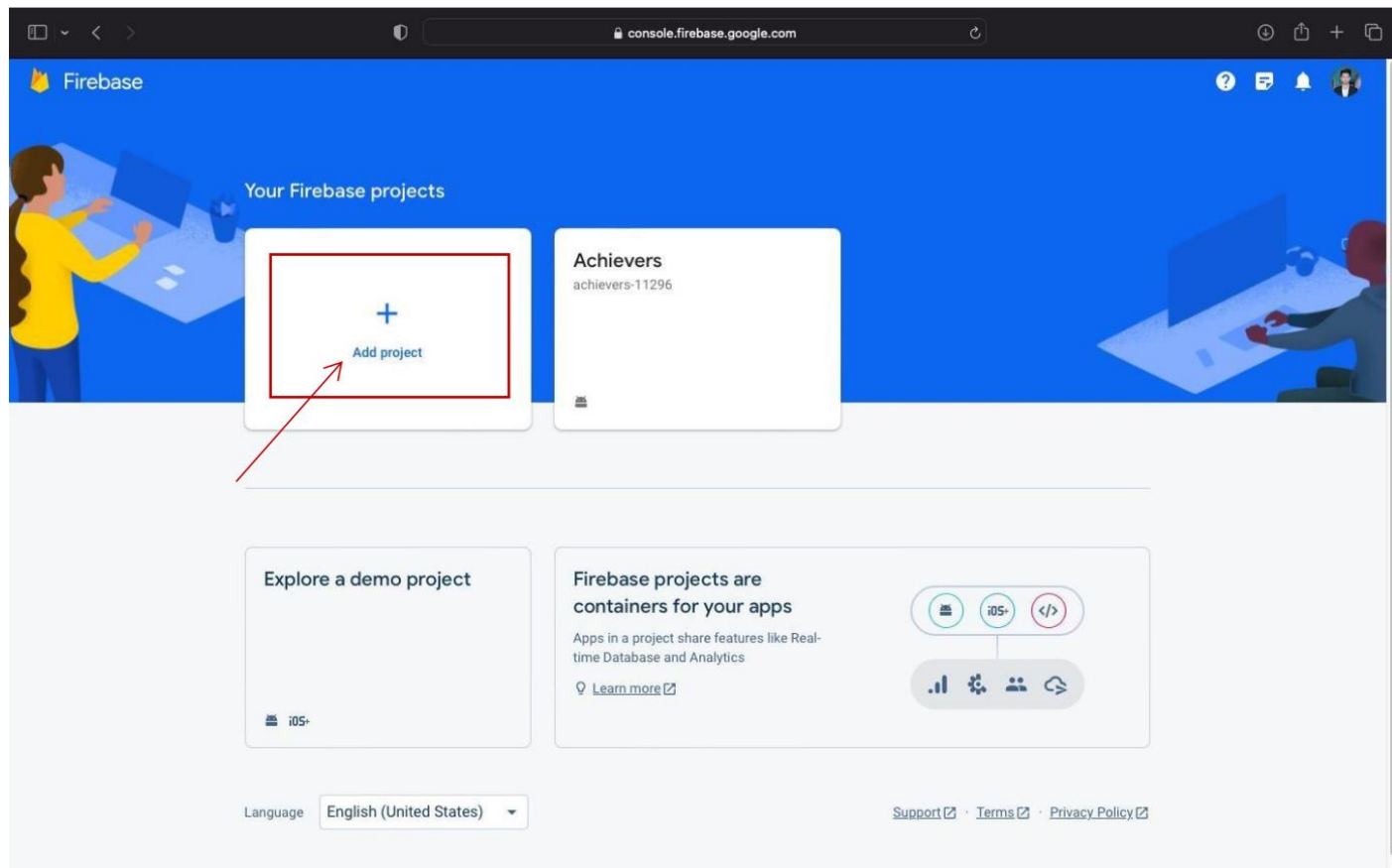
Firestore Setup

Step 1

First of all you have to go to <https://console.firebase.google.com/> and click on Go to the console which is on the right side of the top bar menu.



After tapping on the Go to console a new page will be open.



Now you have to create a new project which you want to connect with the app. Click on Add project to create a new one.

×

Create a project (Step 1 of 3)

Let's start with a name for your project[ⓘ]

Project name

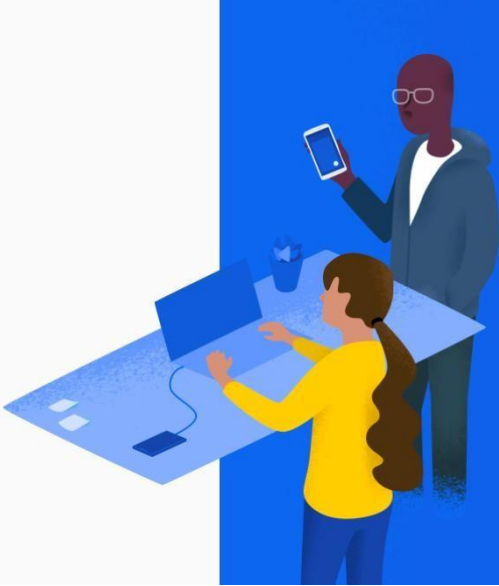
Enter your project name

enter-your-project-name-f8d87

☒

I confirm that I will use Firebase exclusively for purposes relating to my trade, business, craft, or profession.

Continue



Here you have enter a name for your project.then click on the ckeck box and press continue to go to the next step.

×

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[ⓘ]

Crash-free users[ⓘ]

Event-based Cloud Functions triggers[ⓘ]

Free unlimited reporting[ⓘ]

☒

Enable Google Analytics for this project

Recommended

Previous

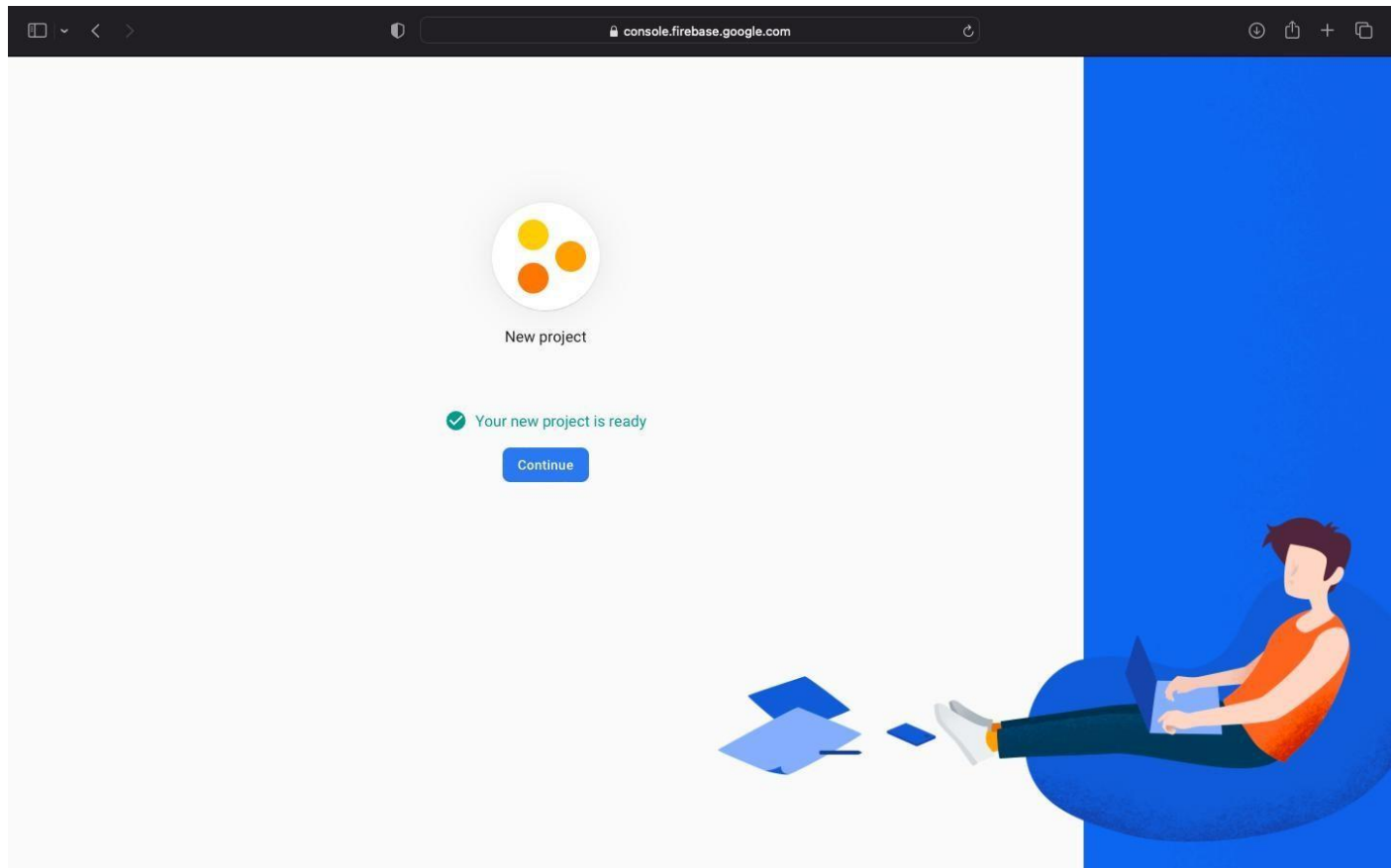
Continue



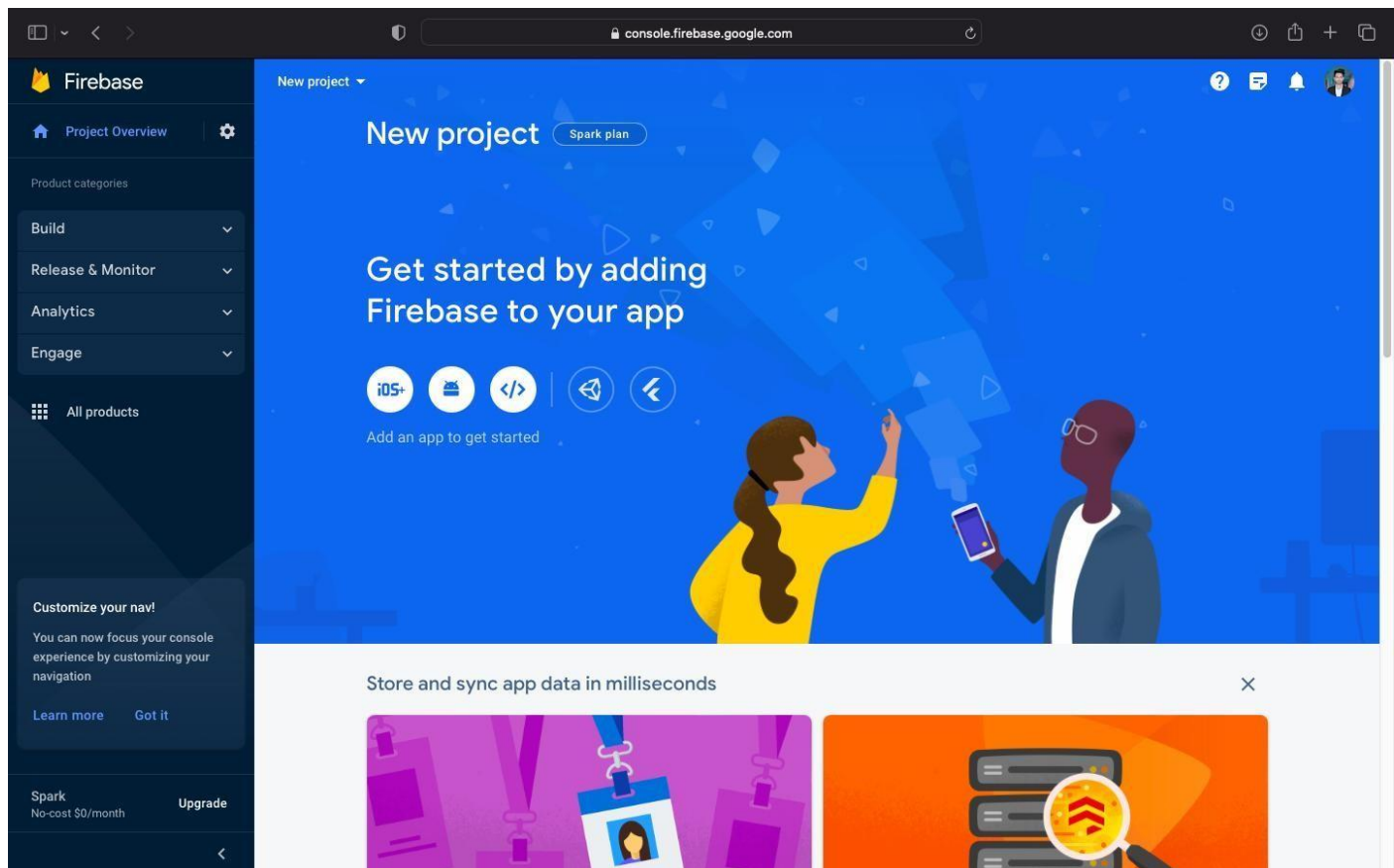
Here if you want you can enable or disable the google analytics for your project. We recommended you to disable it for now and press continue.

9

A new project is created for you. Press continues to view the project.



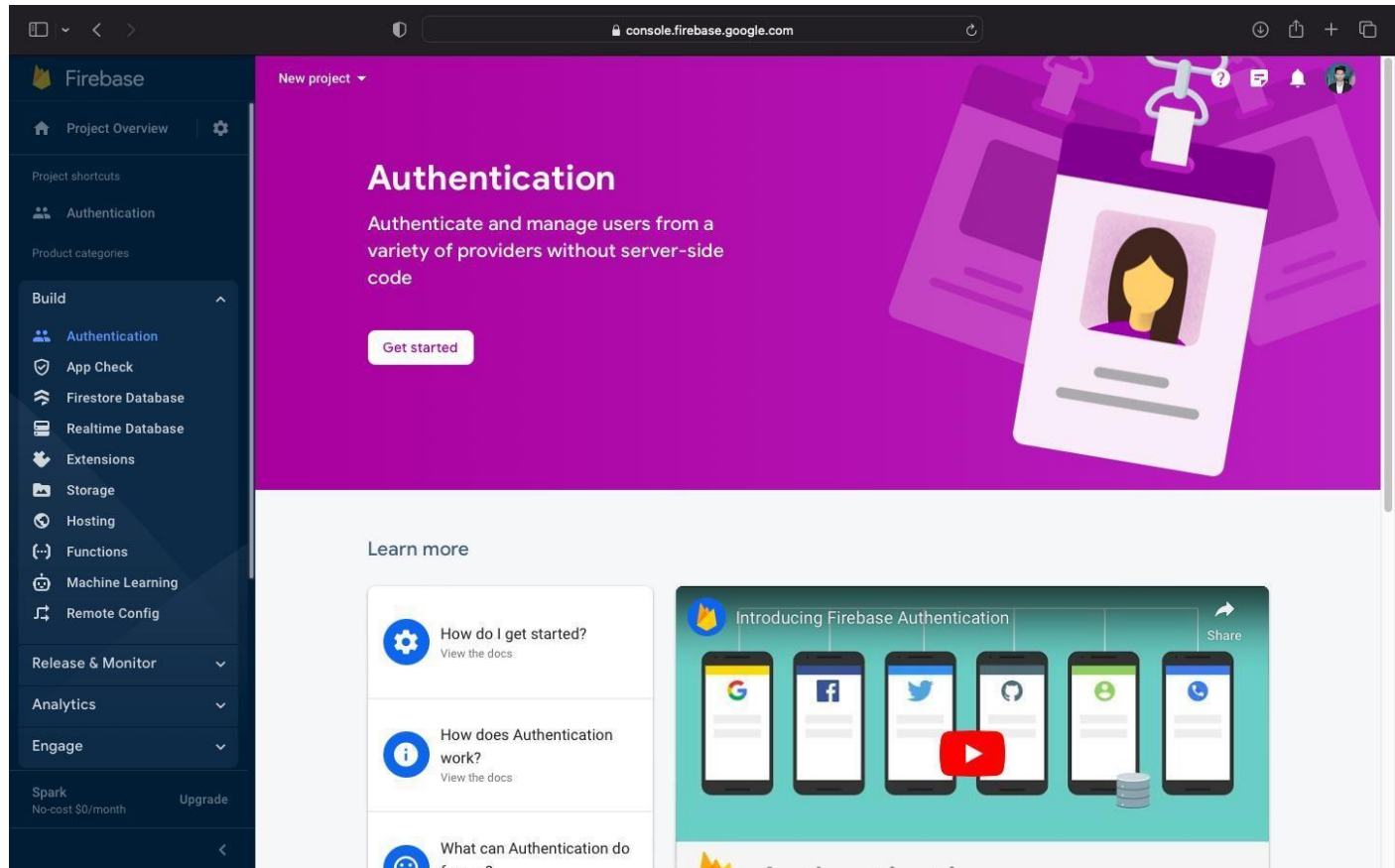
The environment will look like this.



Step 2

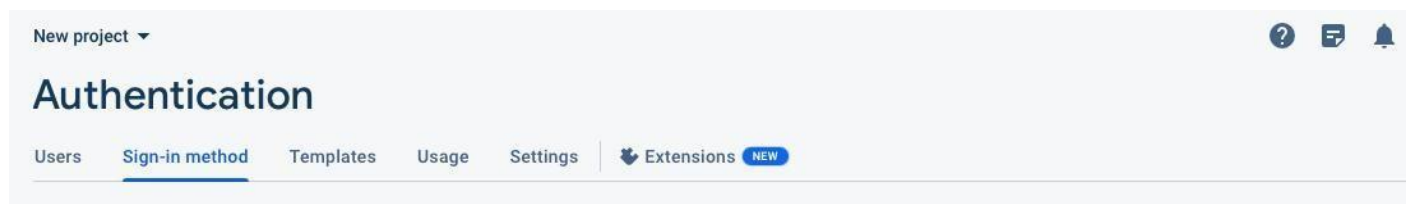
Note that It is required to complete all of the steps one by one before generate the credentials for the app.

Now from the sidebar click on the **Build** section. then select the **Authentication** from the list inside the build section.

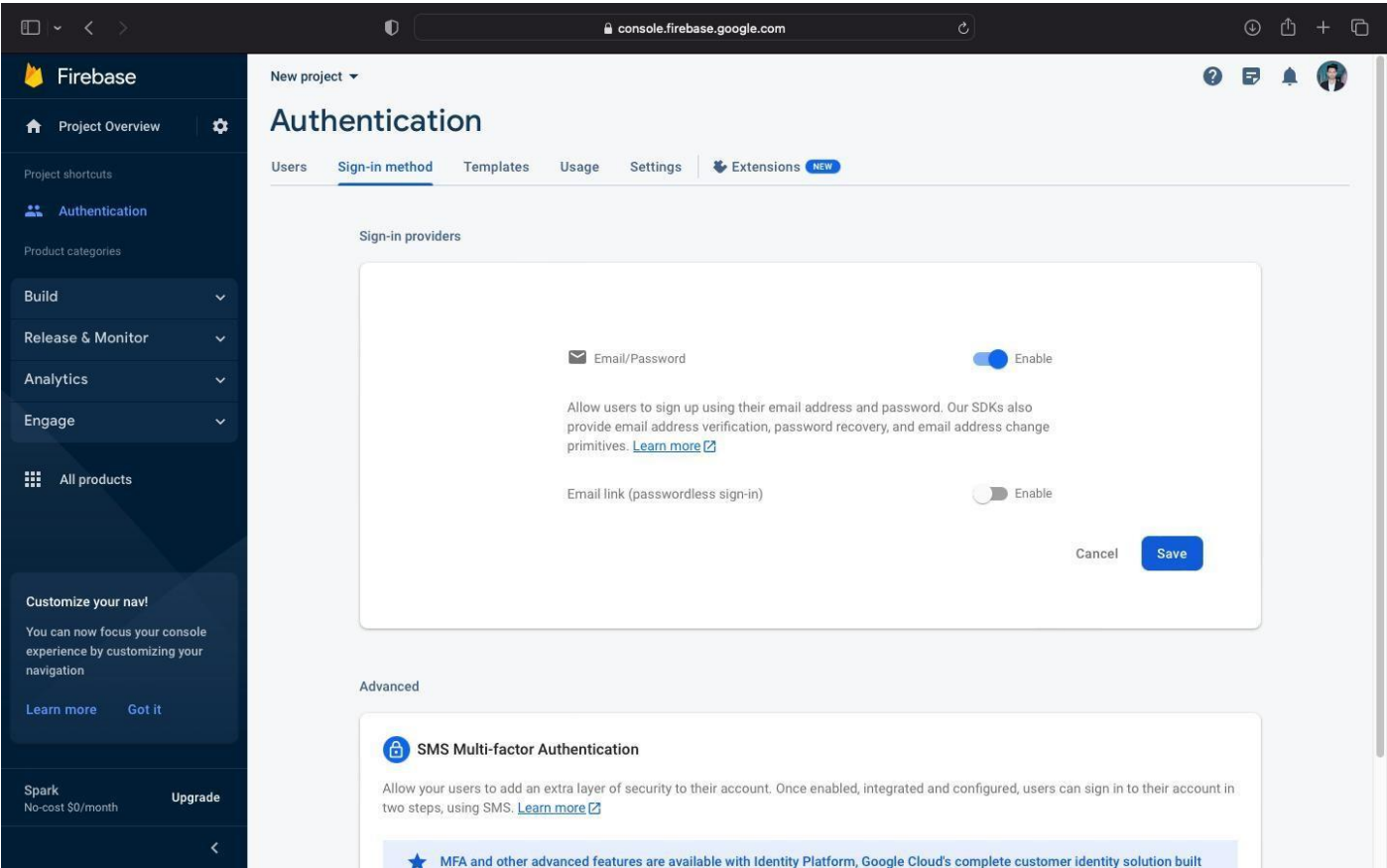
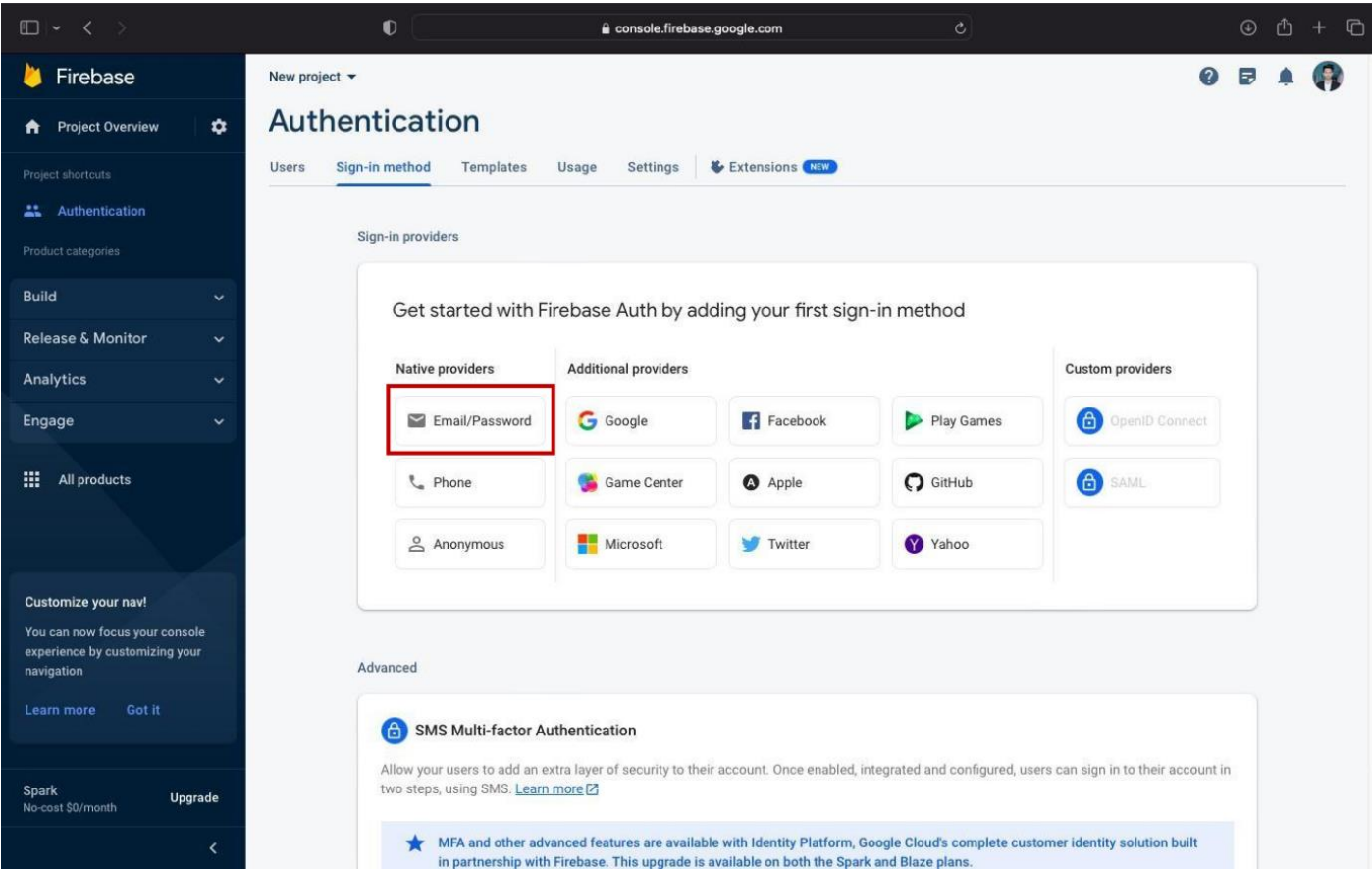


Then a new page will be open like this with the Title named **Authentication**. Press the get started to go to the next step.

A new page will appear with a tab bar. Make sure that you are in the second tab (Sign-in-method).

