



## Introduction To MPOS SAAS

### POS Inventory Management Web App With Firebase, Subscription Packages, PayPal and Invoice Printing

**MPOS SAAS** - Flutter POS Inventory Management App with Firebase is as the name says is dominating and powerful Flutter app contains complete app templates, ready to use screens, widgets, illustrations and stunning screens covering POS Inventory web app. MPOS Saas - Flutter POS Inventory Management Web App with Firebase is developed with the highest quality, ease of reusing widgets, fast, and completely user-friendly interface. You can easily use these to make your Flutter POS Inventory Management Web App. MPOS Saas- Flutter POS Inventory Management Web App with Firebase makes the developer job easy to achieve the modern look and feel of the web application. It saves your hustle and time to develop a perfect design UI for modern-day-use web applications. There is a Subscription System for generating revenue using PayPal (More coming soon).

#### 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: Tue Aug  6 10:03:57 on ttys005
[tahmidtarongo@Tahmids-MacBook-Air ~ % flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.22.3, on macOS 14.6 23G80 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 (4 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.

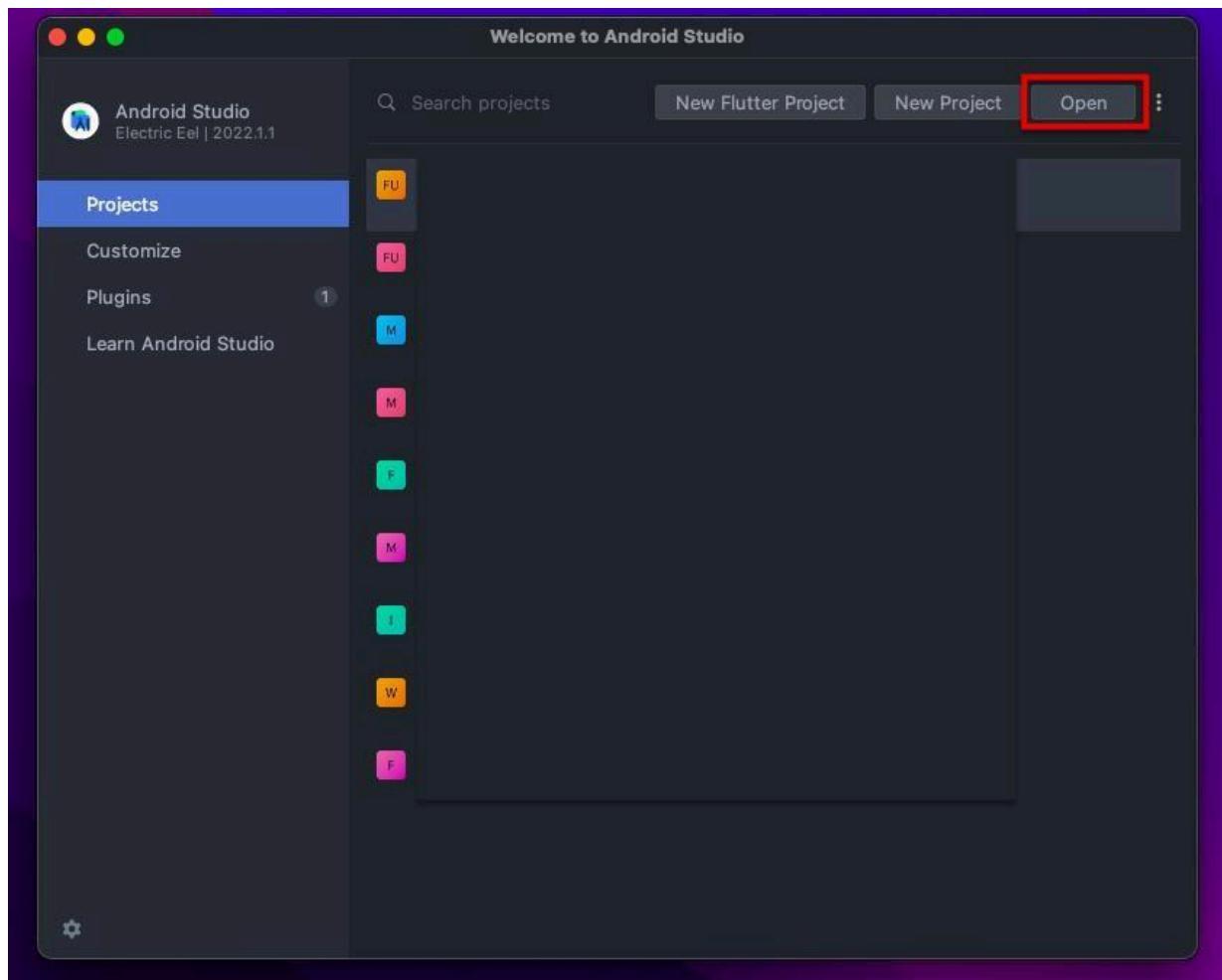
## **Modify App**

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.

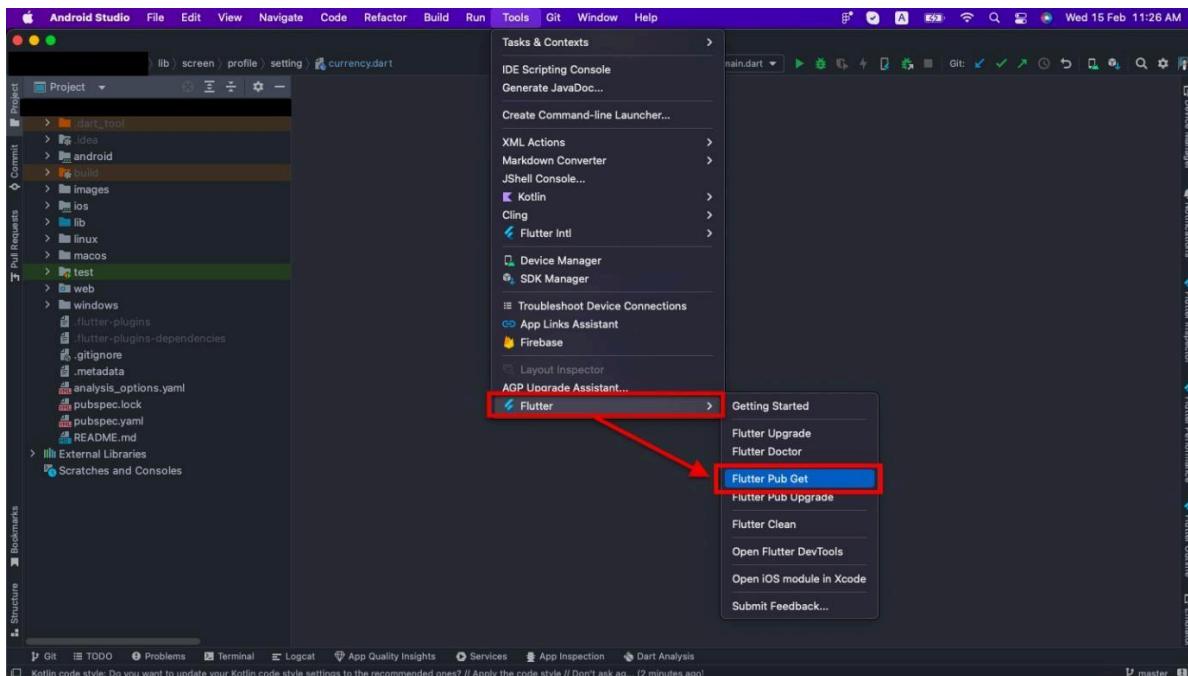


3. Choose the Pos Saas - App / Pos Saas Web/ Pos Saas super admin directory and open it. Note that you have to set the web first. It is located at download or the folder you selected when it was downloaded from the marketplace. This will open the application source code in android studio.

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

## Get