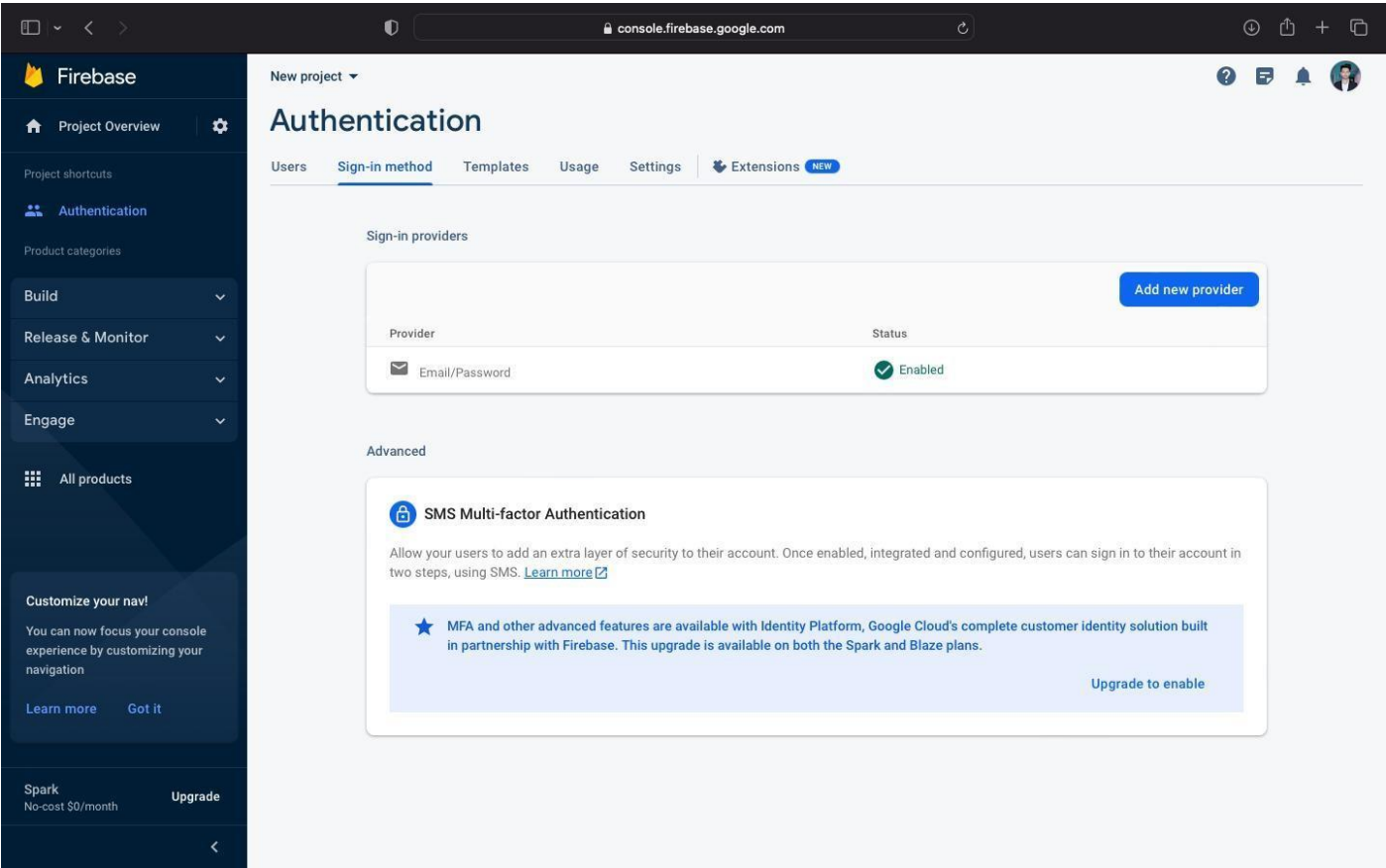


Select **Email/Password** from the list.



Enable the Email/Password switch and hit the save button.

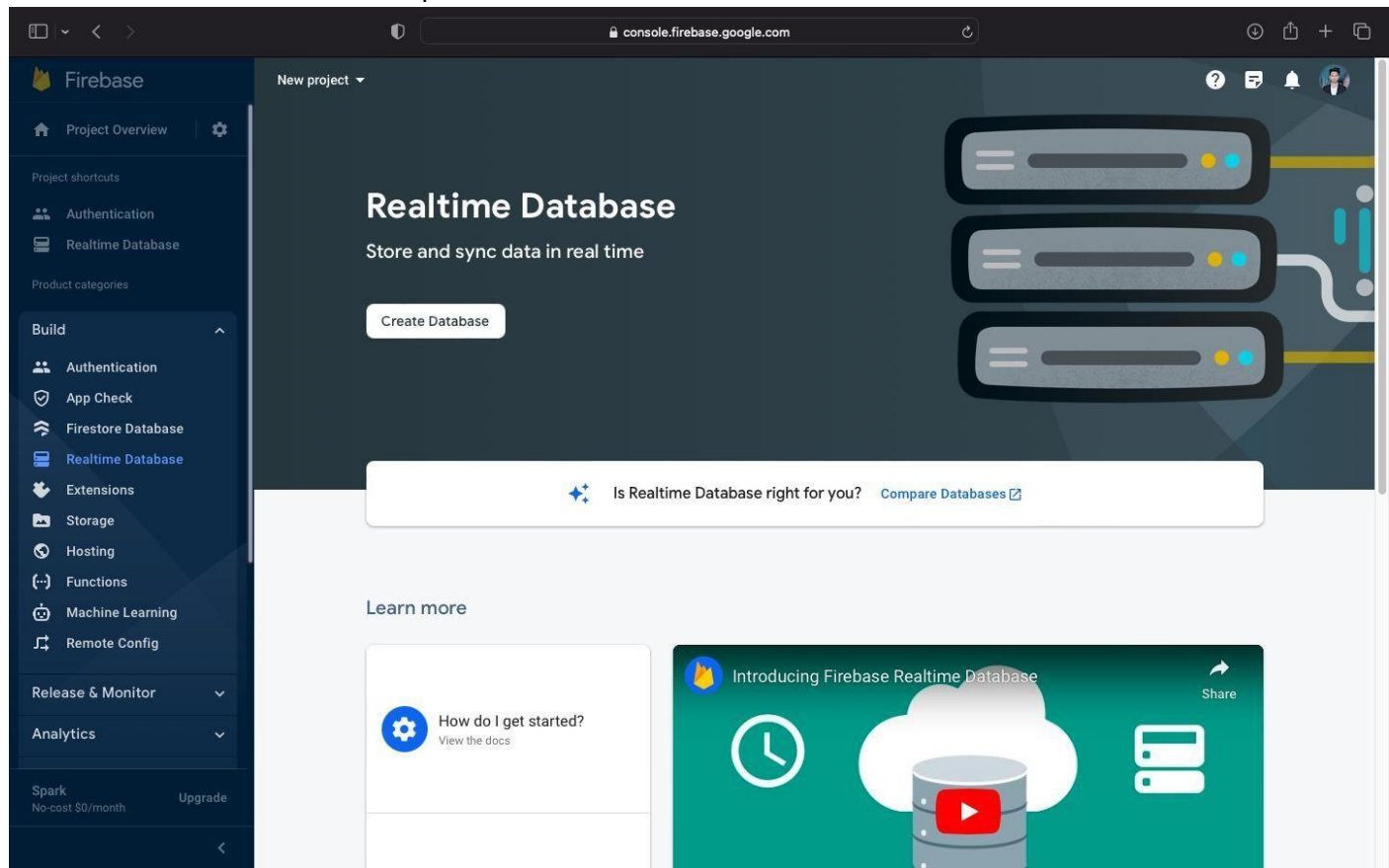
A new page will be open like this. Here Email/Password are enabled now.



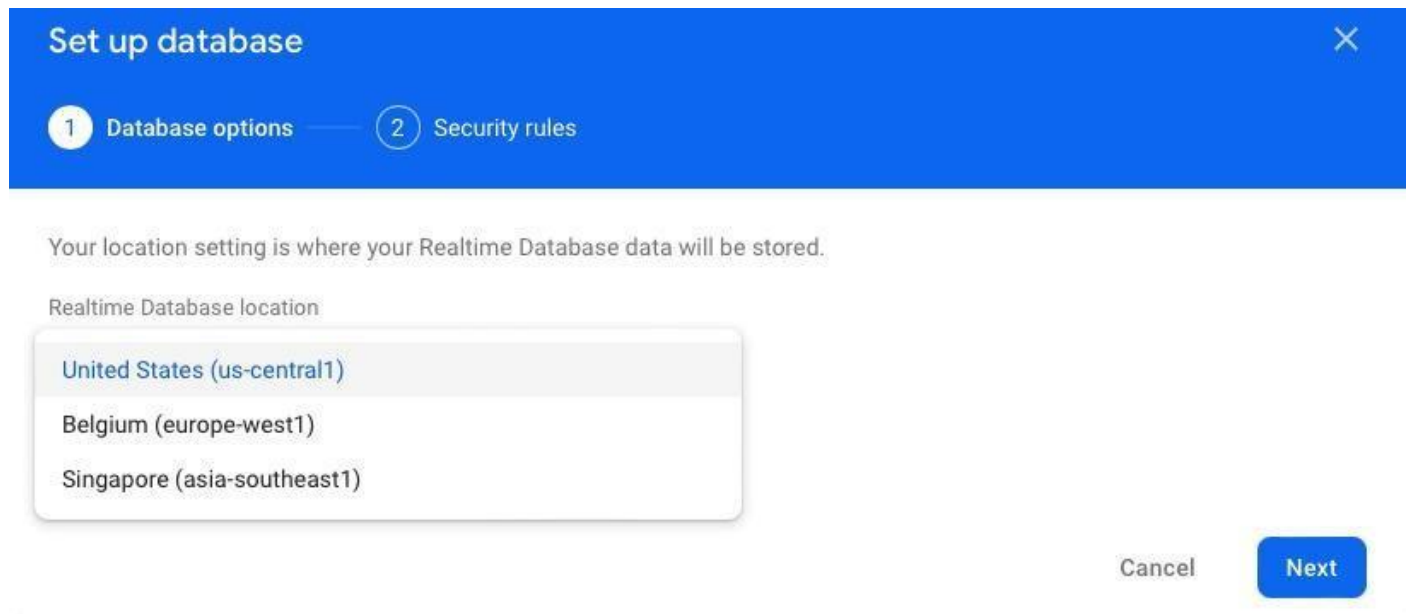
Now the Email/Password is enabled. To setup the project without any error make sure that This authentication type is enabled.

Step 3

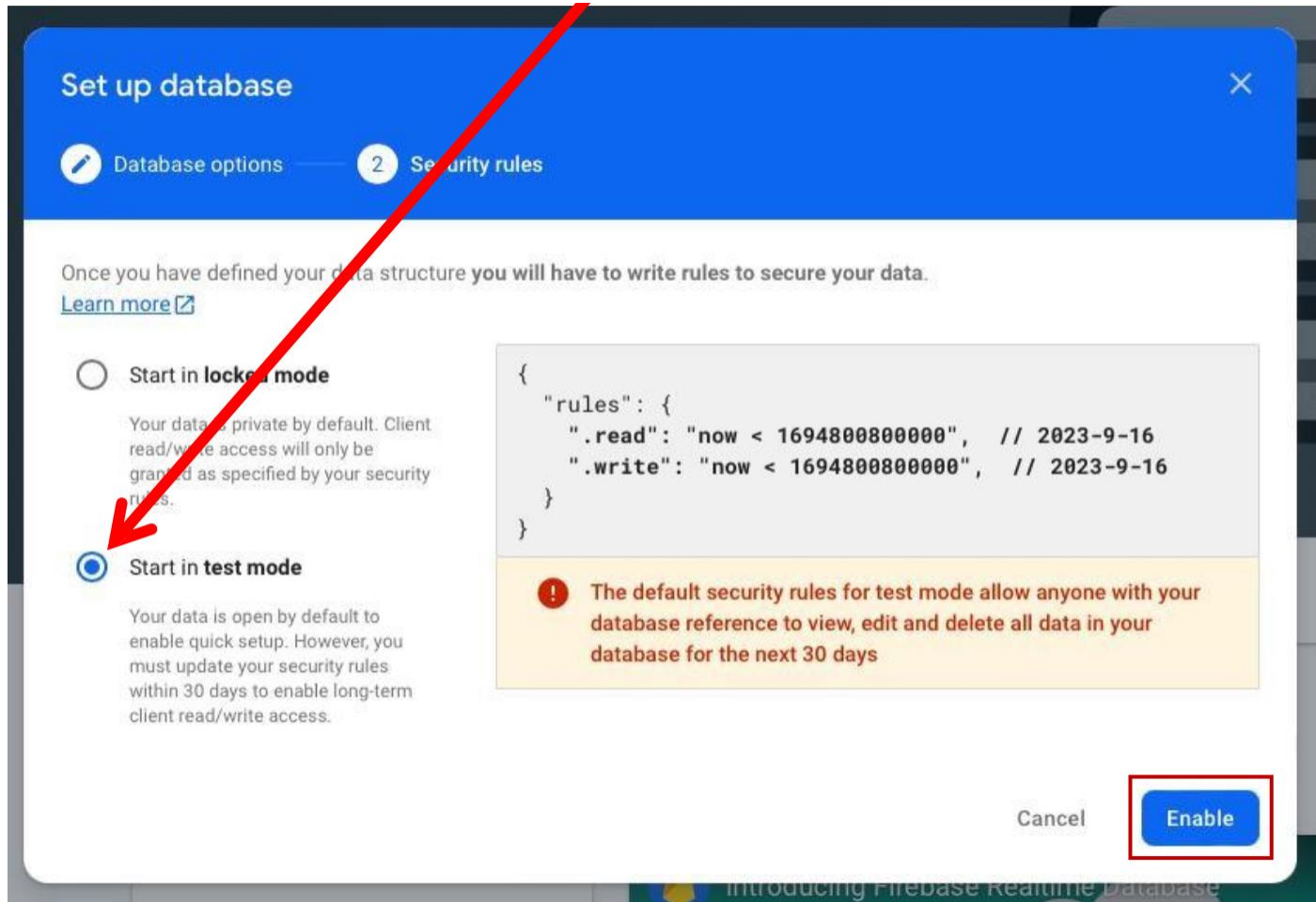
Now from the sidebar click on the **Build** section. then select the **Realtime Database** from the list inside the build section. And press on the **Create Database** button.



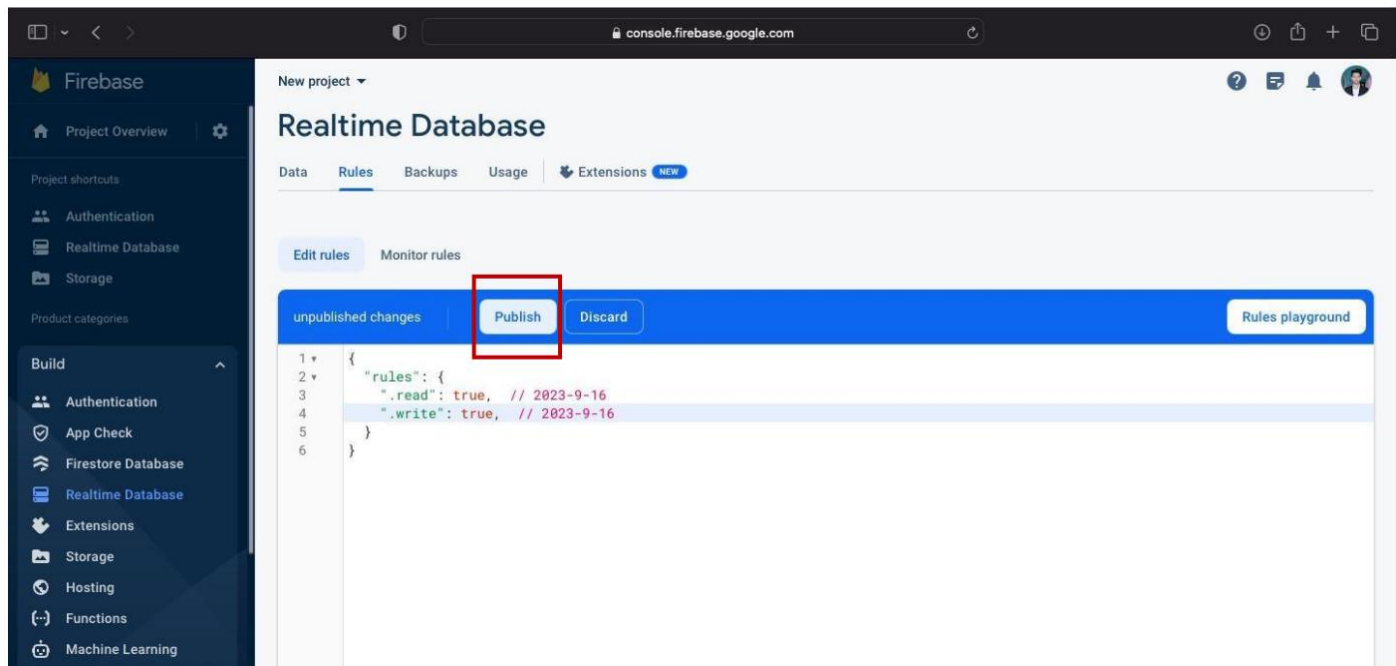
Select any location from the dropdown button and hit the next button.



Here please select the radio button **Start in text mode** and press **Enable**.

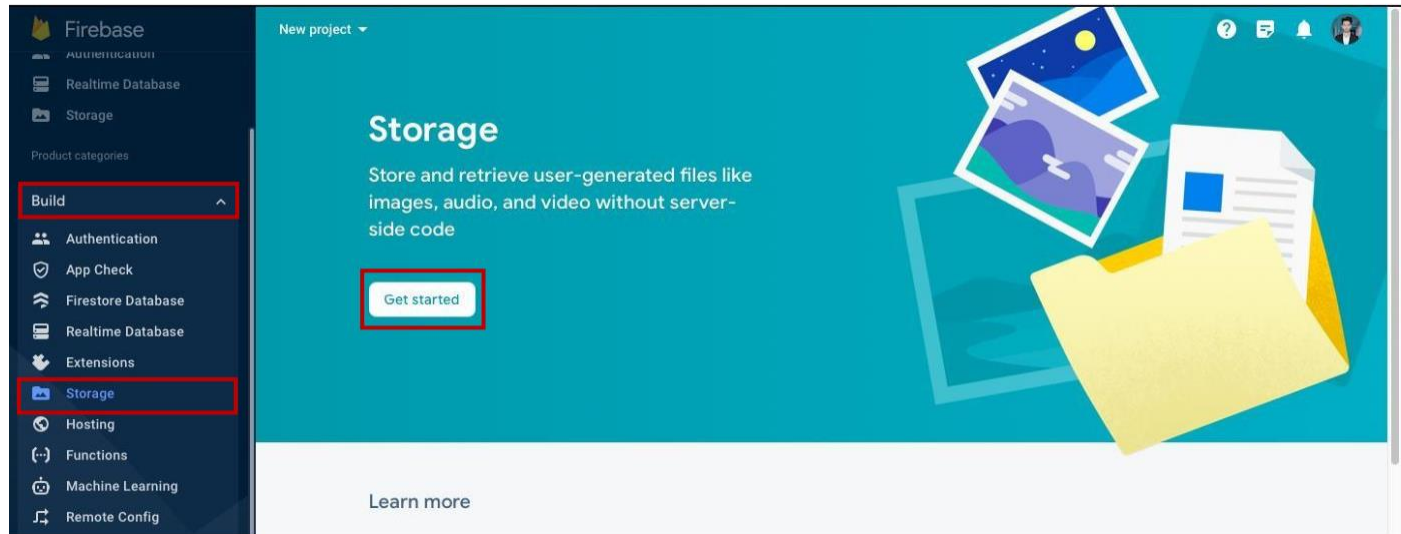


Now in **Realtime Database** page, please select the **Rules** tab and change the read and write section to the true as shown in the picture. And press on **Publish**.

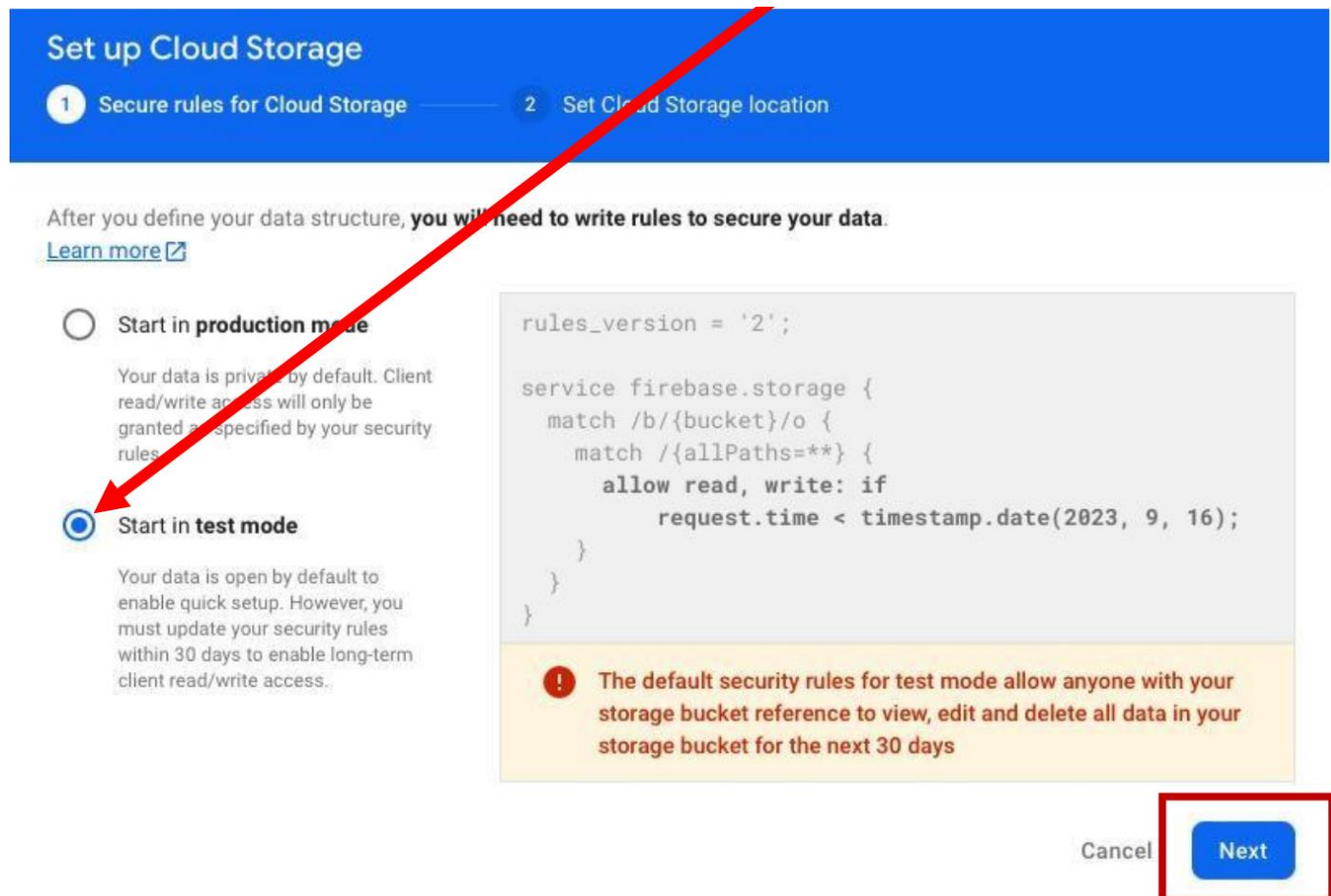


Step 4

Now from the sidebar click on the **Build** section. then select the **Storage** from the list inside the build section.



A popup will be arise, here select the radio button for **Start in test mode** and press the next button.



And then the second popup will arise. And choose the cloud storage location from the dropdown. And press on the **Done** button.

Set up Cloud Storage

Secure rules for Cloud Storage

2 Set Cloud Storage location

Your location setting is where your default Cloud Storage bucket and its data will be stored.

 After you set this location, you cannot change it later. This location setting will also be the default location for Cloud Firestore.

[Learn more](#)

Cloud Storage location

nam5 (us-central)

Blaze Plan customers can choose other locations for additional buckets [click here to upgrade](#)

Cancel

Done

Now in the Storage page, go to the second tab named **Rules**. And Increase the date. Initially it gives only 1 month but you can increase it as you want

Firestore

Project Overview

Project shortcuts

Authentication

Realtime Database

Storage

Product categories

Build

Release & Monitor

Analytics

Engage

All products

Customize your nav!

You can now focus your console experience by customizing your navigation

Learn more

Got it

Spark

No-cost \$0/month

Upgrade

New project

Storage

Files

Rules

Usage

Extensions

Write Security Rules that control access to Storage based on the contents of your Firestore Database.

Learn more

×

Edit rules

Monitor rules



Guard your data with rules that define who has access to it and how it is structured

[View the docs](#)

```
1 rules_version = '2';
2
3 // Craft rules based on data in your Firestore database
4 // allow write: if firestore.get(
5 //   /databases/(default)/documents/users/${request.auth.uid}).data.isAdmin;
6 service firebase.storage {
7   match /b/{bucket}/o {
8
9     // This rule allows anyone with your Storage bucket reference to view, edit,
10    // and delete all data in your Storage bucket. It is useful for getting
11    // started, but it is configured to expire after 30 days because it
12    // leaves your app open to attackers. At that time, all client
13    // requests to your Storage bucket will be denied.
14    //
15    // Make sure to write security rules for your app before that time, or else
16    // all client requests to your Storage bucket will be denied until you Update
17    // your rules
18    match /{allPaths=**} {
19      allow read, write: if request.time < timestamp.date(2023, 9, 16);
20    }
21  }
22 }
```

Rules Playground

Experiment and explore with Security Rules

unpublished changes

Publish

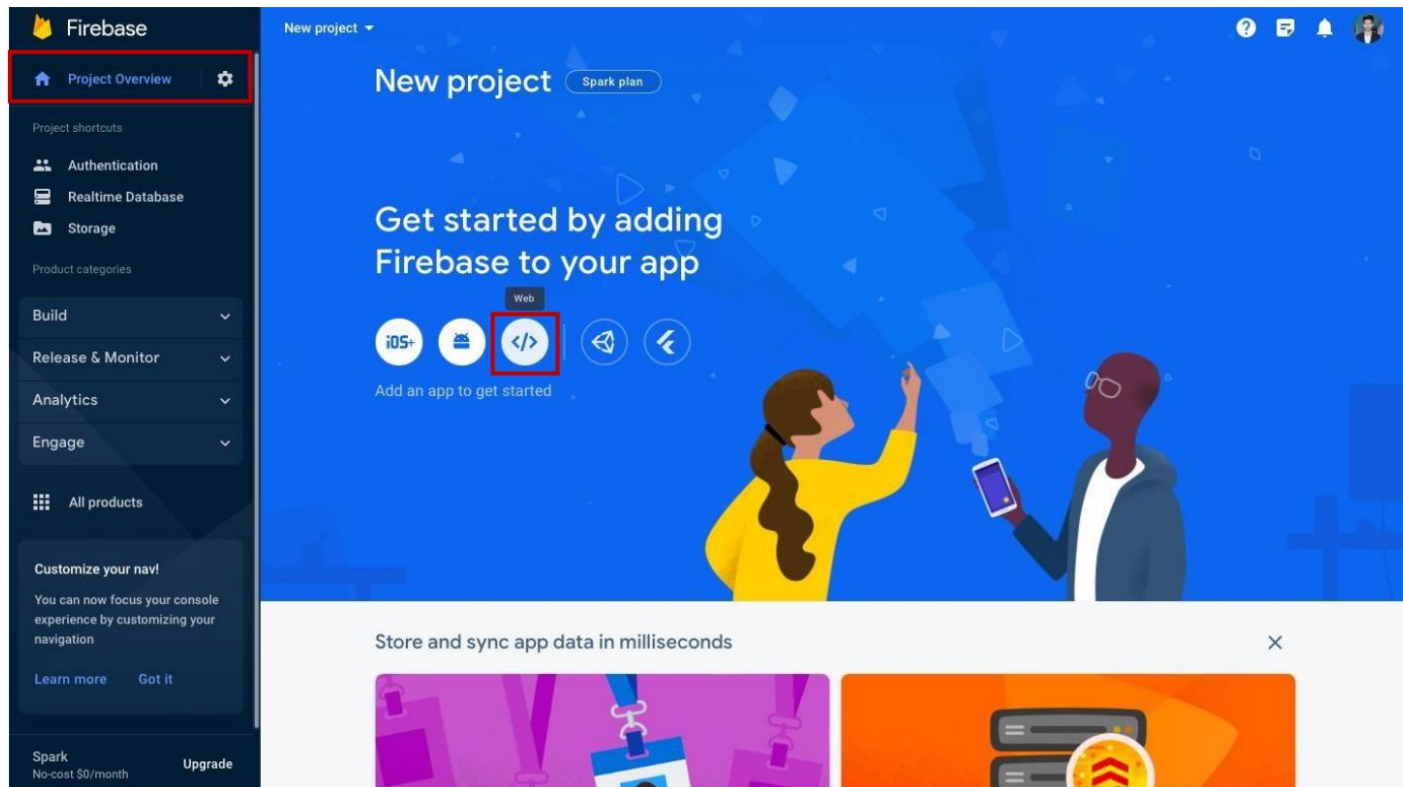
Discard

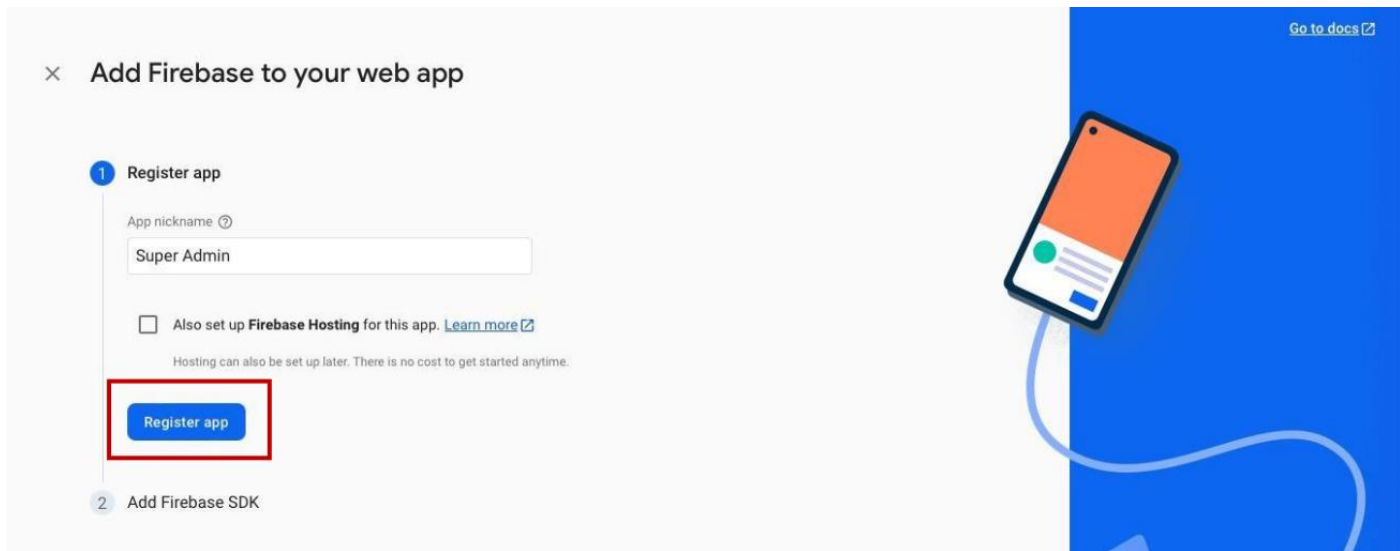
```
1 rules_version = '2';
2
3 // Craft rules based on data in your Firestore database
4 // allow write: if firestore.get(
5 //   /databases/(default)/documents/users/$(request.auth.uid)).data.isAdmin;
6 service firebase.storage {
7   match /b/{bucket}/o {
8
9     // This rule allows anyone with your Storage bucket reference to view, edit,
10    // and delete all data in your Storage bucket. It is useful for getting
11    // started, but it is configured to expire after 30 days because it
12    // leaves your app open to attackers. At that time, all client
13    // requests to your Storage bucket will be denied.
14    //
15    // Make sure to write security rules for your app before that time, or else
16    // all client requests to your Storage bucket will be denied until you Update
17    // your rules
18    match /{allPaths=**} {
19      allow read, write: if request.time < timestamp.date(2025, 9, 16);
20    }
21  }
22 }
```

Don't forget to publish it. So press on **publish** button.

Step 5

Now go back to the **Project overview** section and press on the web button.

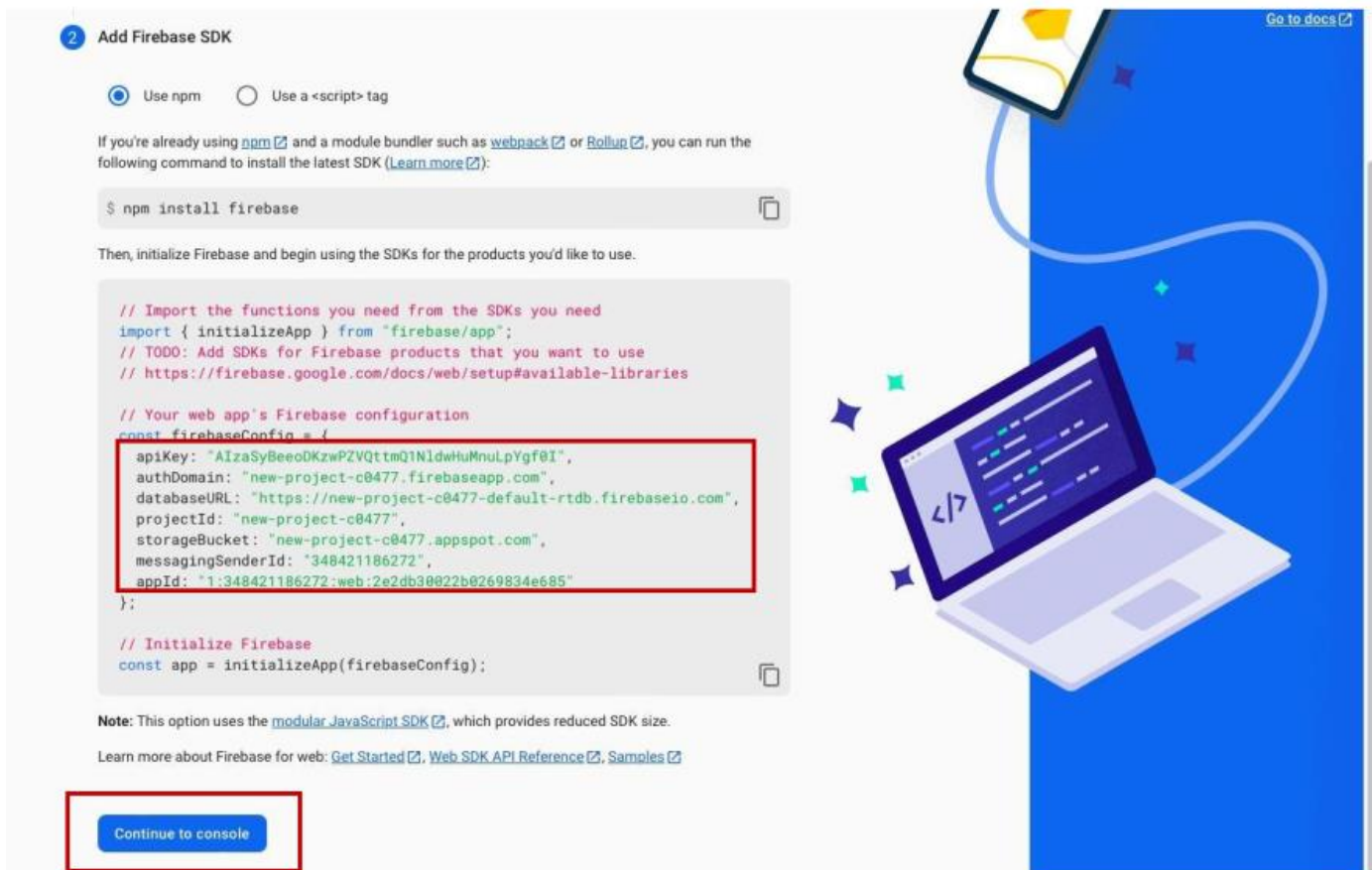




Now write a name for the app. No need to check the firebase hosting for this project and press on the **Register app** button to go to the next step.

Then it will generate the all credentials for the app which you have to copy and store in a safe place. Don't forget to copy the credentials.

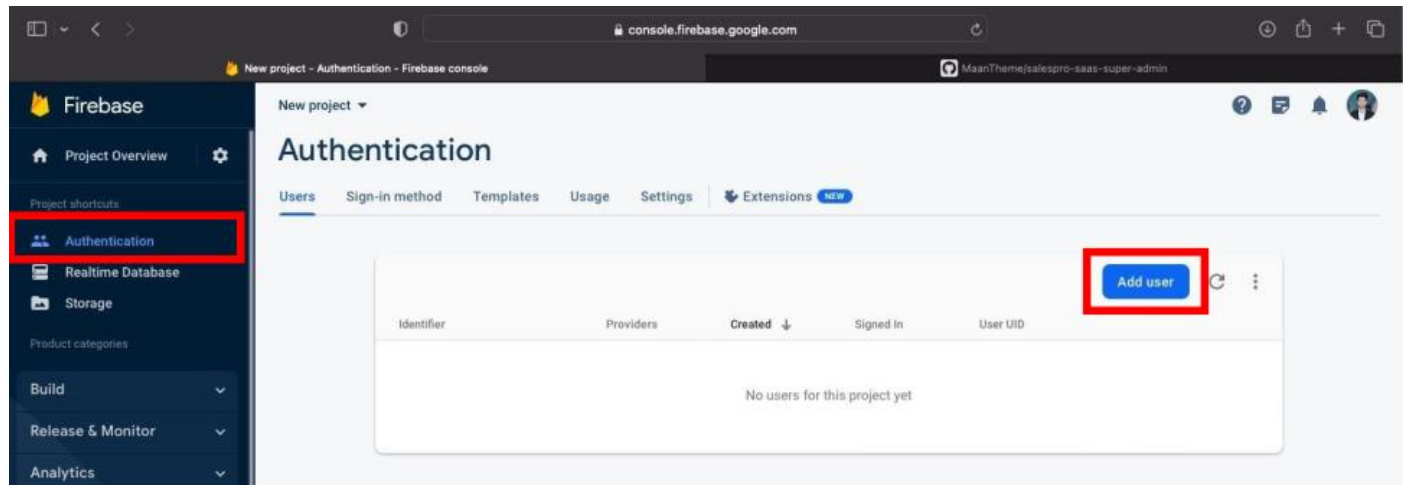
Copy all the codes which is inside the firebaseConfig (Which credentials generate for your project) & press the **Continue to console** button. For example you can see the screen below.



**This is for an example not use
this code on your project**

Step 6

Now again come to the sidebar and go to the **Authentication** section. Here you have to add a new user for your project. Press **Add user** button.



Enter a valid email & password to add a user. And note the email and password to use later.

A screenshot of the 'Add an Email/Password user' form. It has two input fields: 'Email' and 'Password'. The 'Email' field is highlighted with a yellow box. At the bottom right, there are 'Cancel' and 'Add user' buttons. The 'Add user' button is highlighted with a yellow box.

Keep in mind that without this email you can't log in to the Super Admin app panel.

Step 7

To complete setup you have to go back to your project file which you opened in your environment (Android Studio, Visual code or etc).

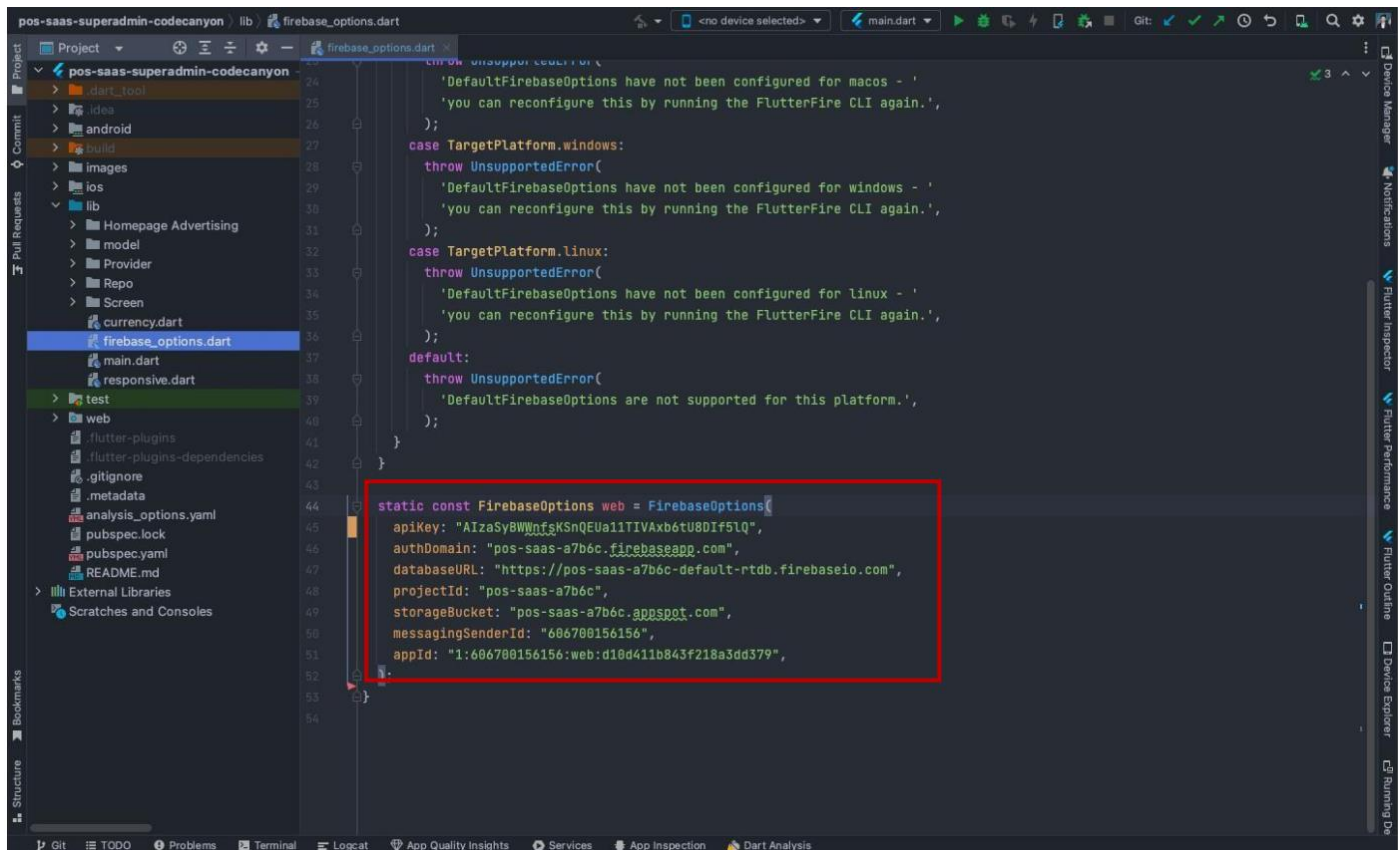
You have to open the dart file named “**firebase_options.dart**”.

Location: Project/lib

(You can also see the project structure to find it easily)

After open the **firebase_option.dart** file, come to the last section of the code, marked with the red color and paste all the credentials.(api key, authDomain, databaseURL, projectId, storageBucket, messagingSenderId,appId) which you copied in the last section.

Make sure that you have pasted the correct code.

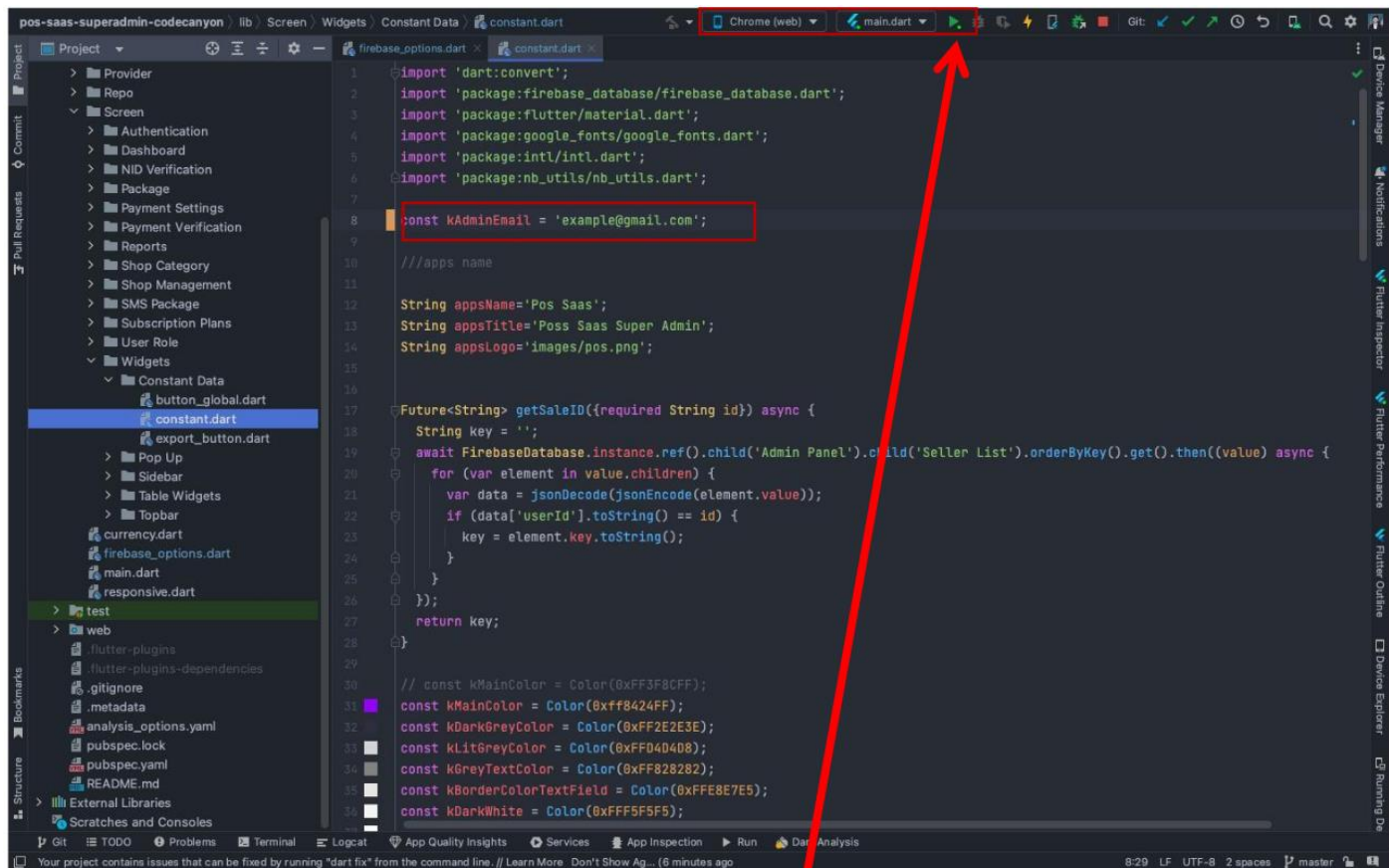


```
24 throw UnsupportedError(
25   'DefaultFirebaseOptions have not been configured for macos - '
26   'you can reconfigure this by running the FlutterFire CLI again.',
27 );
28
29 case TargetPlatform.windows:
30   throw UnsupportedError(
31     'DefaultFirebaseOptions have not been configured for windows - '
32     'you can reconfigure this by running the FlutterFire CLI again.',
33   );
34
35 case TargetPlatform.linux:
36   throw UnsupportedError(
37     'DefaultFirebaseOptions have not been configured for linux - '
38     'you can reconfigure this by running the FlutterFire CLI again.',
39   );
40
41 default:
42   throw UnsupportedError(
43     'DefaultFirebaseOptions are not supported for this platform.',
44   );
45
46 static const FirebaseOptions web = FirebaseOptions(
47   apiKey: "AIzaSy8W0nfgKSnQEUa1TIVAx6tU8DI5lQ",
48   authDomain: "pos-saas-a7b6c.firebaseio.com",
49   databaseURL: "https://pos-saas-a7b6c-default-rtdb.firebaseio.com",
50   projectId: "pos-saas-a7b6c",
51   storageBucket: "pos-saas-a7b6c.appspot.com",
52   messagingSenderId: "606700156156",
53   appId: "1:606700156156:web:d10d411b843f218a3dd379",
54 );
```

Now you have to open the **constant.dart** file.

Location: **Project/lib/screen/widgets/Constant Data/constant.dart**

After open the **constant.dart**, you have to change the email address. Replace the email **example@gmail.com** and write the email address which you used to create the user in firebase authentication panel. Make sure that you have written the valid & the same email.



Then you have to run the app. From the top bar of your environment select a platform were you want to run the application. And press on the Run button

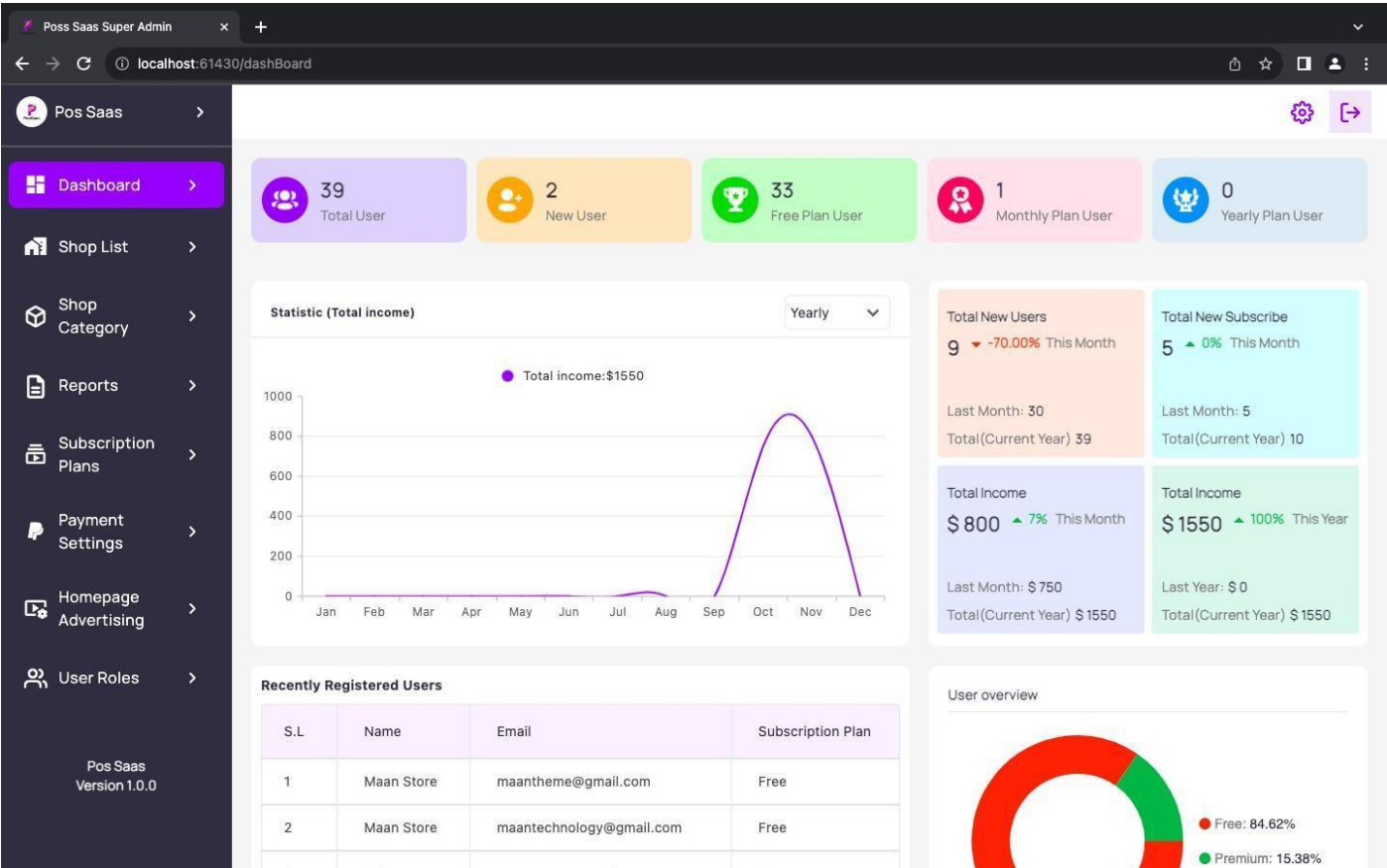
The screenshot shows the 'Pos Saas Login Panel' with the following elements:

- PosSaas logo at the top.
- Title: Pos Saas Login Panel
- Email input field.
- Password input field with a toggle icon for visibility.
- A green 'Login' button.
- A link for 'Forgot password?' with a lock icon.

After opening the app enter the user's email and password. Which you have given to add a new user.

After Login you have to follow some steps and it is required to setting up your app successfully.

After setup your firebase please visit all the menu from sidebar at least one time.It is required for the first time only.



Keep in mind, Add at least one category if the category list is empty.

The screenshot shows the 'Shop Category' page in the 'Pos Saas Super Admin' interface. A modal titled 'Add new Category' is open, allowing the user to add a new category. The modal contains input fields for 'Category Name' and 'Description' (with a placeholder 'Enter Category Description'). At the bottom of the modal are 'CANCEL' and 'SAVE' buttons. In the background, the 'SHOP LIST' table is visible, showing columns for S.L, CATEGORY NAME, DESCRIPTION, CREATED BY, and ACTION. An 'ADD NEW CATEGORY' button is located in the top right corner of the table area.

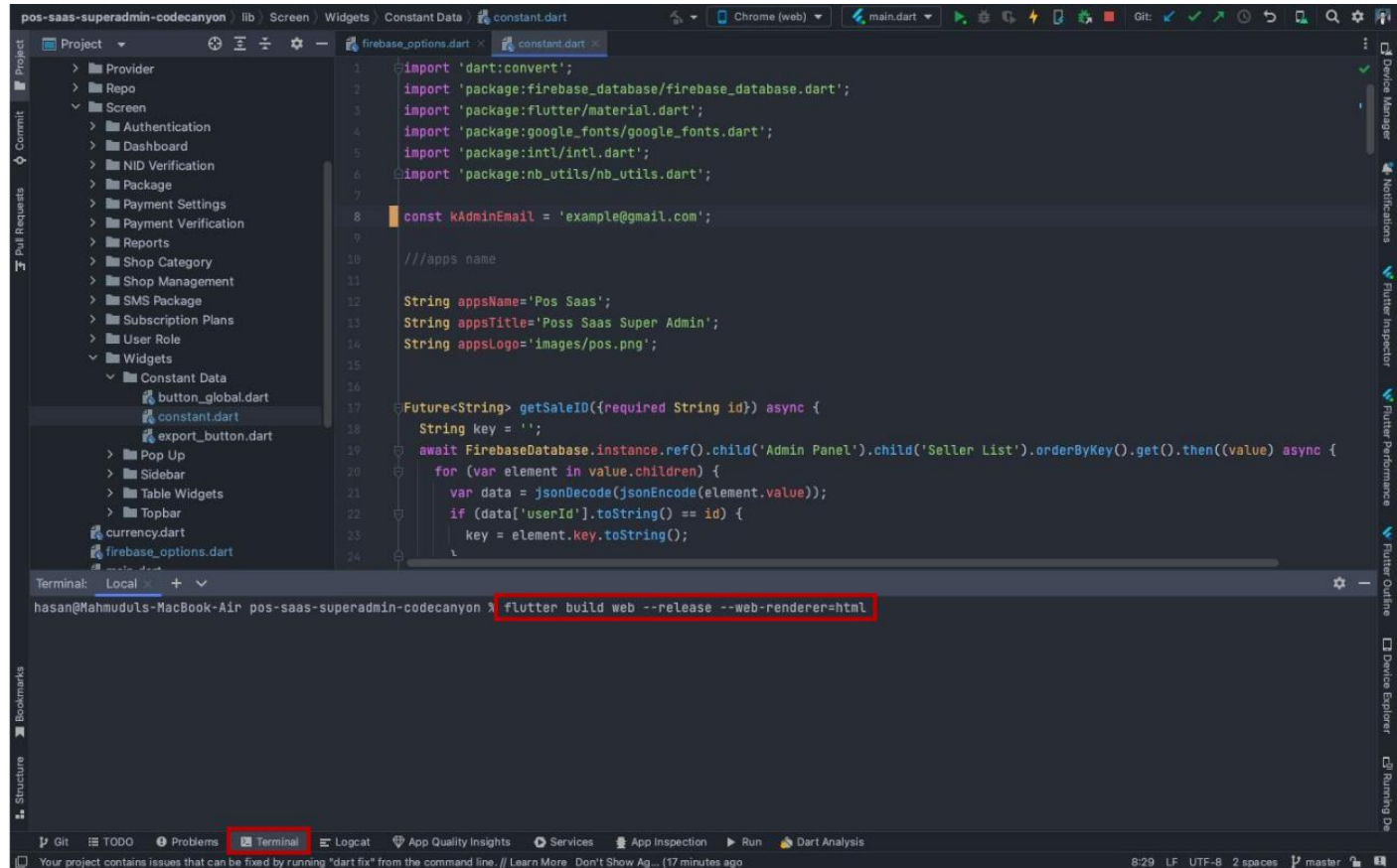
S.L	CATEGORY NAME	DESCRIPTION	CREATED BY	ACTION
1	Fashion	Fashion	Admin	⋮
2			Admin	⋮
3			Admin	⋮
4			Admin	⋮
5			Admin	⋮
6			Admin	⋮

Build File for Server Upload

To build the web application to upload the server, you have to open the terminal section. With android studio you can open it from the bottom bar of the environment. In terminal write the command

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

and press enter.

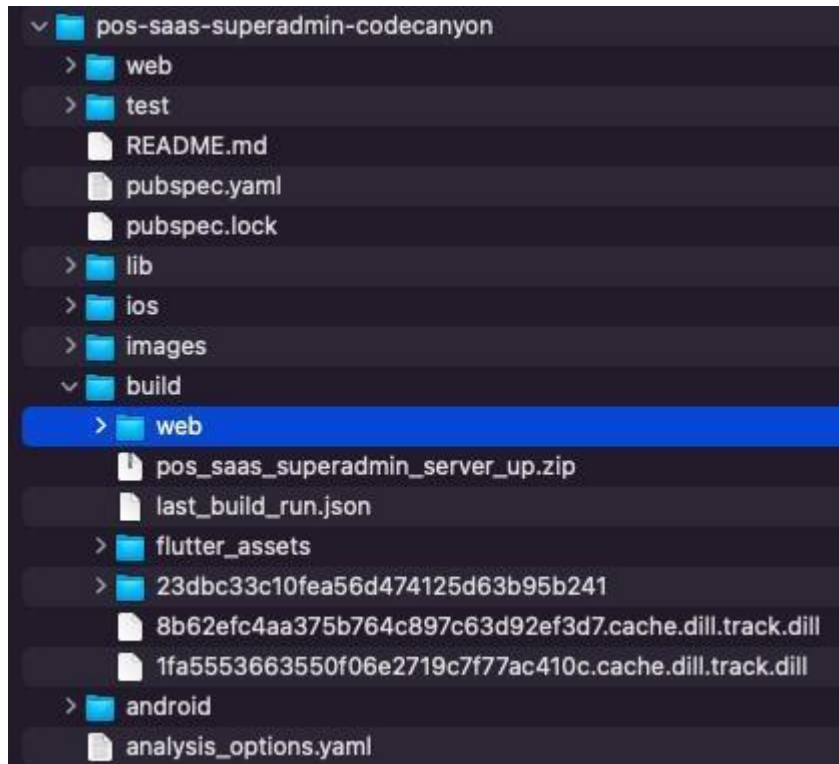


It will take some time to build the file. After build the app, you have to go to the directory where you store the file. And open the web folder which is inside the build folder.

Path: pos-saas-superadmin/build/web

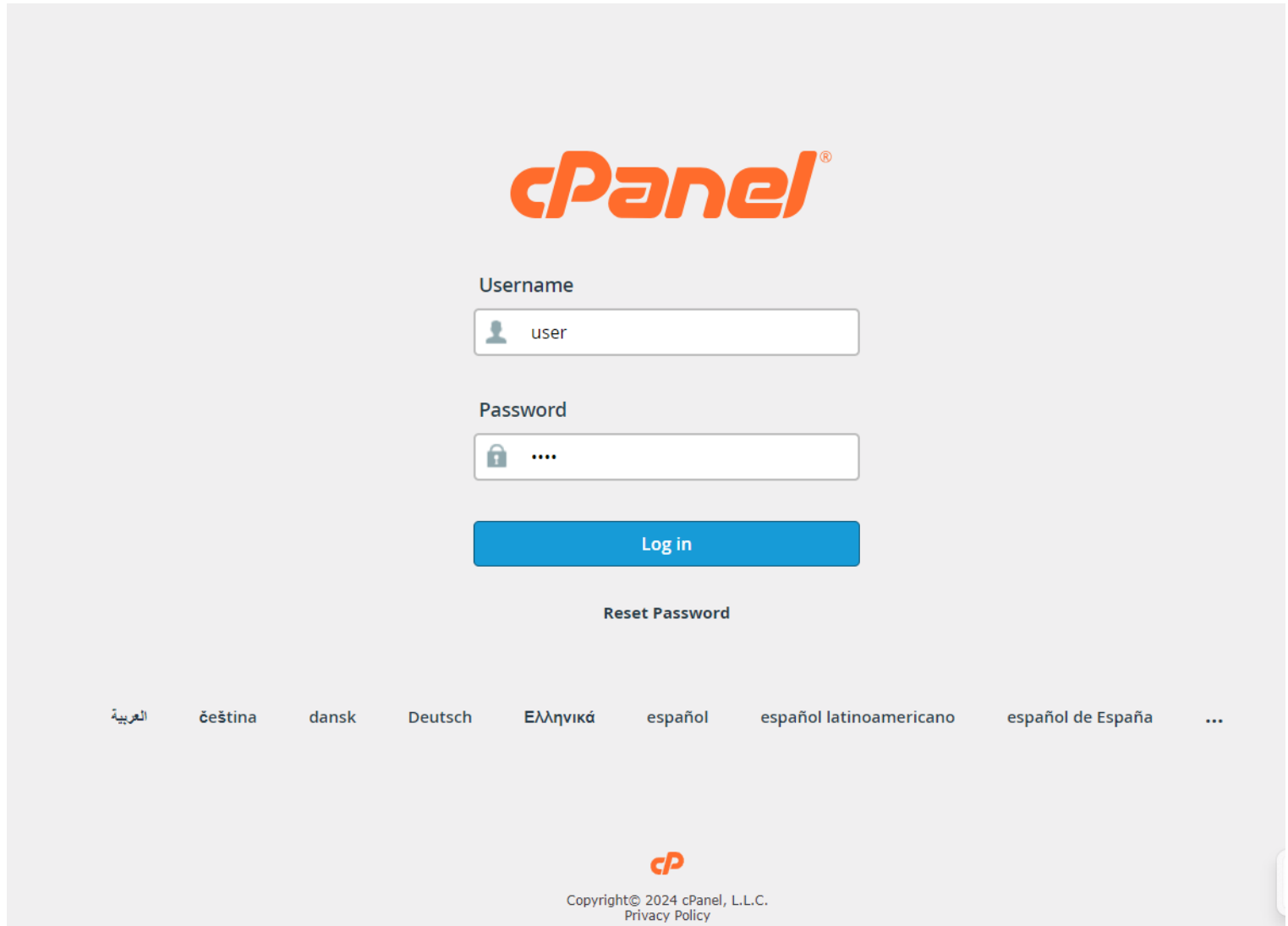
Now compress the web folder and extract it to the server

Make sure you select the web folder which is inside the build folder. Not outside the build folder.



Upload web file to server

Please browse the cPanel url from your laptop/pc and login with your credentials.



The image shows the cPanel login interface. At the top center is the cPanel logo in orange. Below it, there are two input fields: 'Username' with a user icon and the text 'user', and 'Password' with a lock icon and four dots. A blue 'Log in' button is positioned below the password field. Underneath the button is a link for 'Reset Password'. At the bottom, there is a horizontal list of language options: العربية, čeština, dansk, Deutsch, Ελληνικά, español, español latinoamericano, español de España, and an ellipsis. The footer contains the cPanel logo, the copyright notice 'Copyright© 2024 cPanel, L.L.C.', and a link to the 'Privacy Policy'.

Username

user

Password

....

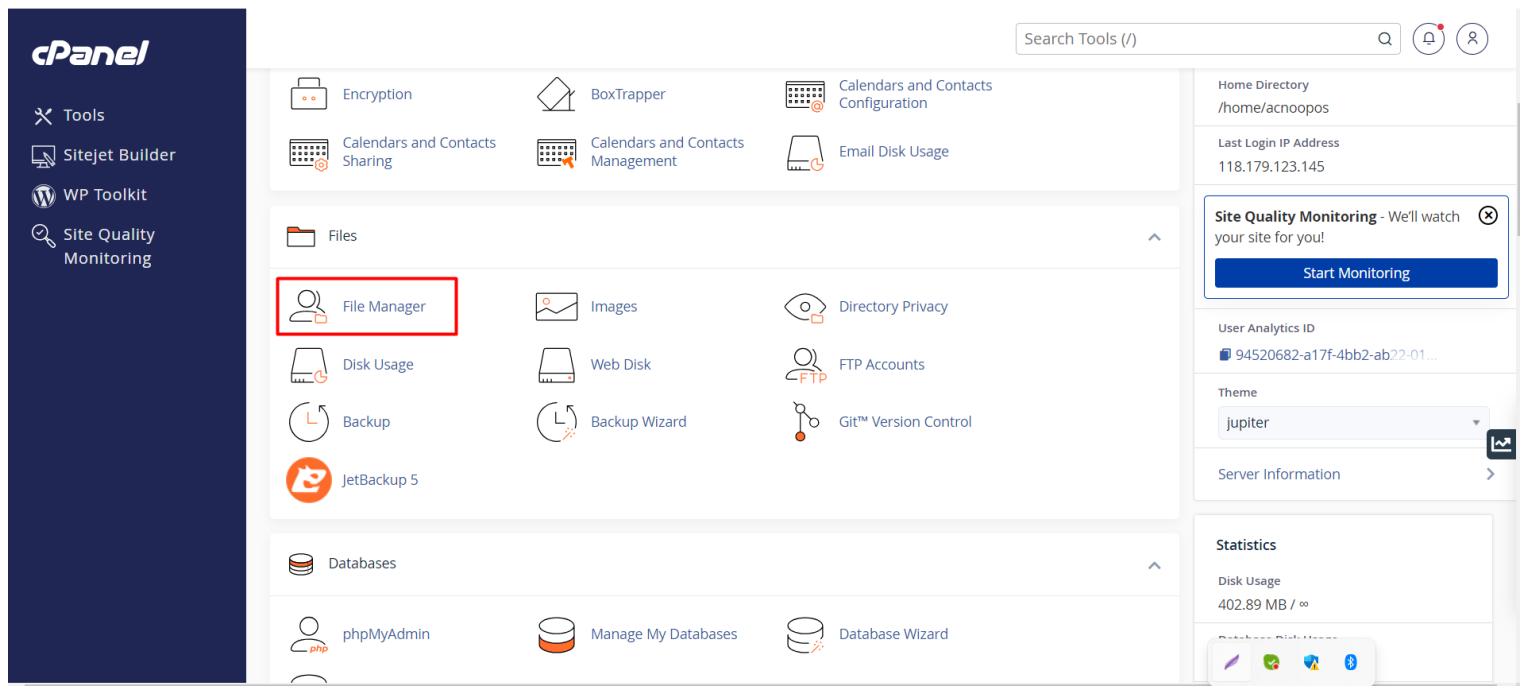
Log in

Reset Password

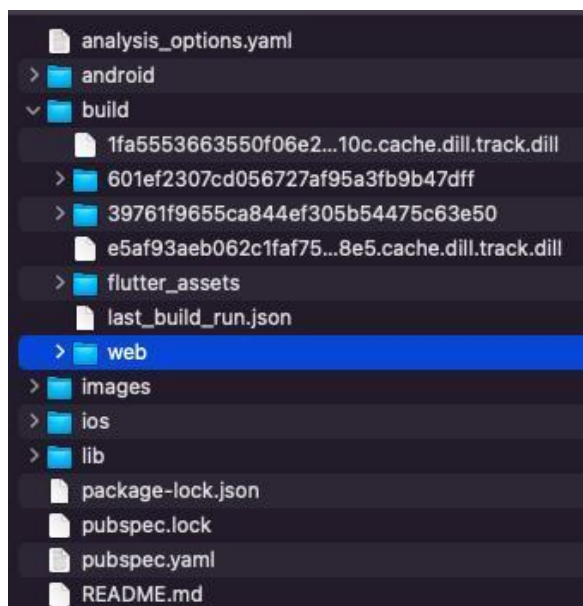
العربية čeština dansk Deutsch Ελληνικά español español latinoamericano español de España ...

cP
Copyright© 2024 cPanel, L.L.C.
Privacy Policy

A new screen will appear with various menu options. From there, select and open "File Manager."

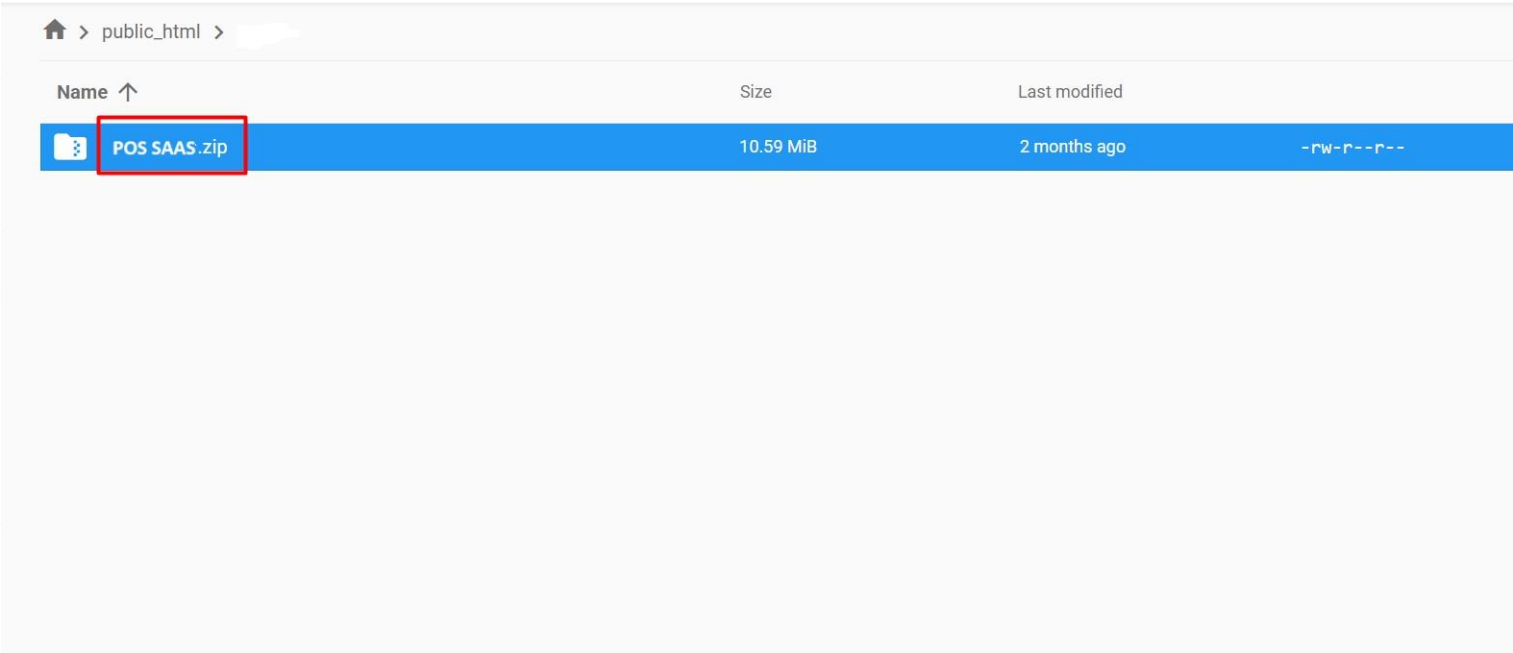


Now upload the server file that you built for your POS SaaS web app. You can find it in the build folder.

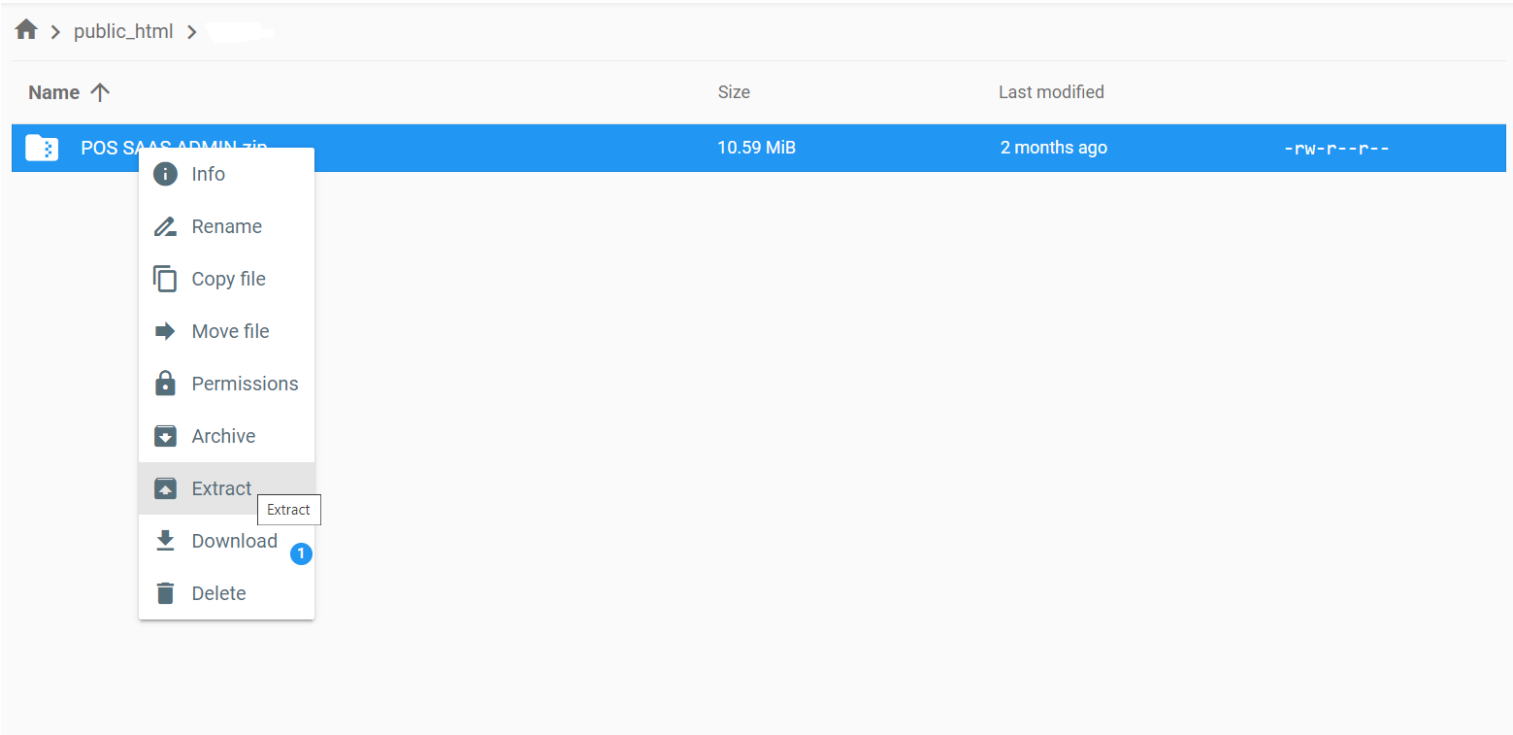


First, compress the web file you built, then proceed to upload it. You can move the file from the build folder to another location to make it easier to find. You can name the folder anything you like, but ensure it's easily identifiable for future reference.

After uploading the file, you will see it in your File Manager. Then, extract the file in this folder to make all the files visible.



To extract the file, place your mouse cursor over the file you uploaded, right-click on it, and select "Extract" from the menu to extract the contents.



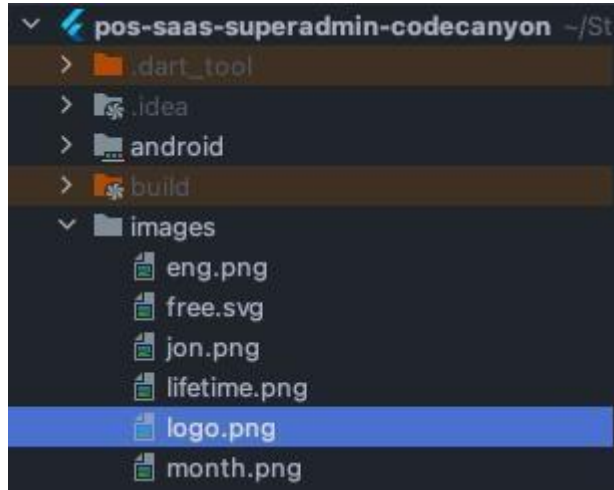
Now, you will be able to see the full contents of the files like this.

Name ↑	Size	Last modified	
 assets	—	a month ago	drwxr-xr-x
 canvaskit	—	a month ago	drwxr-xr-x
 icons	—	a month ago	drwxr-xr-x
 .htaccess	232 B	2 months ago	-rw-r--r--
 .last_build_id	32 B	a month ago	-rw-r--r--
 favicon.png	99.43 KiB	a month ago	-rw-r--r--
 flutter.js	7.6 KiB	a month ago	-rw-r--r--
 flutter_bootstrap.js	7.91 KiB	a month ago	-rw-r--r--
 flutter_service_worker.js	10.24 KiB	a month ago	-rw-r--r--
 index.html	1.95 KiB	a month ago	-rw-r--r--
 main.dart.js	3.51 MiB	a month ago	-rw-r--r--
 manifest.json	934 B	a month ago	-rw-r--r--
 version.json	108 B	a month ago	-rw-r--r--

Change the App logo

To change the logo in the easy way, you have to make a logo for your app, and rename it to **Logo.png**

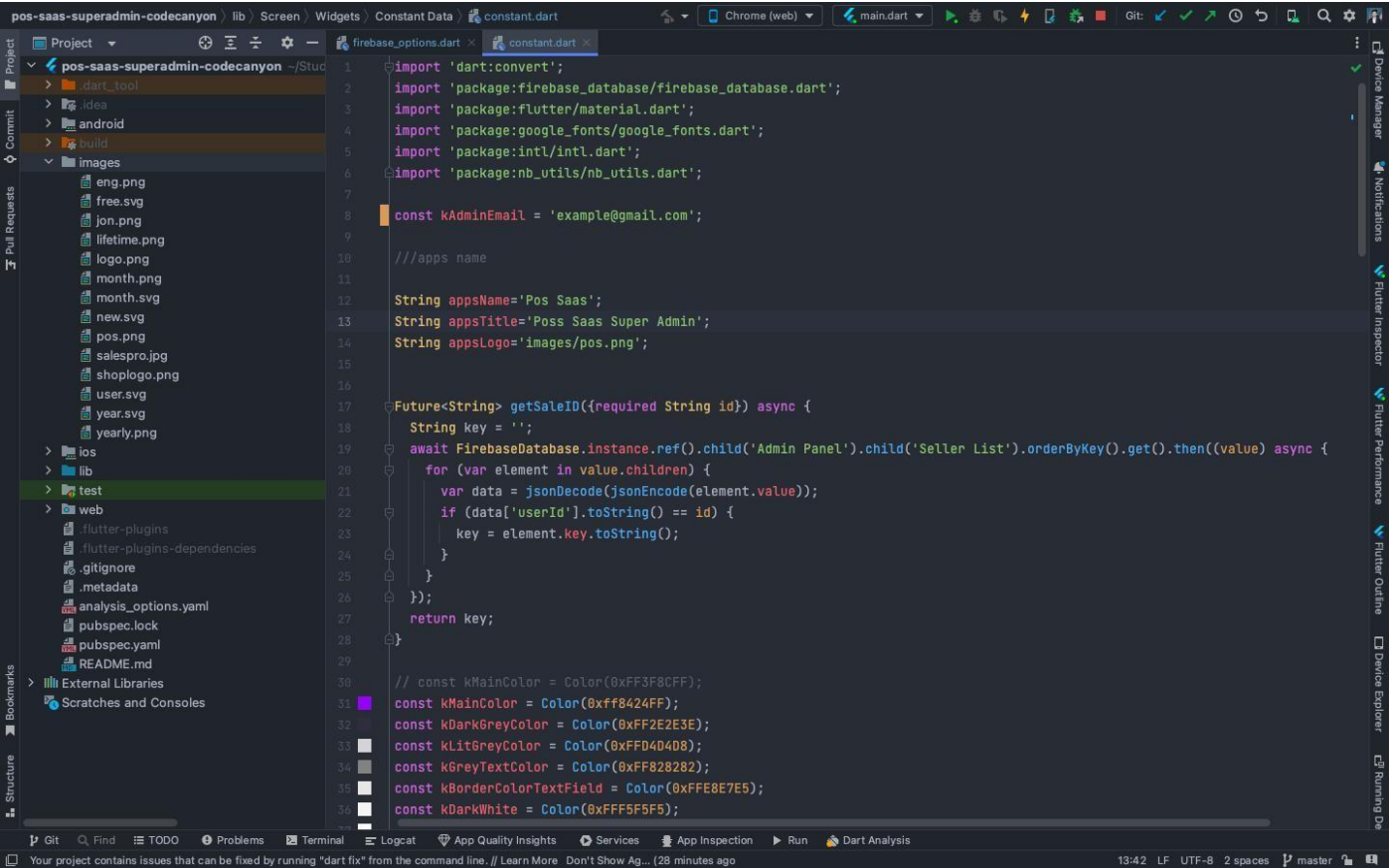
Now open the project, copy your logo from the local folder and paste it to the images directory in the project.



It will give you a warning to replace or override the image. You have to agree for that. And the image will replace the previous image.

App name change

To change the app name which uses inside the app, you have to go to the constant.dart file and change the name as you want.



```
1 import 'dart:convert';
2 import 'package:firebase_database/firebase_database.dart';
3 import 'package:flutter/material.dart';
4 import 'package:google_fonts/google_fonts.dart';
5 import 'package:intl/intl.dart';
6 import 'package:nb_utils/nb_utils.dart';
7
8 const kAdminEmail = 'example@gmail.com';
9
10 ///apps name
11
12 String appName='Pos Saas';
13 String appTitle='Pos Saas Super Admin';
14 String appLogo='images/pos.png';
15
16
17 Future<String> getSaleID({required String id}) async {
18   String key = '';
19   await FirebaseDatabase.instance.ref().child('Admin Panel').child('Seller List').orderByKey().get().then((value) async {
20     for (var element in value.children) {
21       var data = jsonDecode(jsonEncode(element.value));
22       if (data['userId'].toString() == id) {
23         key = element.key.toString();
24       }
25     }
26   });
27   return key;
28 }
29
30 // const kMainColor = Color(0xFF3F8CFF);
31 const kMainColor = Color(0xFF8424FF);
32 const kDarkGreyColor = Color(0xFF2E2E3E);
33 const kLitGreyColor = Color(0xFFD4D4D8);
34 const kGreyTextColor = Color(0xFF828282);
35 const kBorderColorTextField = Color(0xFFE8E7E5);
36 const kDarkWhite = Color(0xFFFF5F5F5F);
```

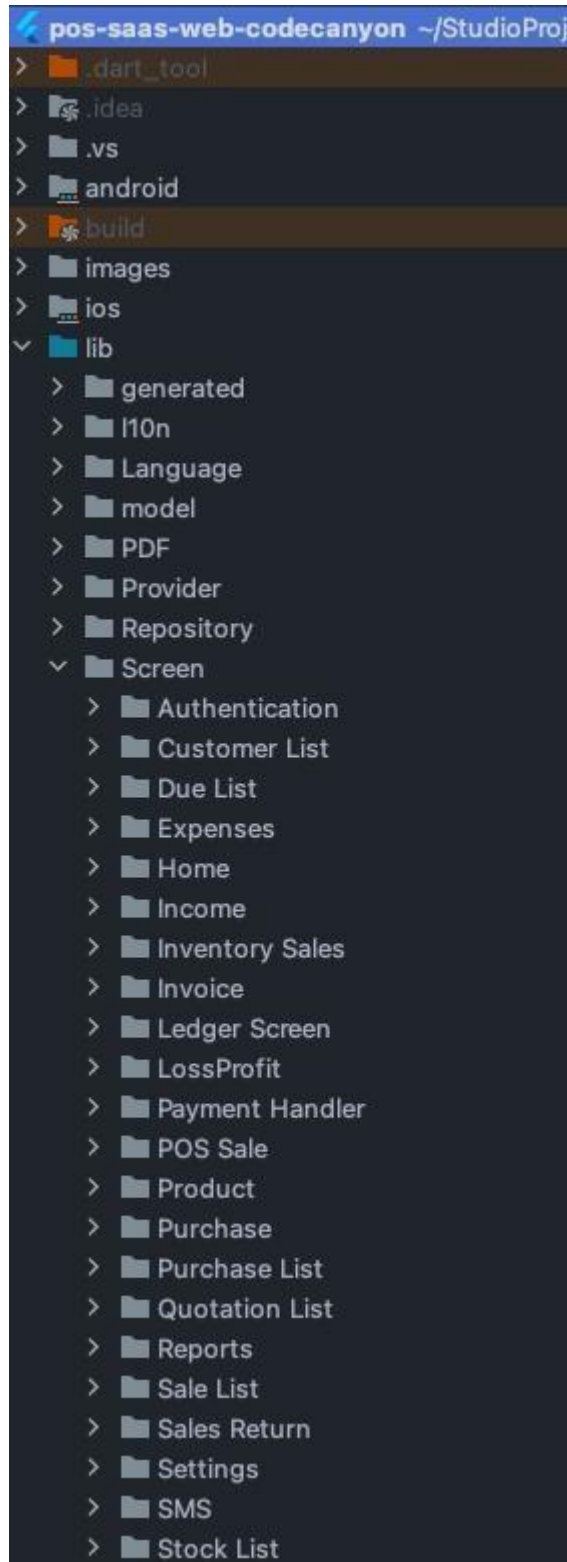
After doing all the tasks successfully, without any error then build the file for server upload.

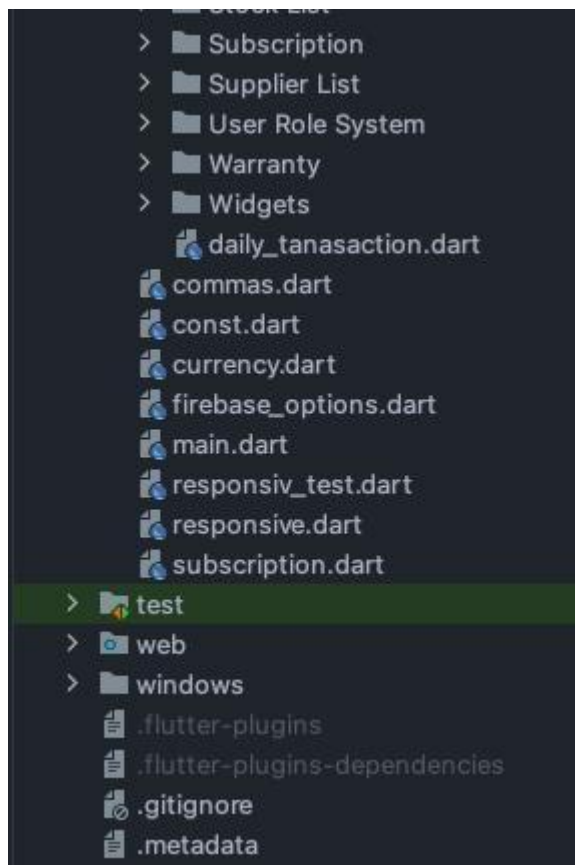
End of the Super Admin setting.

POS Saas Web

Open and run the project on you environment. You can follow the previous instructions to open.

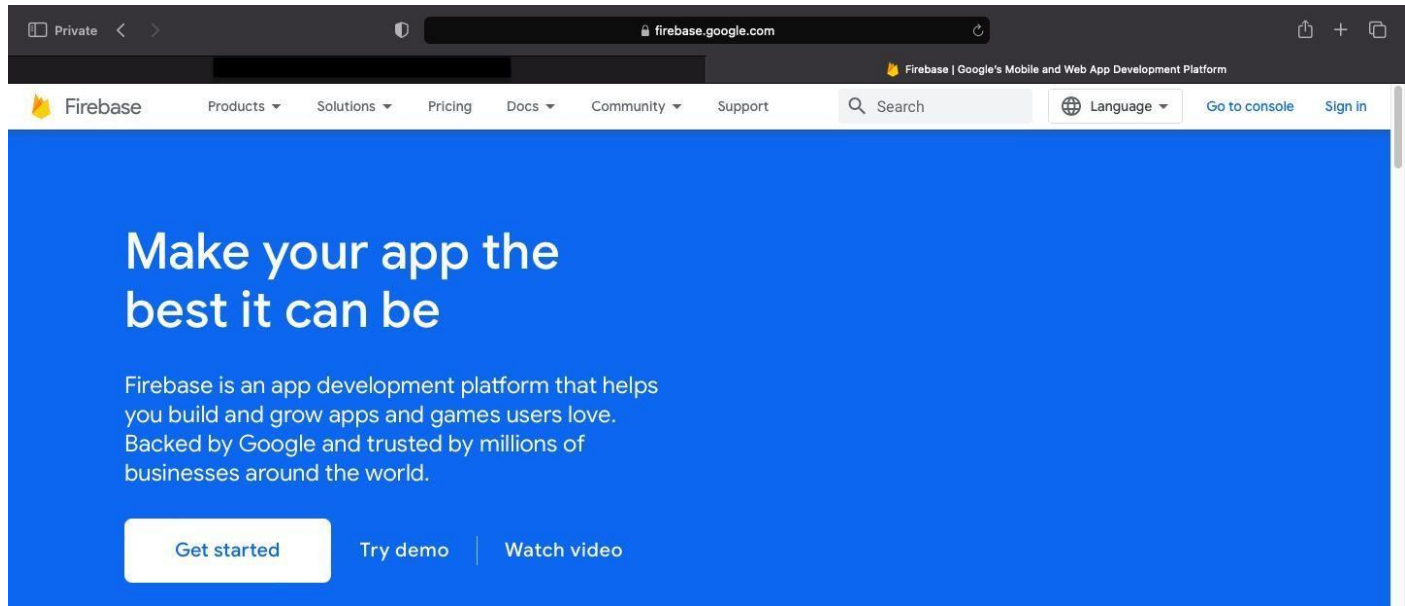
Project structure:





- **Images:** Apps/widgets/snippets images as well as applcon are stored here.
- **Lib:** Applications main files and directories are located here.
 - **Screen:** Full app's UI codes are stored here.
 - **Widgets:** All the reusable widgets are stored here, like(Top bar, Side bar).Also the constant file located in the folder. Where the all the colors are stored which is used to design the app.
 - In **firebase_options.dart** all the code stored which is related with the firebase.From here you can easily change the firebase credentials.
 - **main.dart:** All the routes have been declared here.

Now open the URL <https://console.firebase.google.com/> and click on Go to the console which is on the right side of the top bar menu.

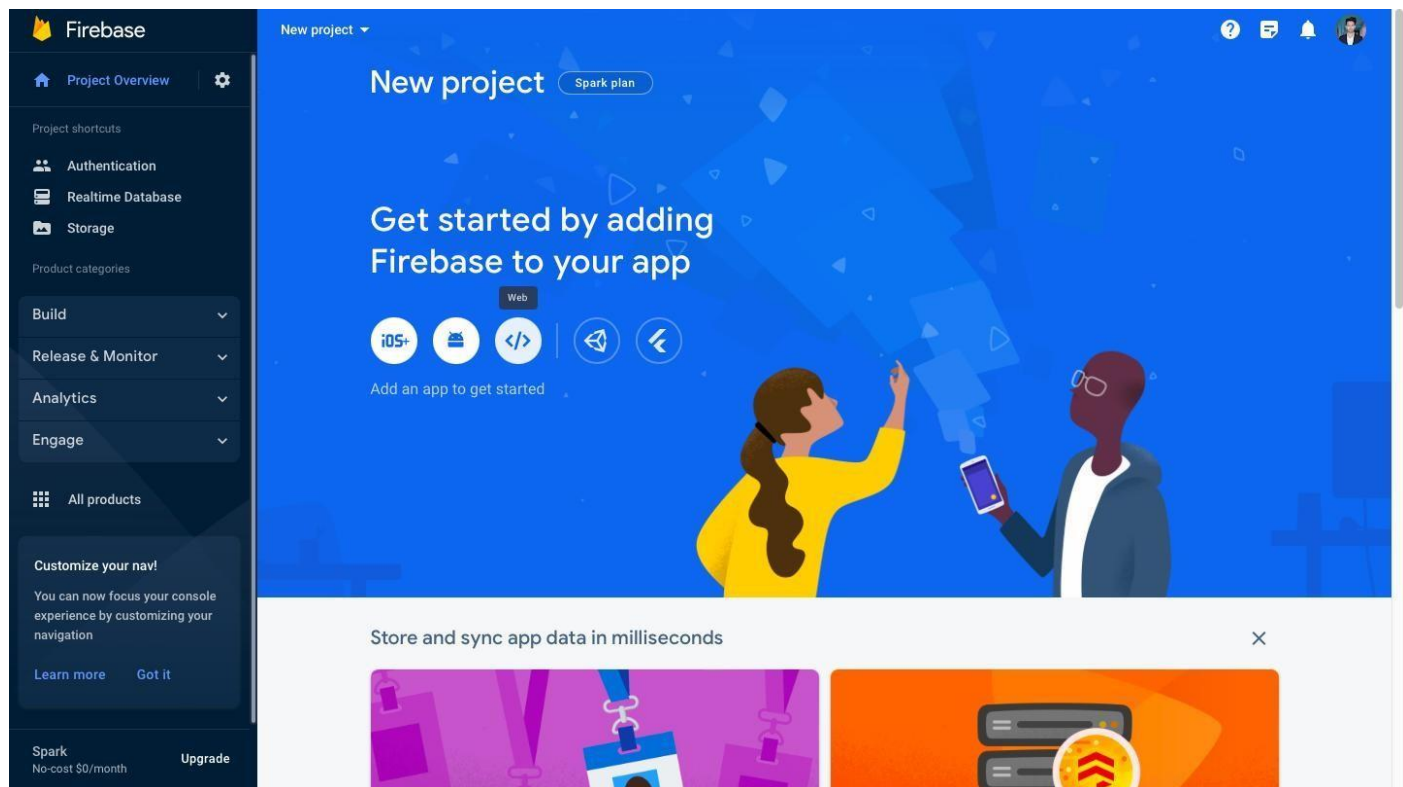


And choose Your created project, which you created for the **POS Saas Super Admin** & open it.

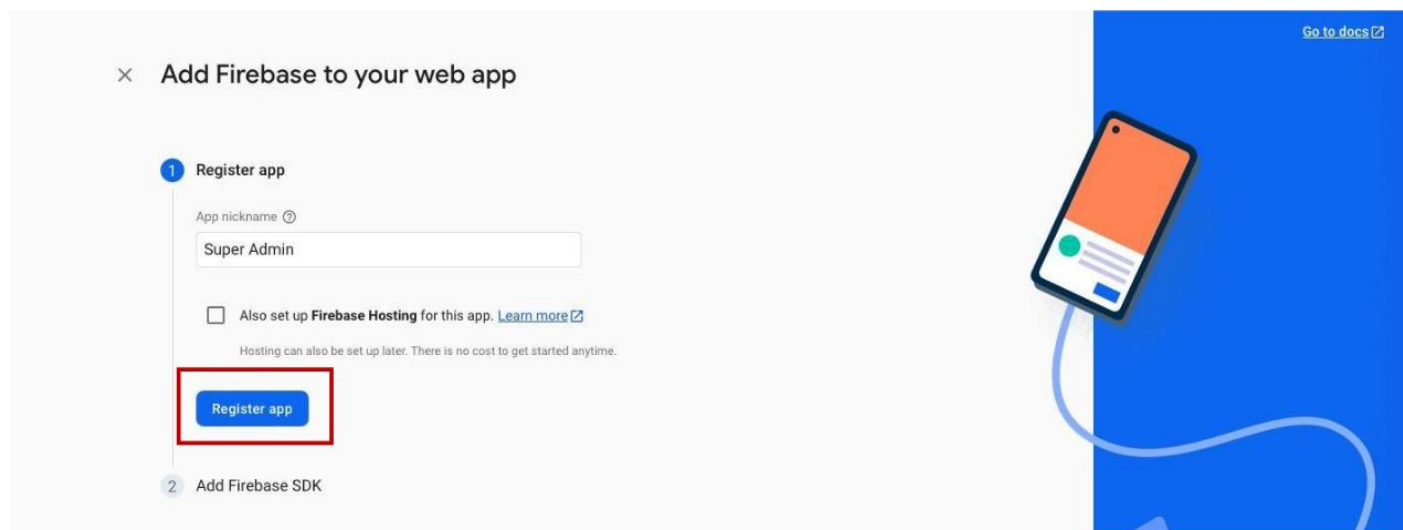


Step 1

Press on the **Project overview** section and press on the **web** button.



Now write a name for your web app. No need to check the firebase hosting for this project and press on the **Register app** button to go to the next step.



Then it will generate the all credentials for the app which you have to copy and store in a safe place. Don't forget to copy the credentials.

Copy all the codes which is inside the firebaseConfig (Which credentials generate for your project) & press the Continue to console button. For example you can see the screen below.

2 Add Firebase SDK

☒ Use npm ☐ Use a <script> tag

If you're already using [npm](#) and a module bundler such as [webpack](#) or [Rollup](#), you can run the following command to install the latest SDK ([Learn more](#)):

```
$ npm install firebase
```

Then, initialize Firebase and begin using the SDKs for the products you'd like to use.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyBeeoDKzwPZVQttmQ1NldwHuMnuLpYgf0I",
  authDomain: "new-project-c0477.firebaseio.com",
  databaseURL: "https://new-project-c0477-default-rtdb.firebaseio.com",
  projectId: "new-project-c0477",
  storageBucket: "new-project-c0477.appspot.com",
  messagingSenderId: "348421186272",
  appId: "1:348421186272:web:2e2db30822b0269834e685"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
```

Note: This option uses the [modular JavaScript SDK](#), which provides reduced SDK size.

Learn more about Firebase for web: [Get Started](#), [Web SDK API Reference](#), [Samples](#)

Continue to console

Go to docs



**This is for an example not use
this code on your project**

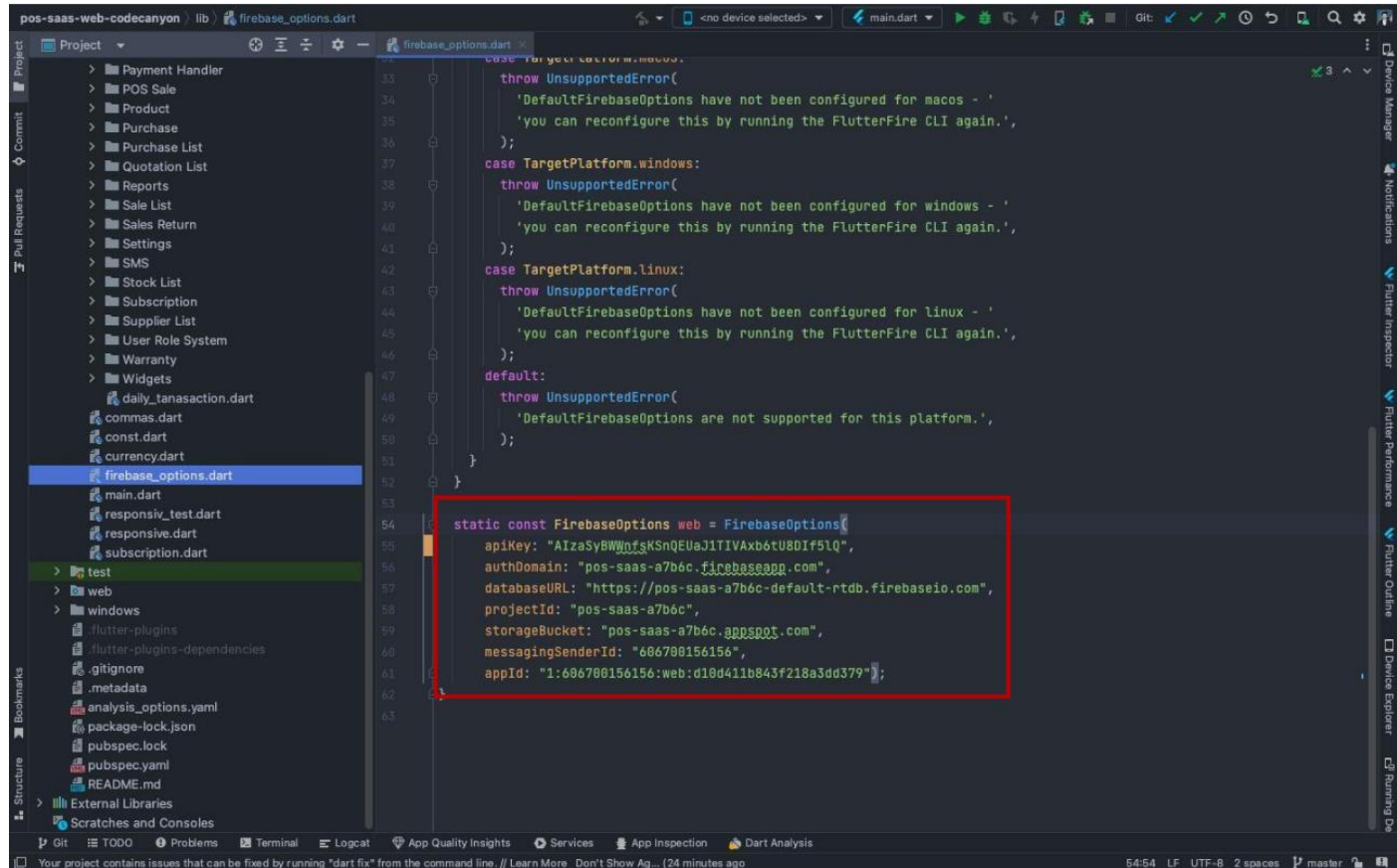
Now open the dart file named “**firebase_options.dart**”.

Location: Project/lib

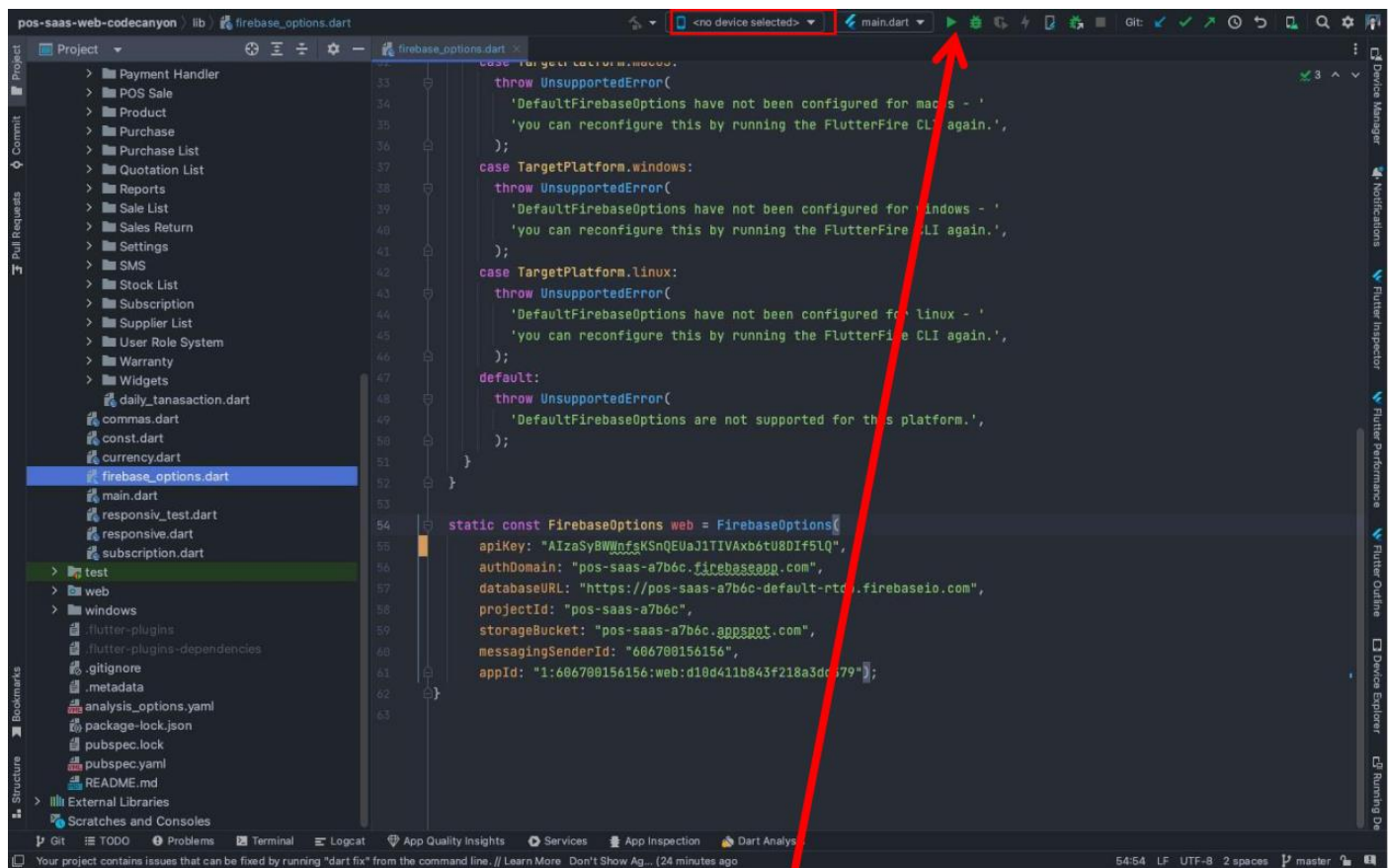
(You can also see the project structure to find it easily)

After open the **firebase_option.dart** file, come to the last section of the code, marked with the red color and paste all the credentials.(api key, authDomain, databaseURL, projectId, storageBucket, messagingSenderId,appId) which you copied in the last section.

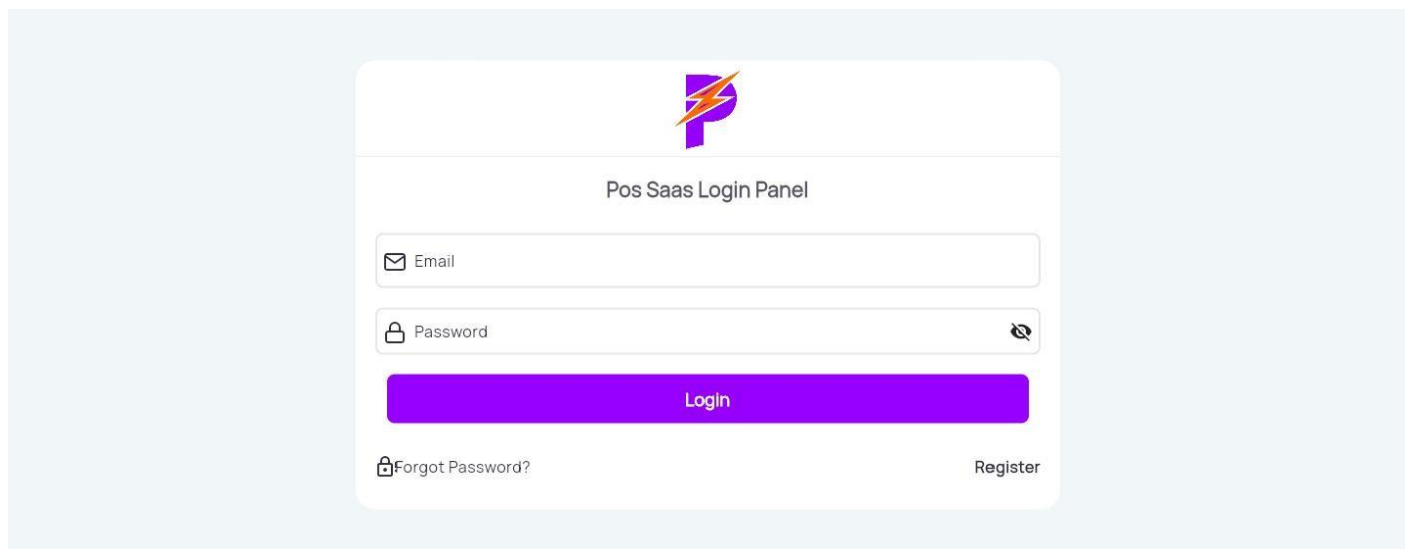
Make sure that you have pasted the correct code.



```
33 case TargetPlatform.macos:
34   throw UnsupportedError(
35     'DefaultFirebaseOptions have not been configured for macos - '
36     'you can reconfigure this by running the FlutterFire CLI again.',
37   );
38 case TargetPlatform.windows:
39   throw UnsupportedError(
40     'DefaultFirebaseOptions have not been configured for windows - '
41     'you can reconfigure this by running the FlutterFire CLI again.',
42   );
43 case TargetPlatform.linux:
44   throw UnsupportedError(
45     'DefaultFirebaseOptions have not been configured for linux - '
46     'you can reconfigure this by running the FlutterFire CLI again.',
47   );
48 default:
49   throw UnsupportedError(
50     'DefaultFirebaseOptions are not supported for this platform.',
51   );
52 }
53
54 static const FirebaseOptions web = FirebaseOptions(
55   apiKey: "AIzaSyBWWnf5KSnQE0aJITIVAx6tU8DIr5LQ",
56   authDomain: "pos-saas-a7b6c.firebaseio.com",
57   databaseURL: "https://pos-saas-a7b6c-default-rtdb.firebaseio.com",
58   projectId: "pos-saas-a7b6c",
59   storageBucket: "pos-saas-a7b6c.appspot.com",
60   messagingSenderId: "686788156156",
61   appId: "1:686788156156:web:d18d411b843f218a3dd379";
62
63
```

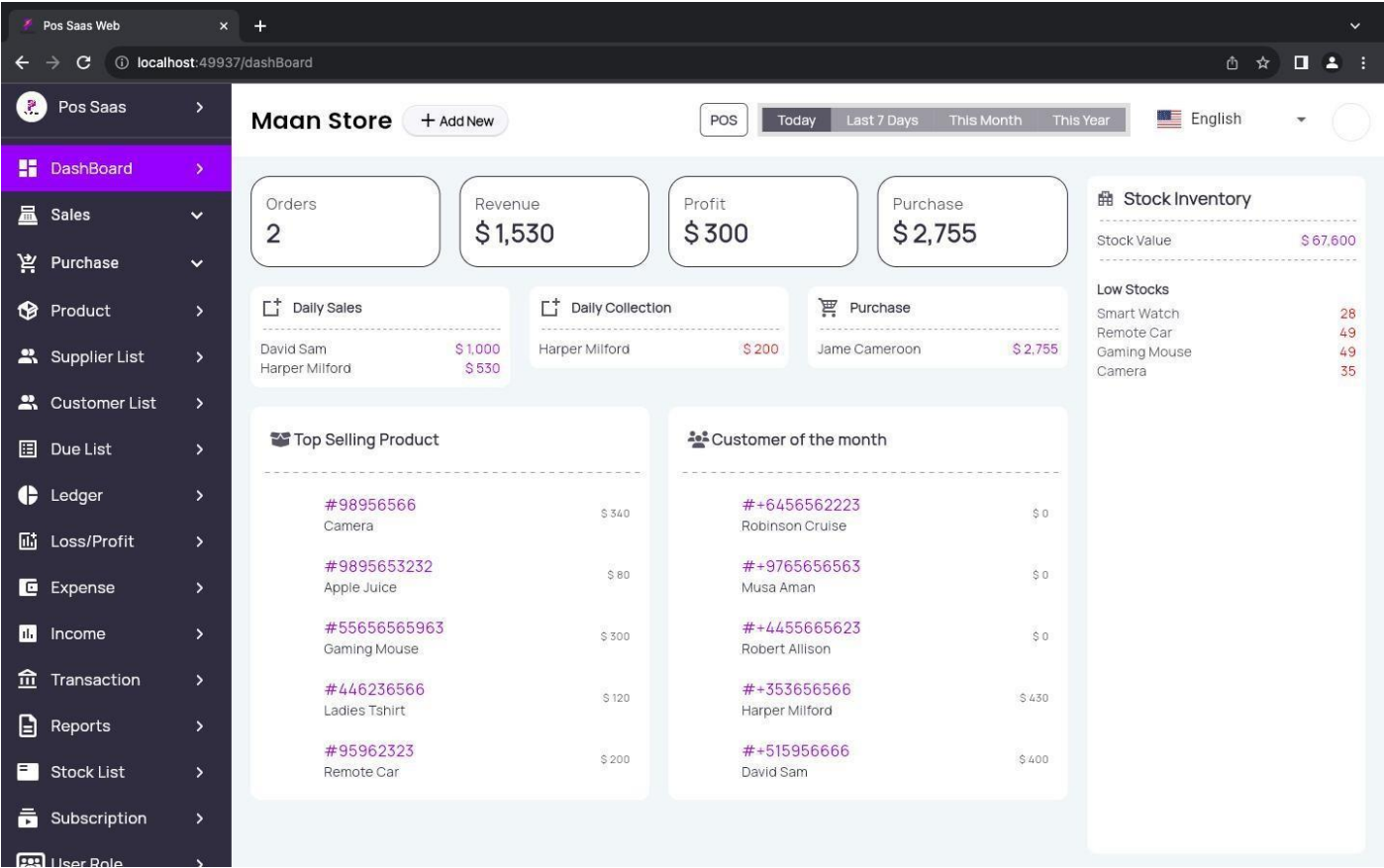


Then you have to run the app. From the topbar of your environment select a platform where you want to run the application. And press on the **Run** button



After opening the app, register an account. If the registration is complete successfully, it will open the web dashboard.

This is the dashboard screen of the POS SAAS web.



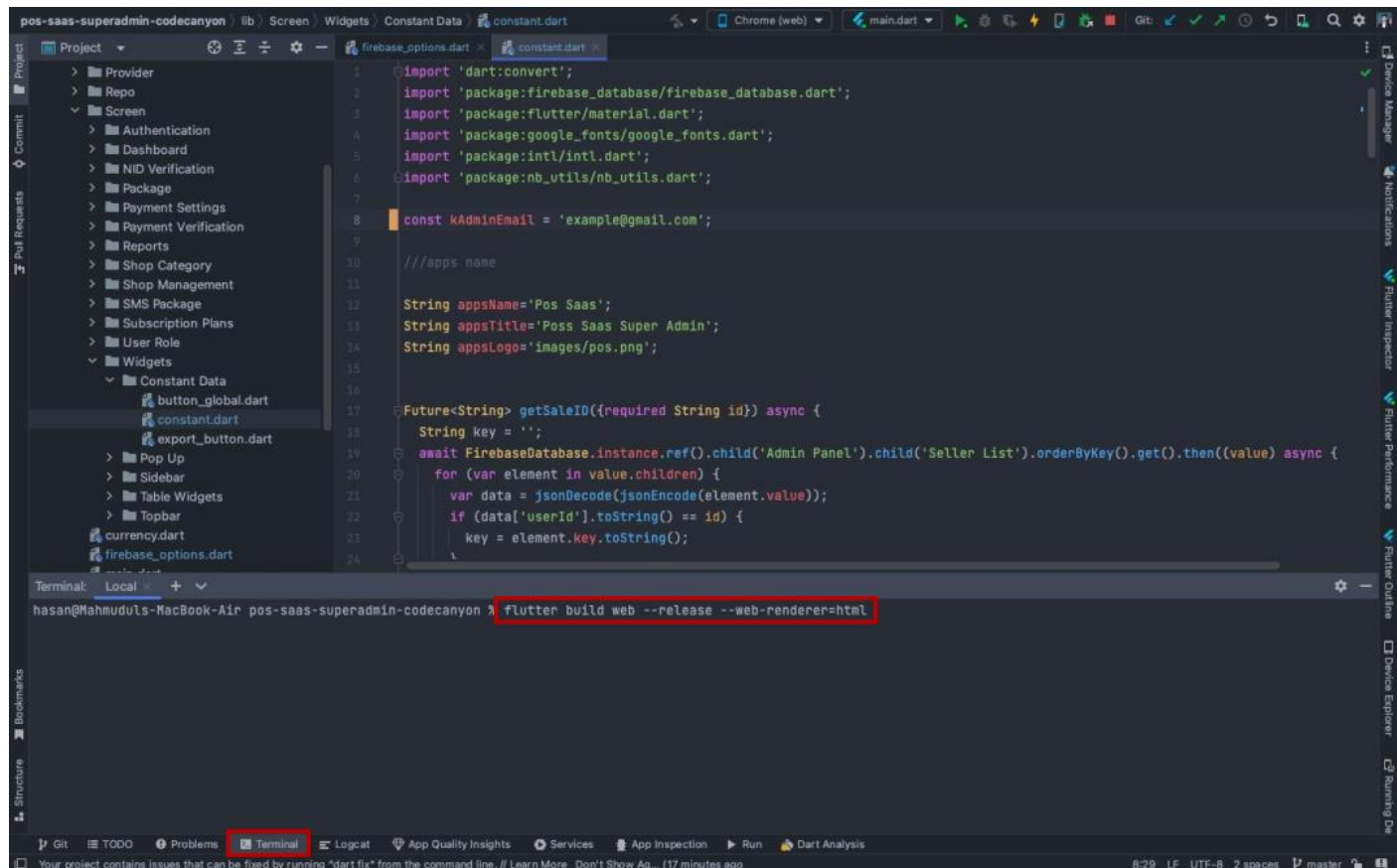
To make it live you have to build the web html file & upload it to the server.

Build File For Server Upload

To build the web application to upload the server, you have to open the terminal section. With android studio you can open it from the bottom bar of the environment. In terminal write the command

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

and press “Enter”.

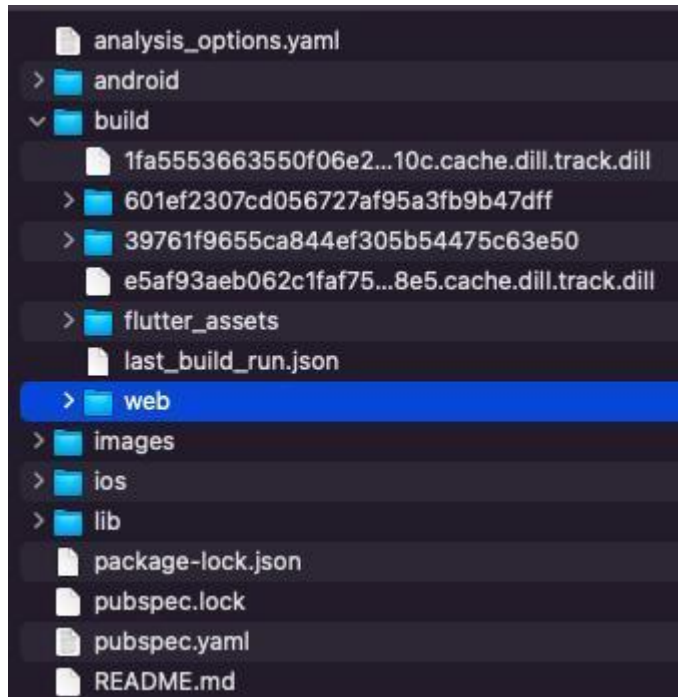


It will take some time to build the file. After build the app, you have to go to the directory where you store the file. And open the web folder which is inside the build folder.

Path: salespro-saas-super-admin/build/web

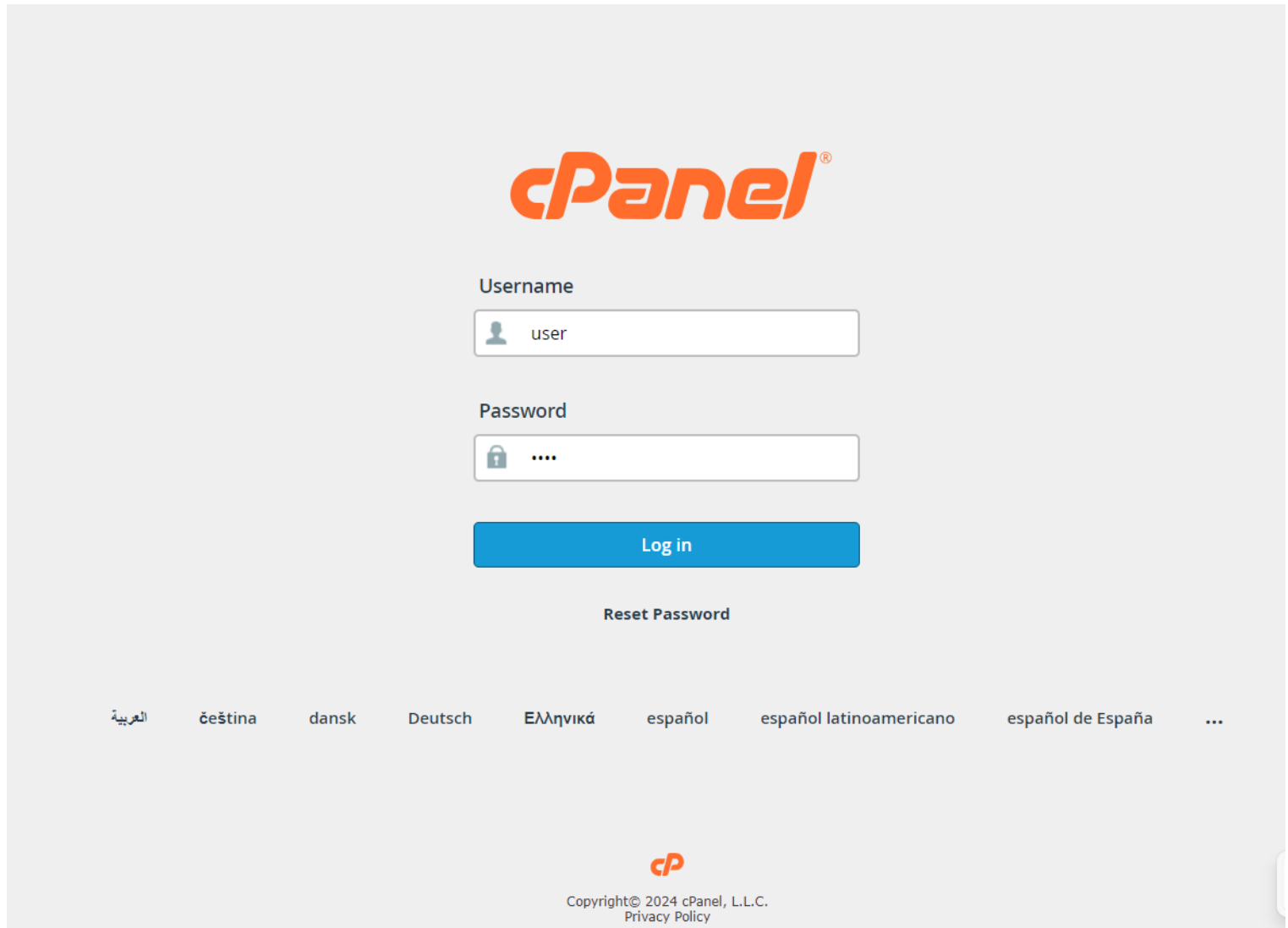
Now compress the web folder and extract it to the server.

Make sure you select the web folder which is inside the build folder. Not outside the build folder.

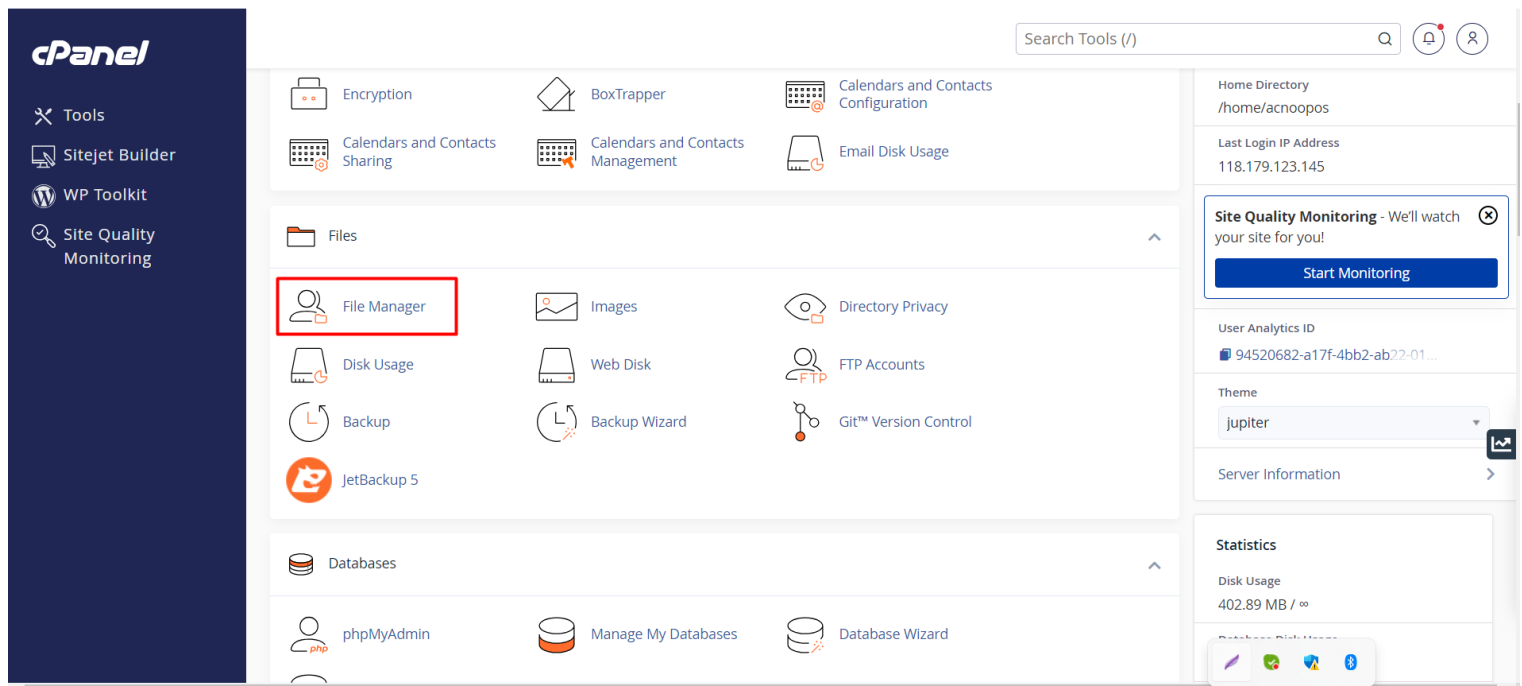


Upload admin web file to server

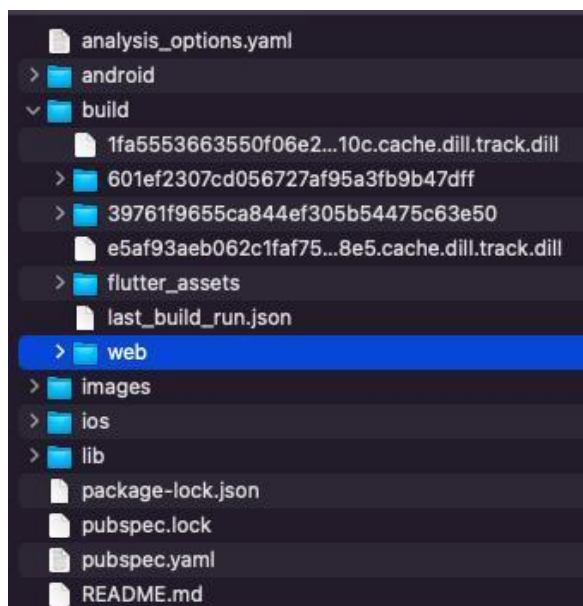
Open the cPanel from your laptop/pc and login with your credentials.

The image shows the cPanel login interface. At the top center is the cPanel logo in orange. Below it, there are two input fields: 'Username' with a user icon and the text 'user', and 'Password' with a lock icon and four dots. A blue 'Log in' button is positioned below the password field. Underneath the button is a link for 'Reset Password'. At the bottom, there is a horizontal list of language options: العربية, čeština, dansk, Deutsch, Ελληνικά, español, español latinoamericano, español de España, and an ellipsis. The footer contains the cPanel logo and the text 'Copyright© 2024 cPanel, L.L.C. Privacy Policy'.

A new screen will appear with various menu options. From there, select and open **"File Manager."**

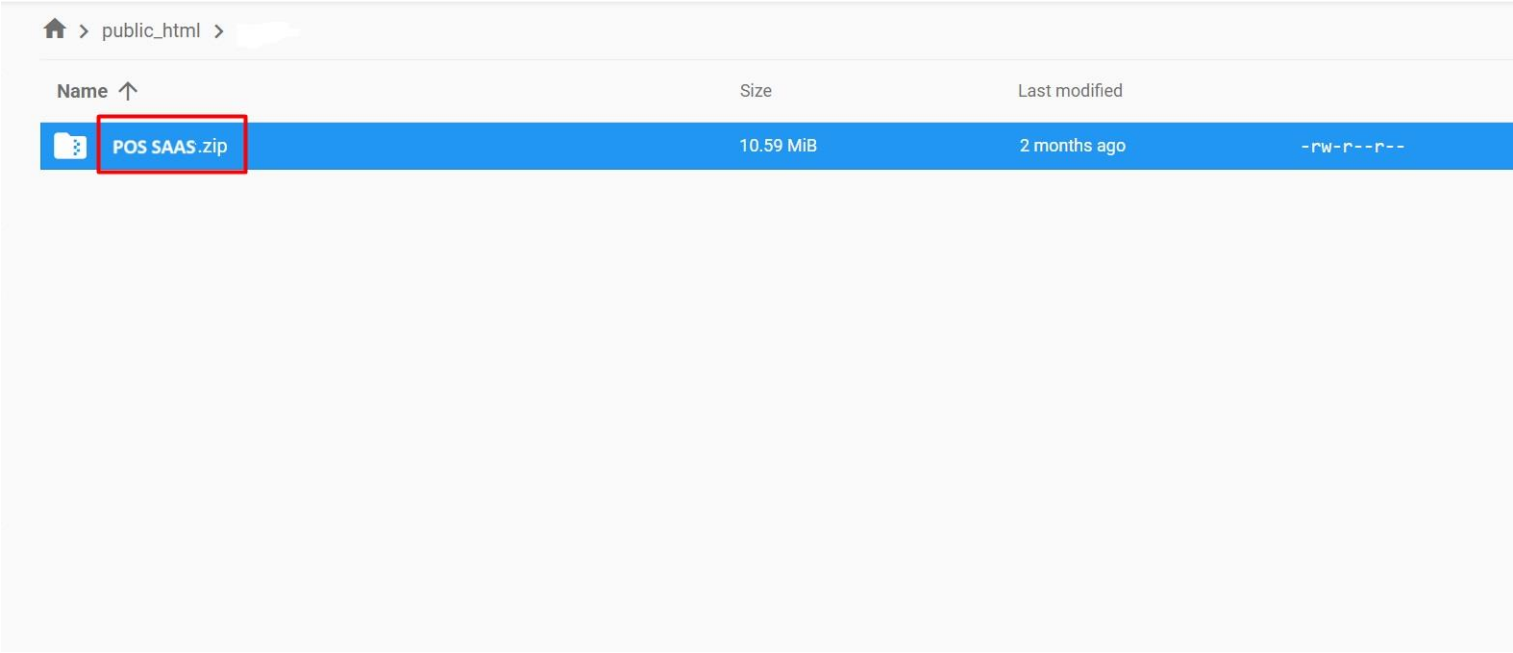


Now upload the server file that you built for your POS SaaS web app. You can find it in the build folder.

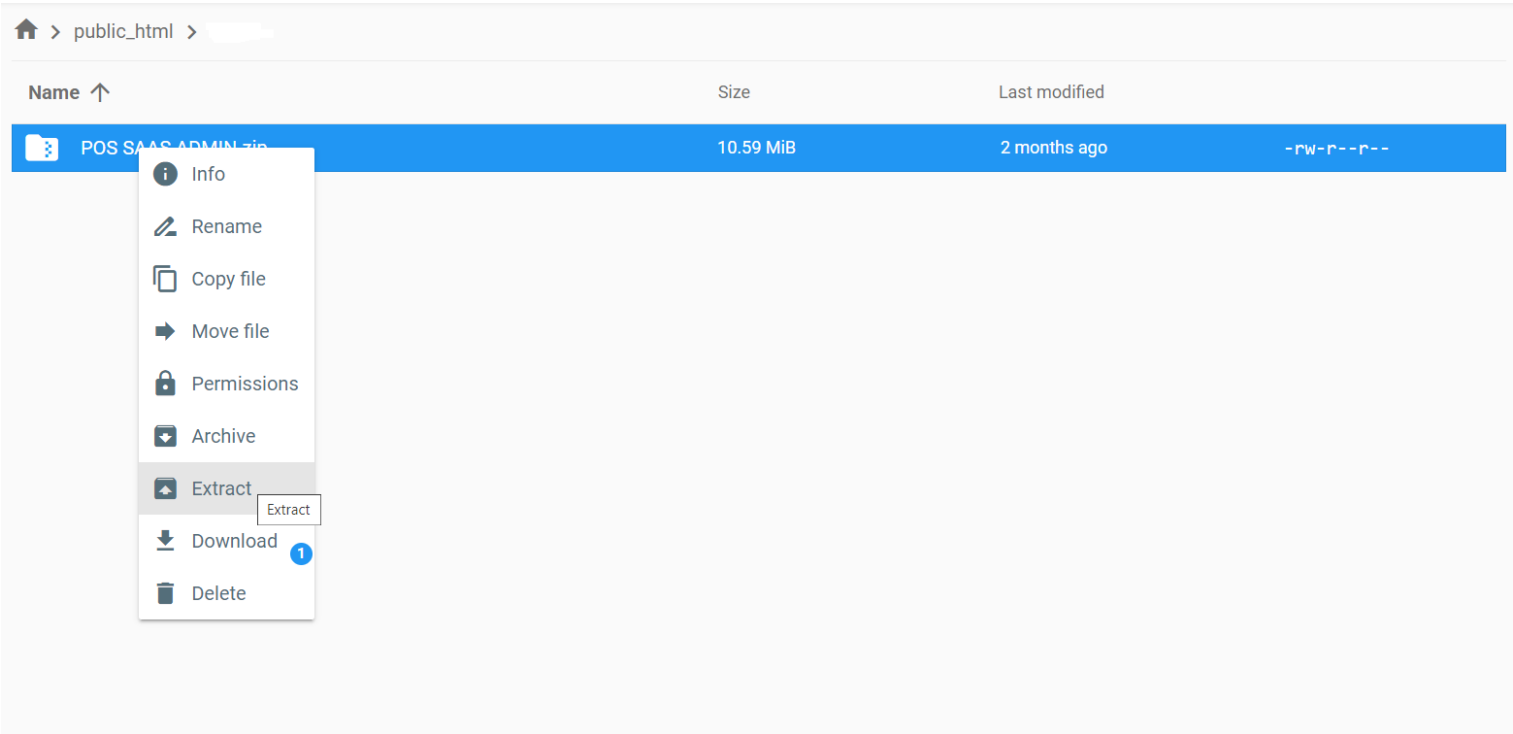


First, compress the web file you built, then proceed to upload it. You can move the file from the build folder to another location to make it easier to find. You can name the folder anything you like, but ensure it's easily identifiable for future reference.

After uploading the file, you will see it in your File Manager. Then, extract the file in this folder to make all the files visible.



To extract the file, place your mouse cursor over the file you uploaded, right-click on it, and select "Extract" from the menu to extract the contents.



Now, you will be able to see the full contents of the files like this.

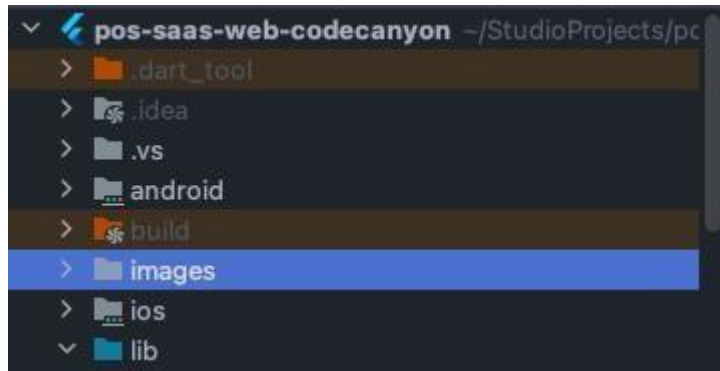
Name ↑		Size	Last modified	
	assets	—	a month ago	drwxr-xr-x
	canvaskit	—	a month ago	drwxr-xr-x
	icons	—	a month ago	drwxr-xr-x
	.htaccess	232 B	2 months ago	-rw-r--r--
	.last_build_id	32 B	a month ago	-rw-r--r--
	favicon.png	99.43 KiB	a month ago	-rw-r--r--
	flutter.js	7.6 KiB	a month ago	-rw-r--r--
	flutter_bootstrap.js	7.91 KiB	a month ago	-rw-r--r--
	flutter_service_worker.js	10.24 KiB	a month ago	-rw-r--r--
	index.html	1.95 KiB	a month ago	-rw-r--r--
	main.dart.js	3.51 MiB	a month ago	-rw-r--r--
	manifest.json	934 B	a month ago	-rw-r--r--
	version.json	108 B	a month ago	-rw-r--r--

Change the App logo

To change the logo in the easy way, you have to make a logo for your app, and rename it to

Pos.png

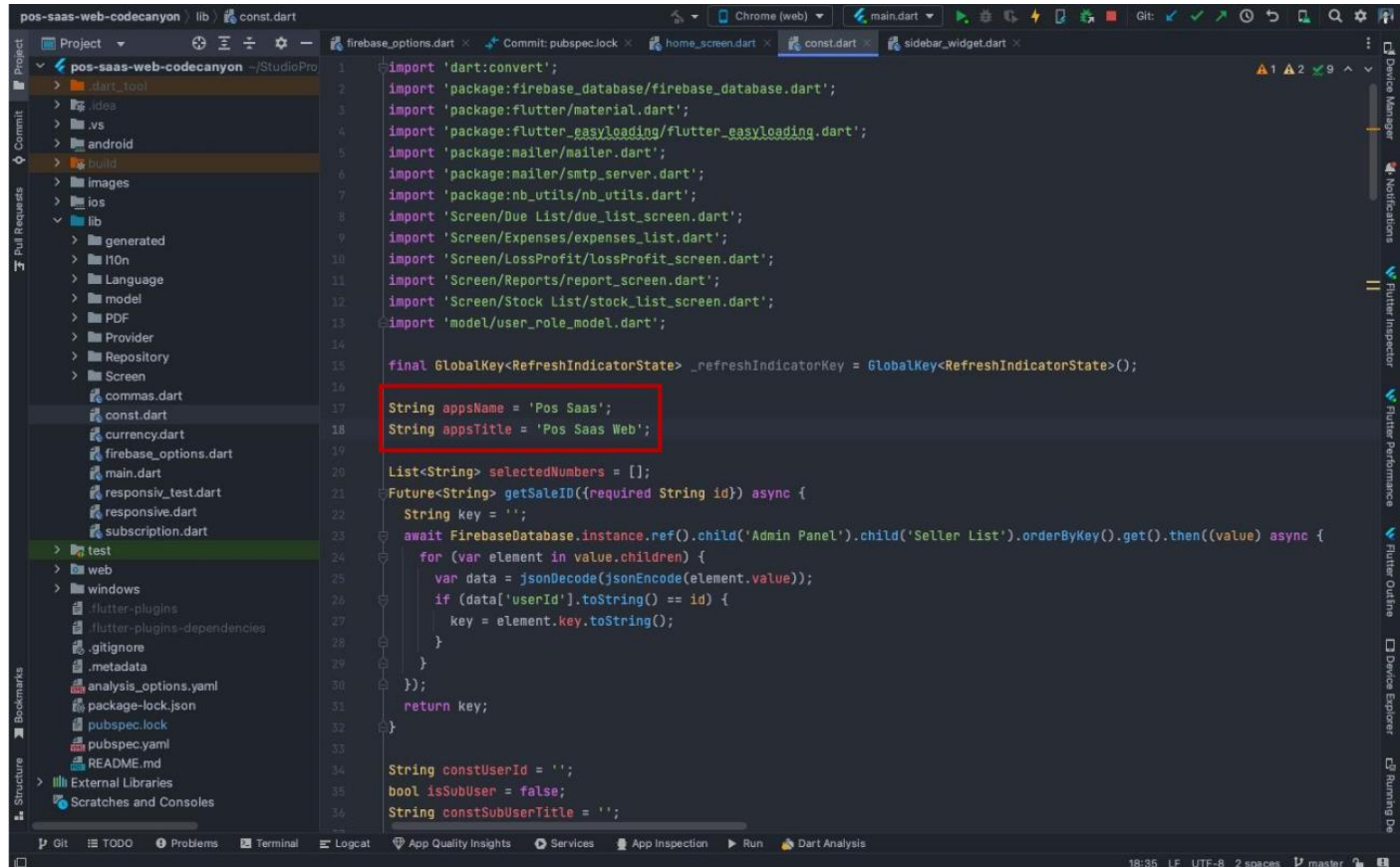
Now open the project, copy your logo from the local folder and paste it to the images directory in the project.



It will give you a warning to replace or override the image. You have to agree for that. And the image will replace the previous image.

App name change

To change the app name which uses inside the app, you have to go to the `const.dart` file and change the name as you want.



The screenshot shows an IDE with the `const.dart` file open. The file contains various imports and constants. The following lines are highlighted with a red box:

```
String appName = 'Pos Saas';
String appTitle = 'Pos Saas Web';
```

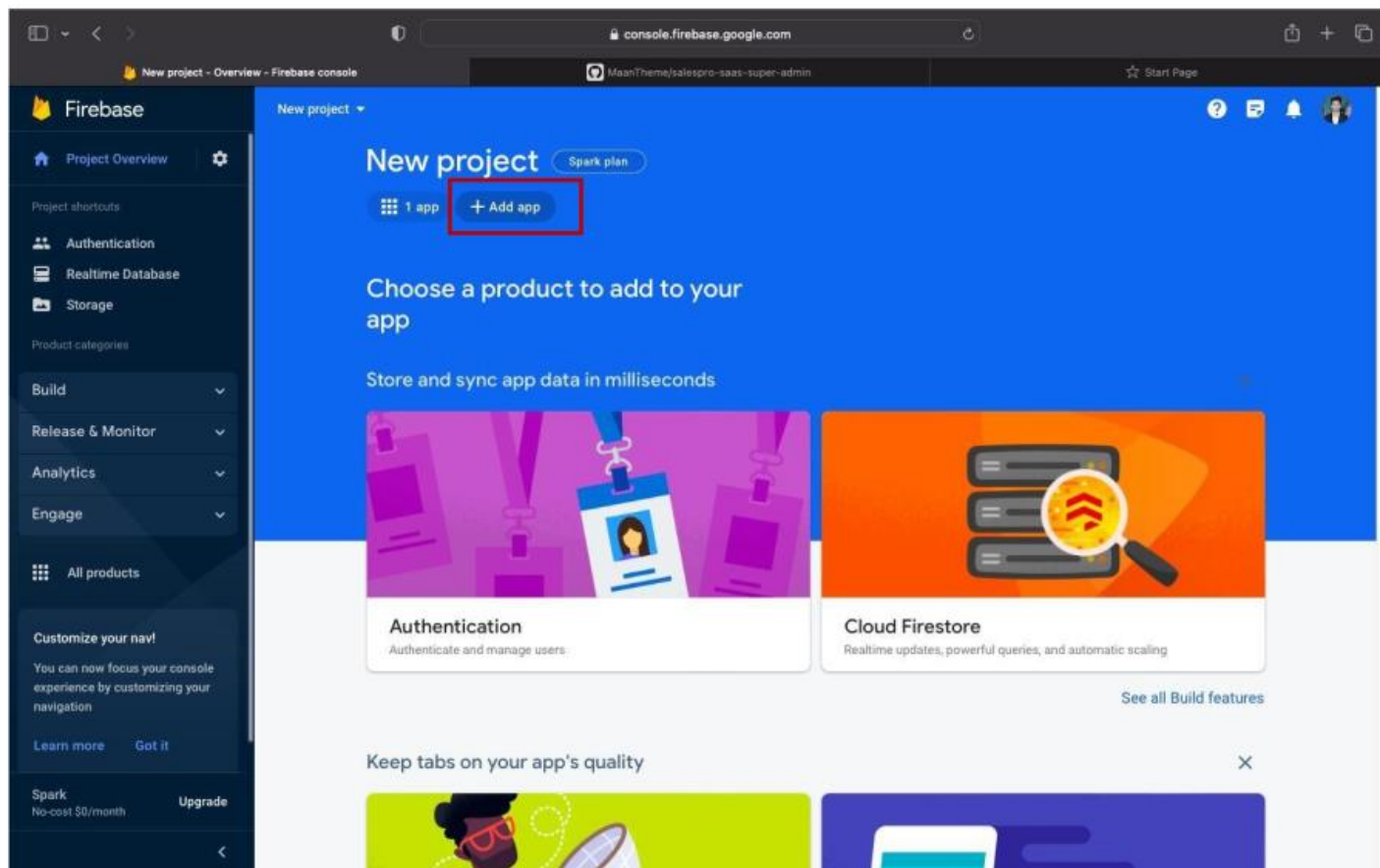
The rest of the file includes imports for Firebase, Flutter, and other packages, followed by a `GlobalKey` definition and a `getSaleID` function.

After doing all the tasks successfully, without any error then build the file for server upload.

POS SaaS Mobile App Setting

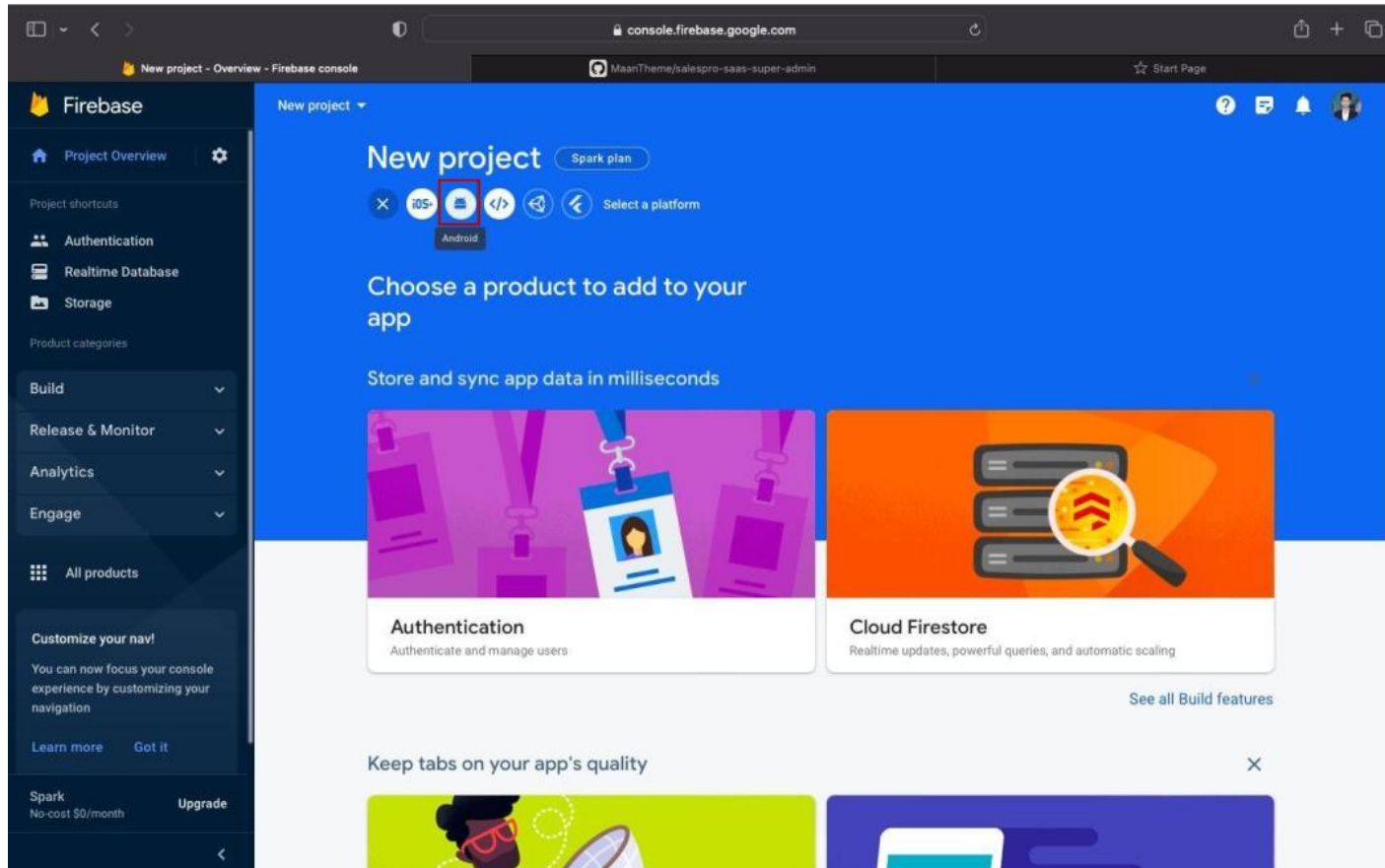
Run the app like the web and super admin in any environment.

Then setting up the firebase with the app. For this you have to go to the firebase console again. And go to the project which we created for the POS saas web & super admin.

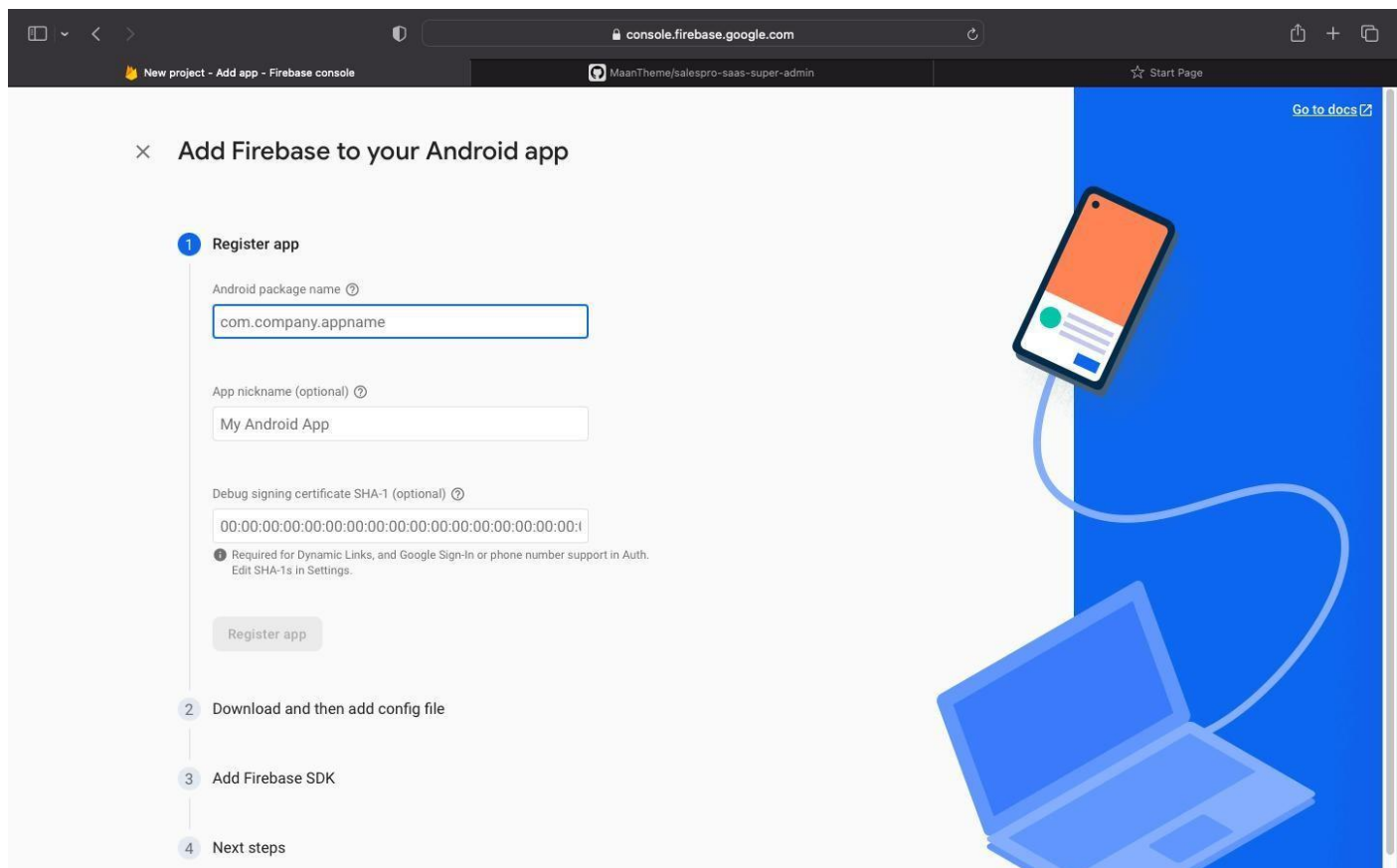


Now click on the **Add app** button to connect the app with the firebase.

Select the android button from the list.

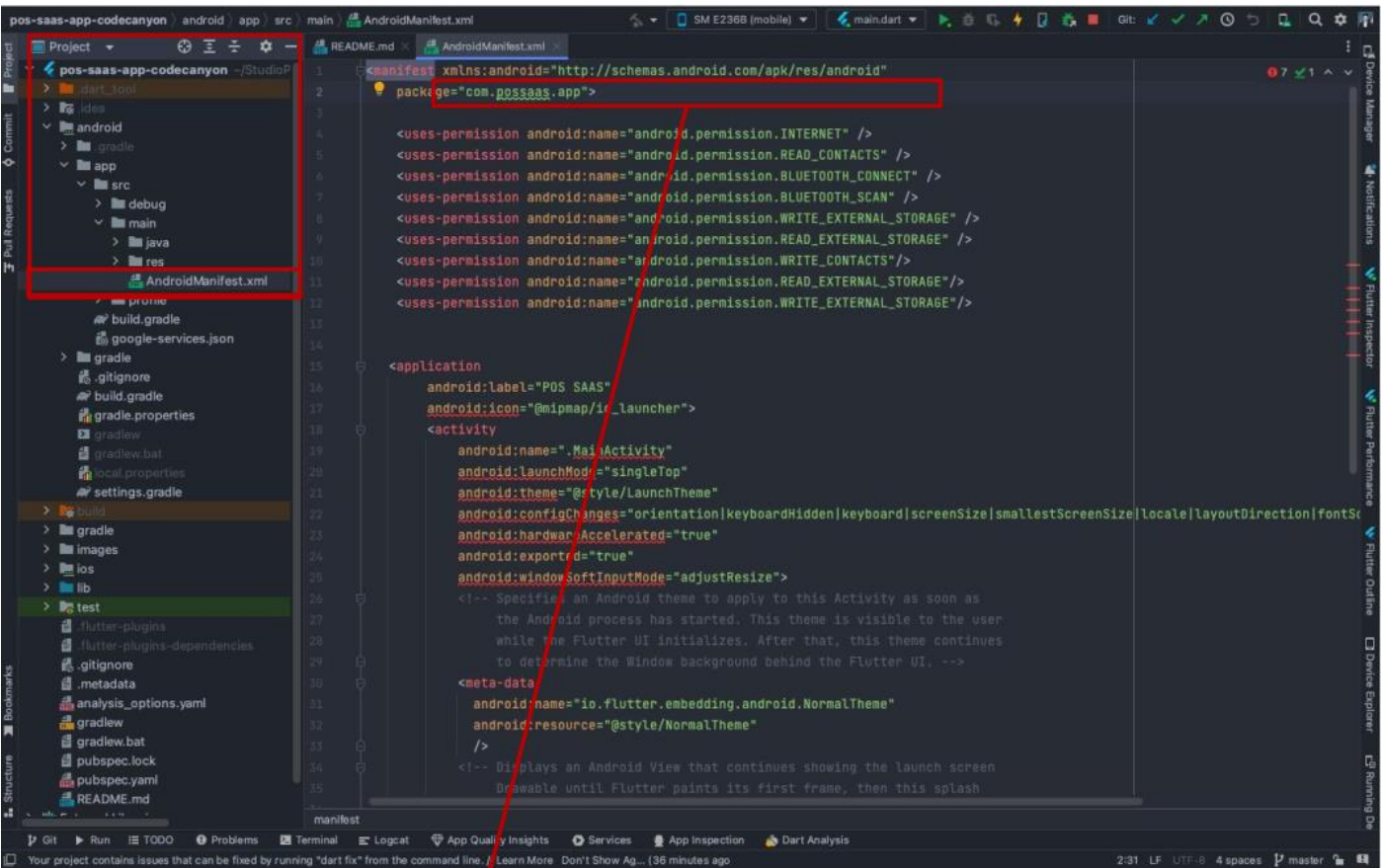


A new page will appear and there you have to give all the information.



For android package name you have to go to your environment. And go to the AndroidManifest.xml file. And at the top of the screen you can see the package name.

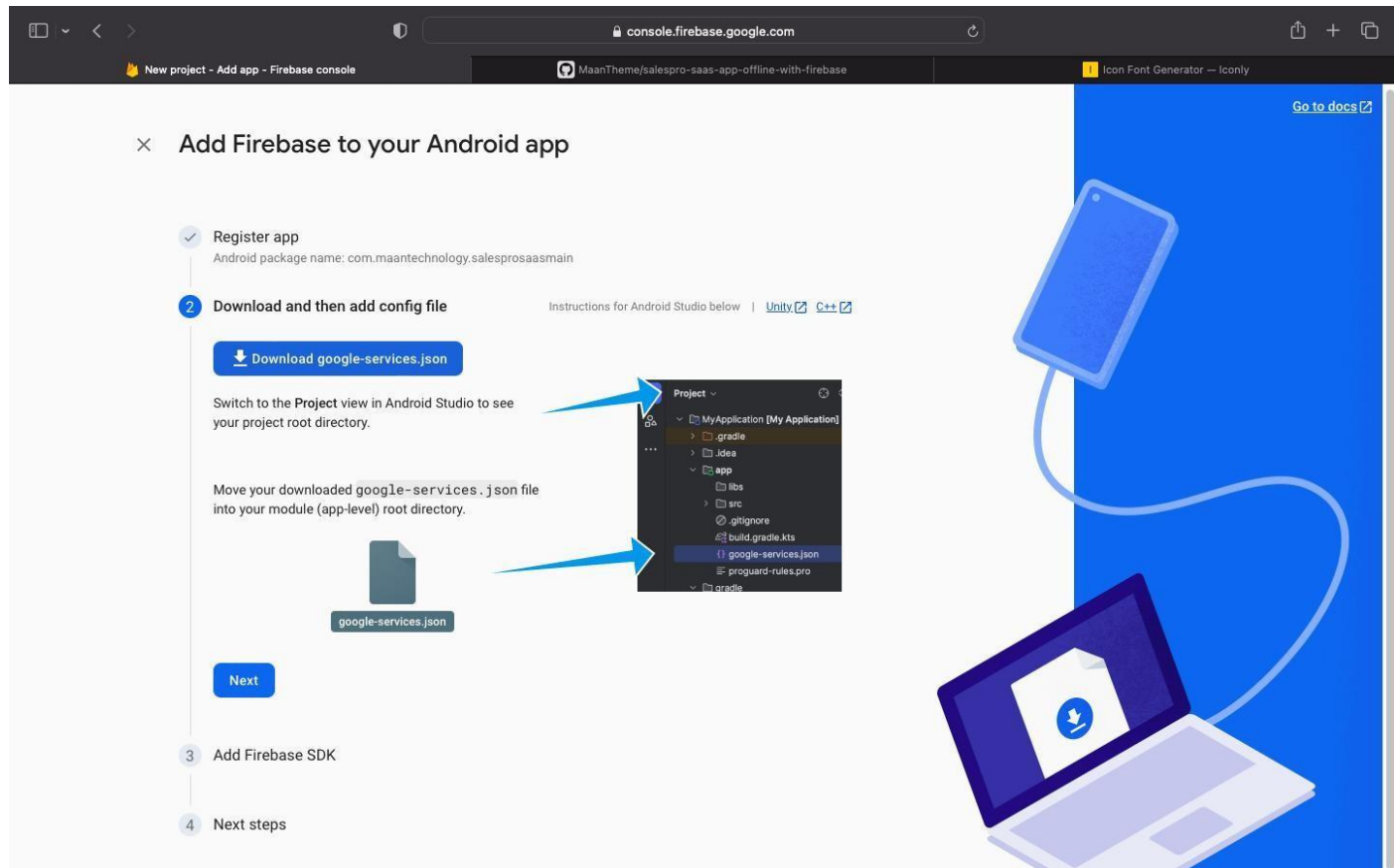
path: Project/android/app/src/main



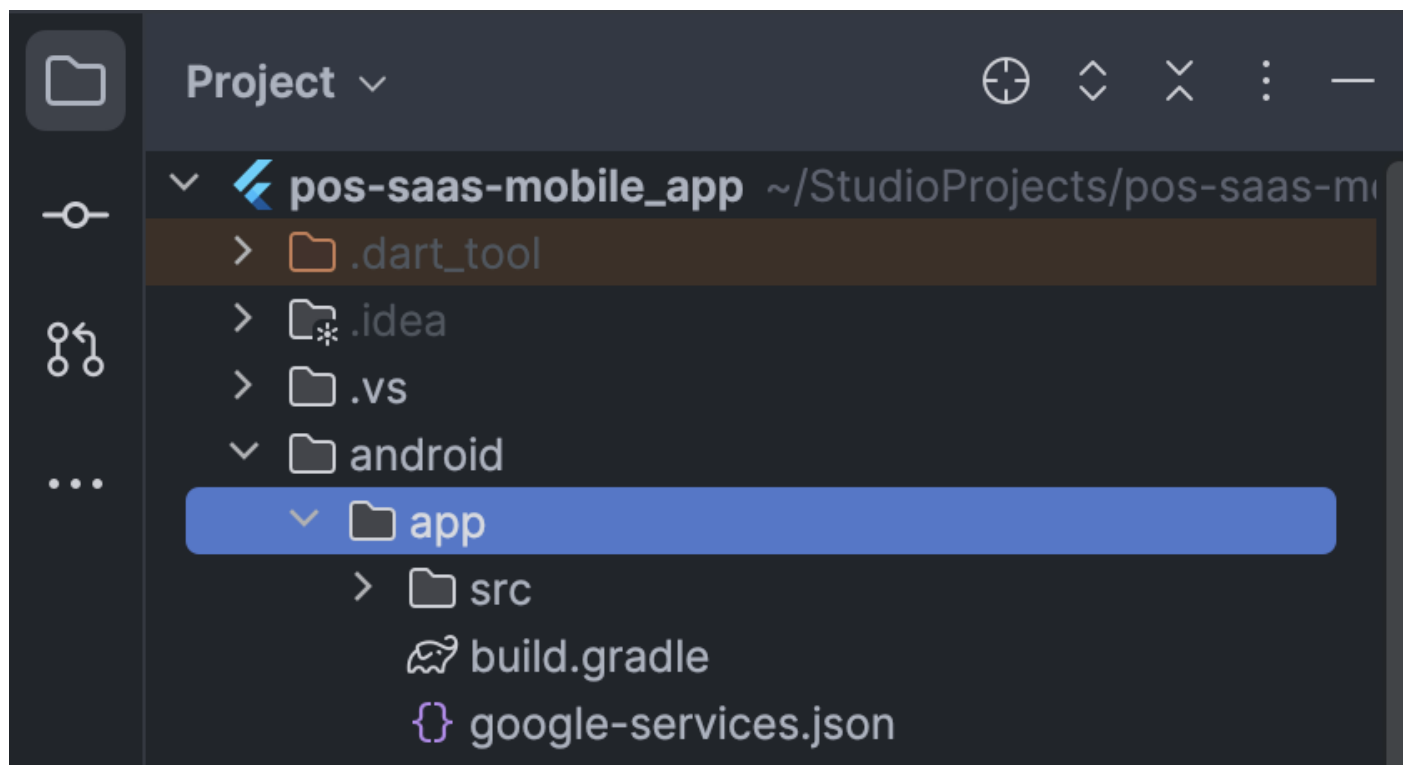
Copy the package name (inside the double quotation)from there and paste it to the Android package name and press on Register App

[illegible]

Then it will generate a `google-services.json` file. Which you have to download and press on next again next & then Continue to console. Don't forget to download the file.



Now copy the `google-service.json` file, and open your project. Paste the `google-service.json` file inside the app. To do it easily just select the app directory and paste the file.

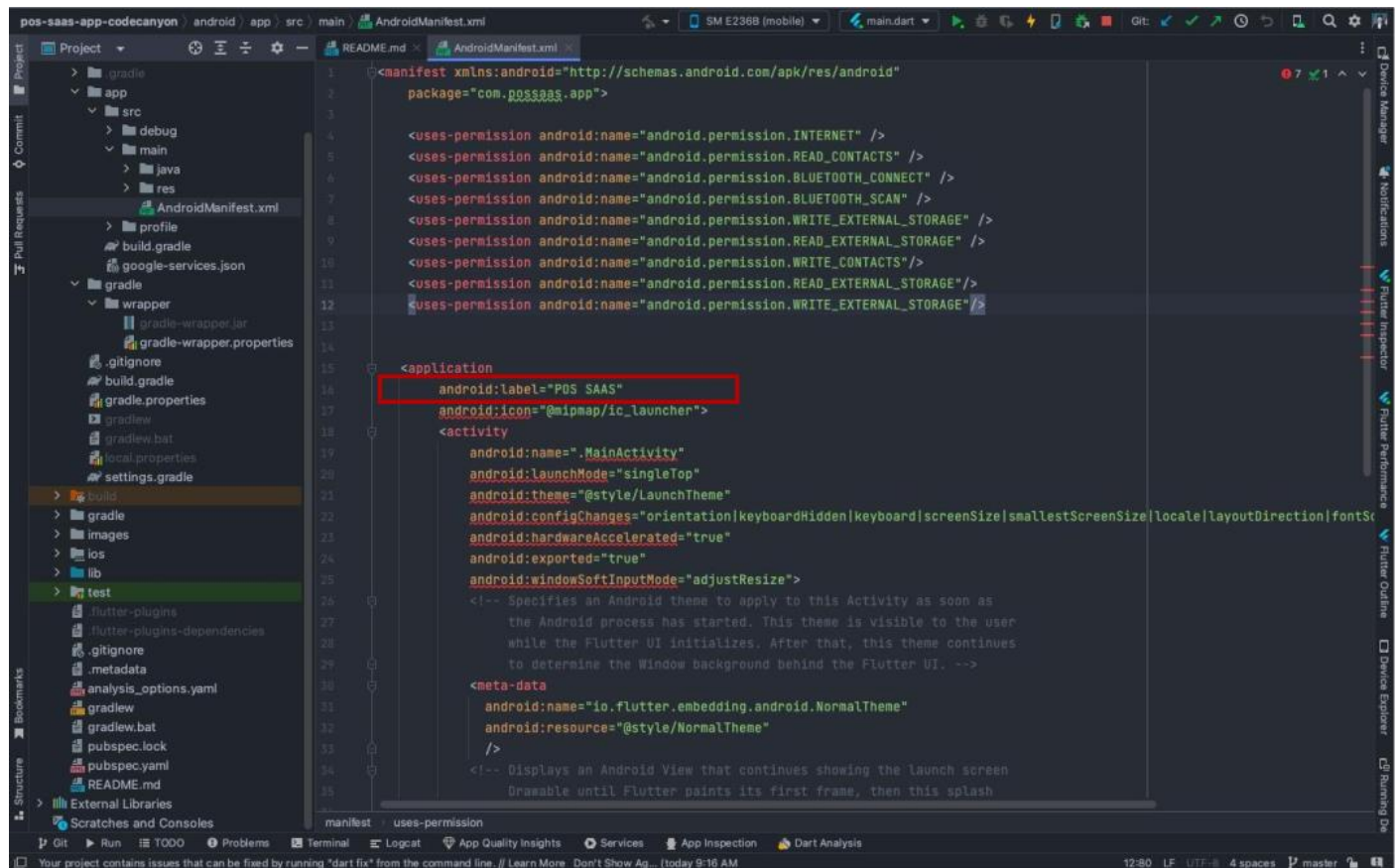


Now go to the environment and run the project.

Change the app name

To change the app name you have to go to the AndroidManifest.xml file.

path: Project/android/app/src/main

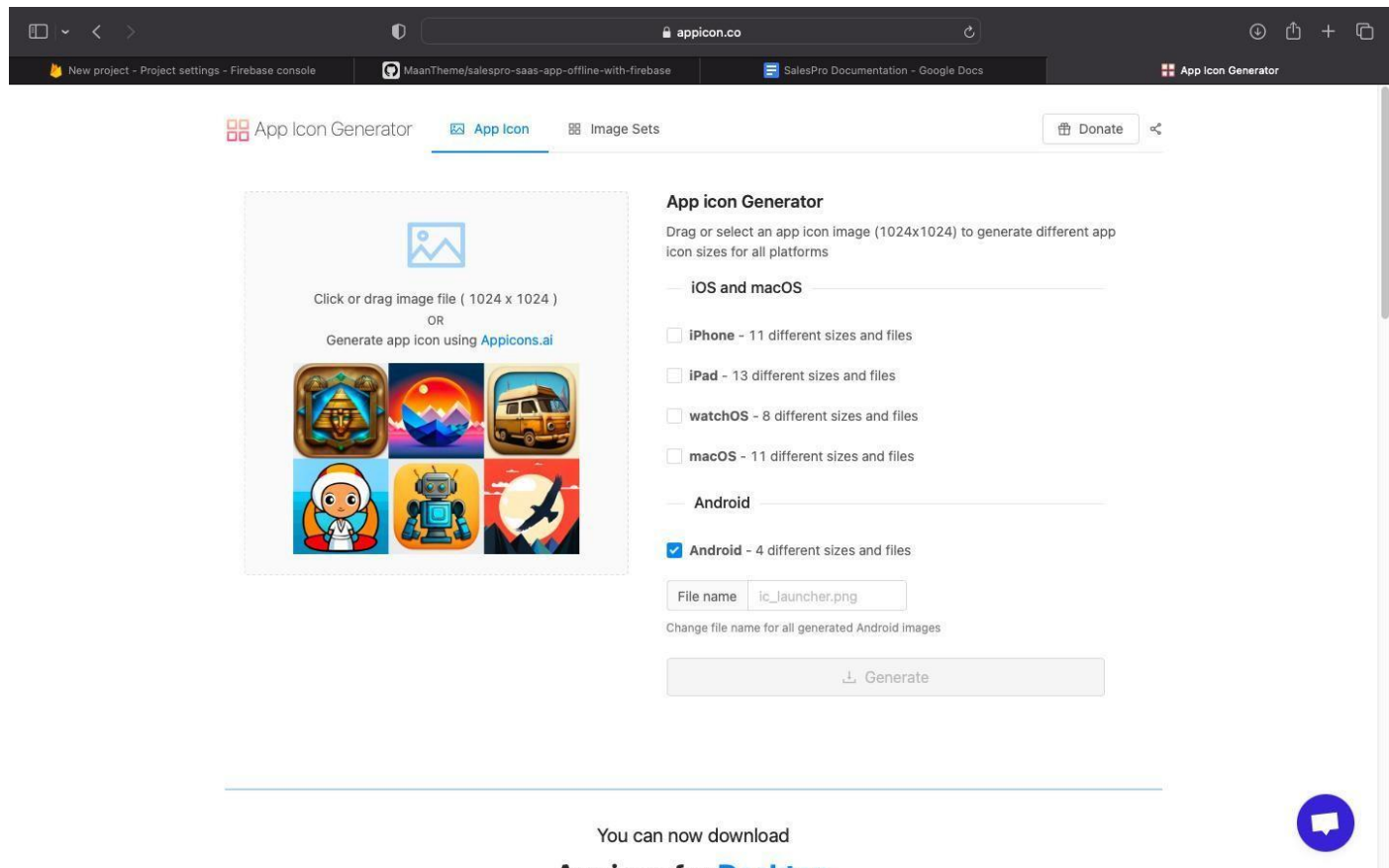


After opening the file change the **android:label = "AppName"**

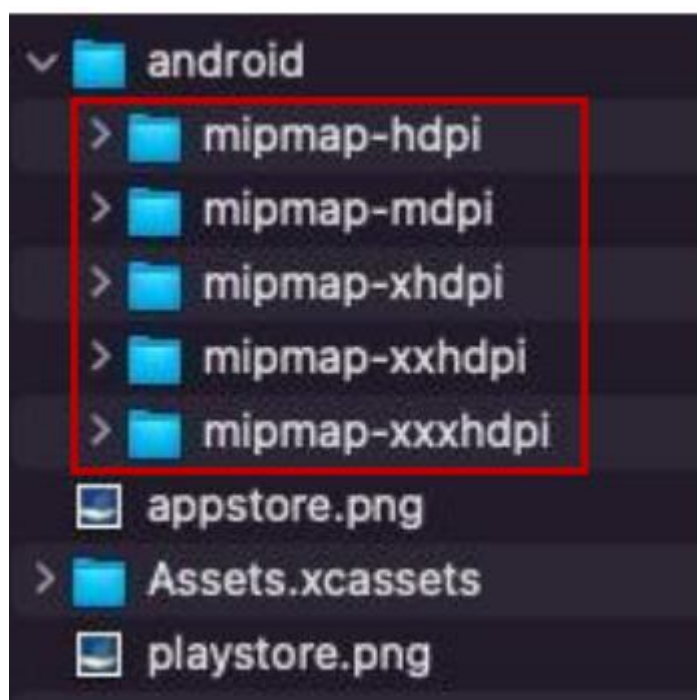
And build the app. The App name will be changed.

Change the app logo

For this first you have to design a logo for your app. Then to create different sizes of logo easily, you have to go to <http://appicon.co>. Then drag and drop or upload your image file to the site. And select only Android from the list. No need to give any file name. And click on generate. It will generate a zip file for you.

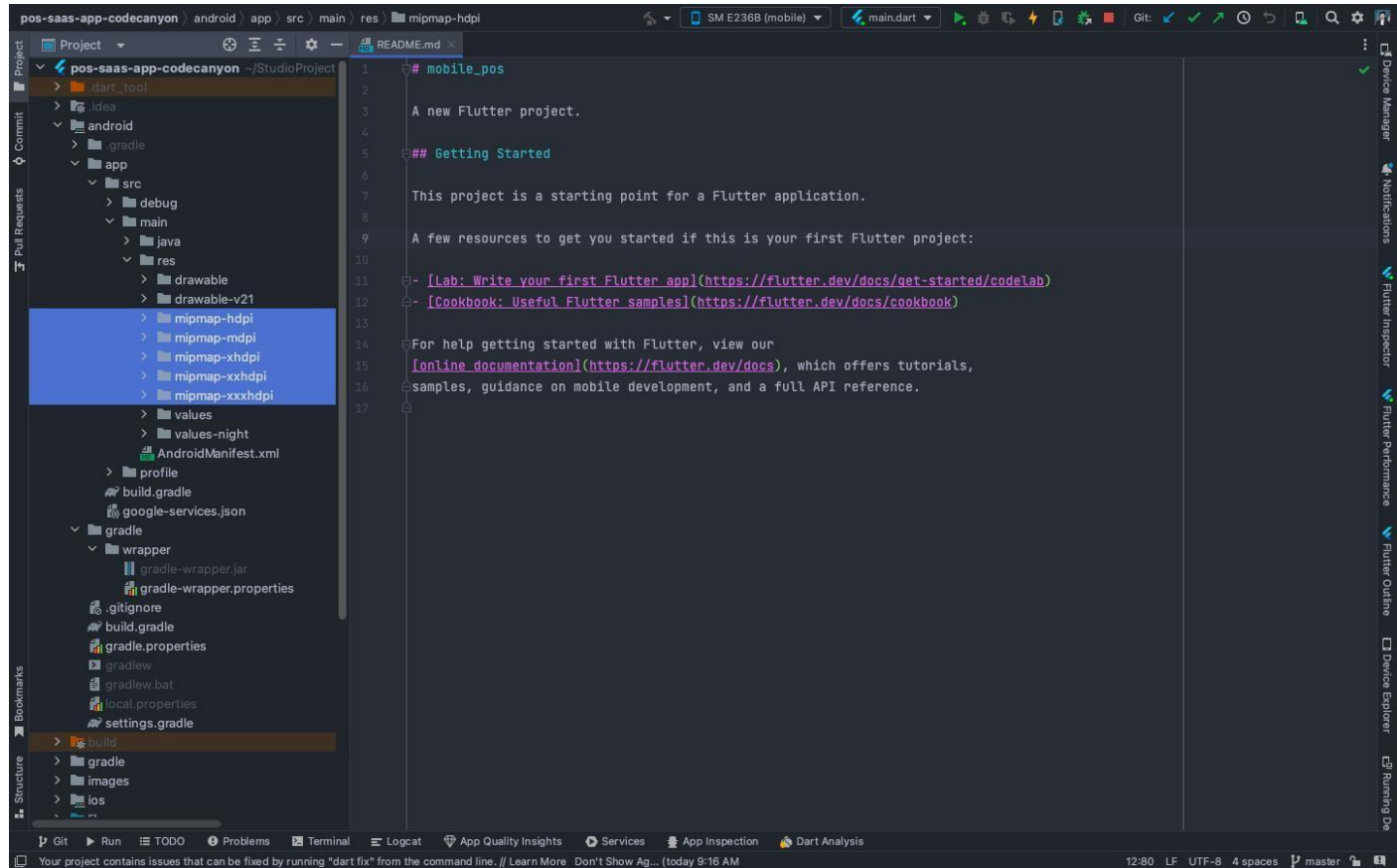


Now Unzip the file and enter the folder.



Copy marked these 5 folders. And go to the environment where the project is opened.

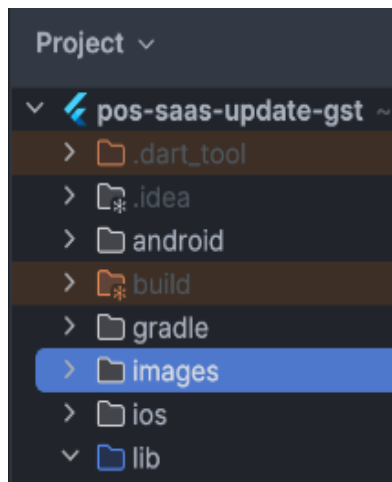
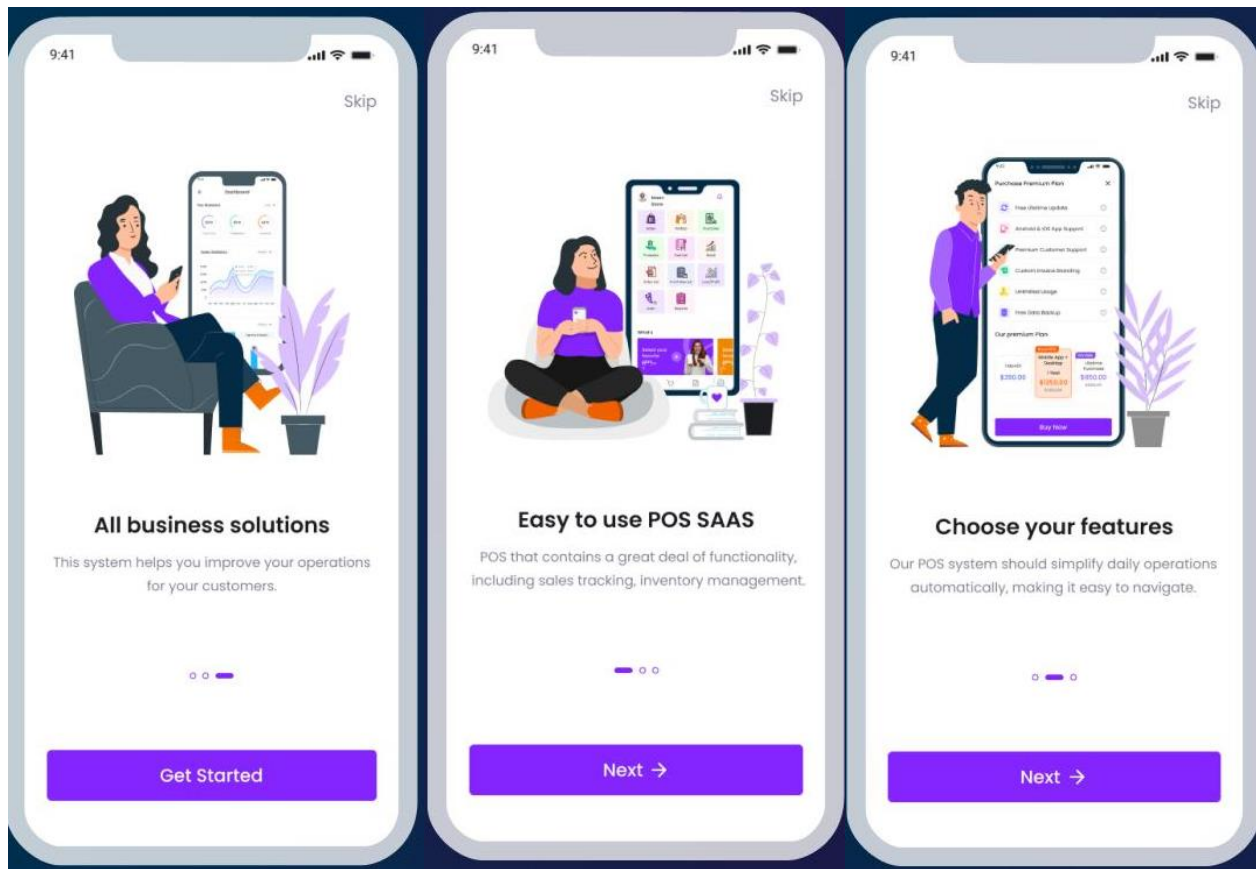
Now from the project go to the res directory.
path: Project/android/app/src/main/res/



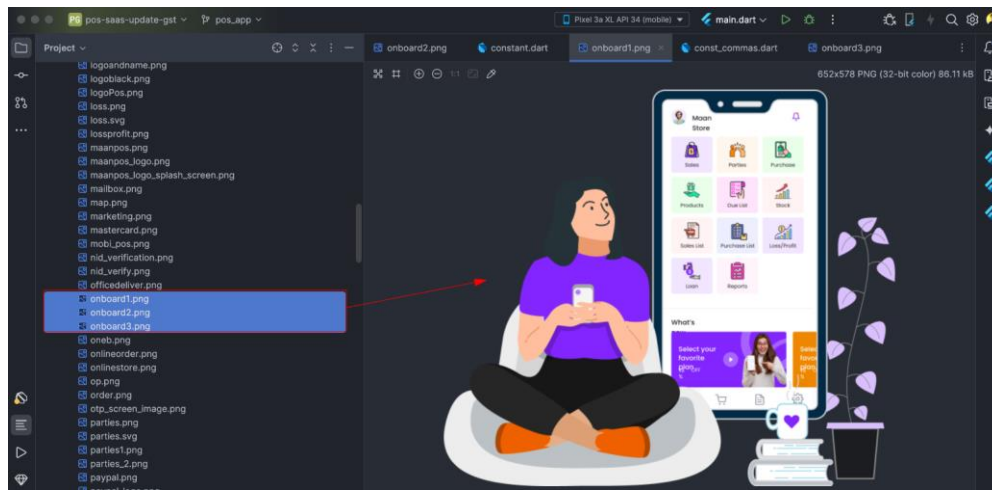
Now select the marked five folder and paste the folder you have copied from the zip and replace it.

Now build the APK. It will show the new logo.

Change Onboarding image:



- Open images folder



Select Images:

- Choose three images that you want to use for the onboarding screens.
- Ensure the images are appropriate and formatted correctly for the app's display requirements.

Rename Images:

- Rename the selected images with the following filenames:
 - **First Image:** `onboard1.png`
 - **Second Image:** `onboard2.png`
 - **Third Image:** `onboard3.png`
- Make sure to keep the file extension as `.png`.

Replace Existing Images:

- Locate the folder where the current onboarding images are stored. This is typically named `images`
- Copy and paste your newly renamed images into this folder.
- If prompted, allow the new images to replace the existing ones.

Restart the App:

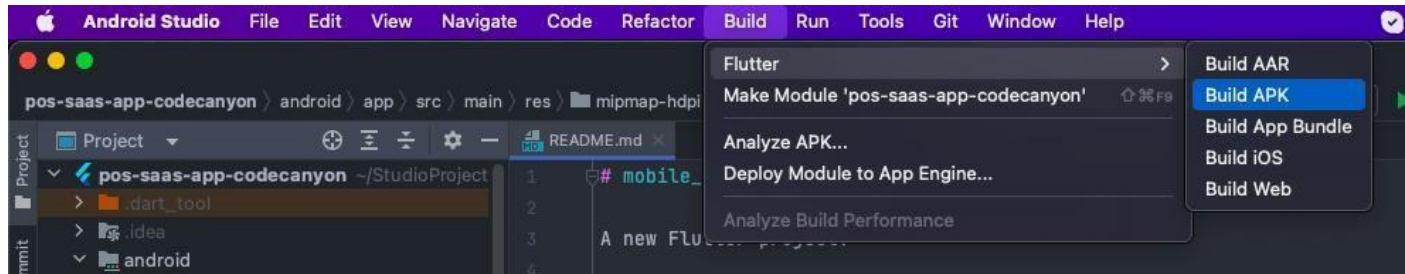
- Close the app completely to ensure all changes take effect.

Restart the app and verify that the new onboarding images are displayed correctly.

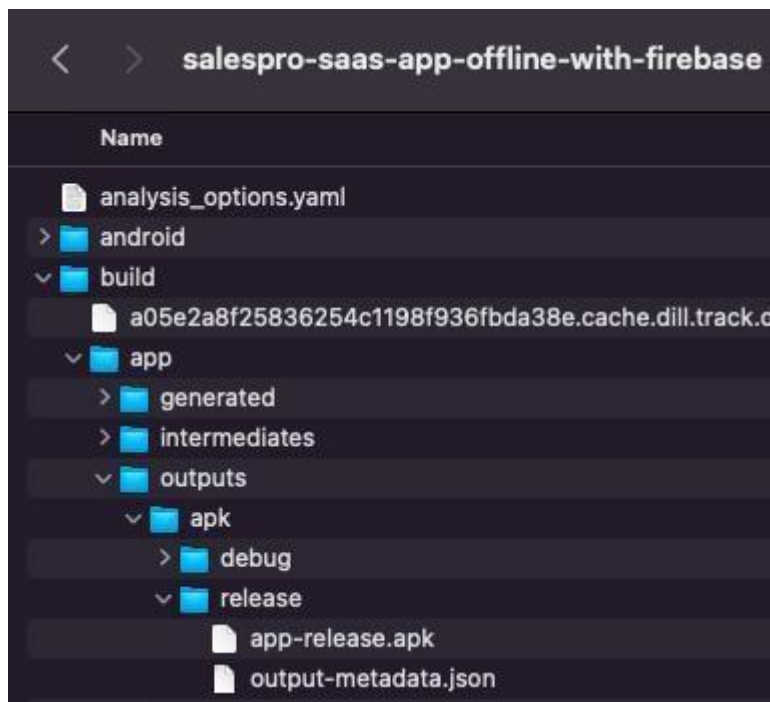
Project version Update:

Build release apk

To build an APK, go to the menu section of your environment, then select the **Build** section and move the cursor on **Flutter** and then select the **Build APK**. That's all. After sometimes it will build a release APK file.



After completing the build process, go to the build directory of the main project directory. It will build a new release apk file.



path: build/app/outputs/release

Inside the release directory you can find the release.

To publish your app in google play store you can follow this instruction:

<https://docs.flutter.dev/deployment/android>

Help & Support

Support skype: <https://join.skype.com/invite/kEPqImF1Vfqk>

Support email: acnooteam@gmail.com

You can expect an answer within 24-48 hours, usually as soon as possible in the order they were received.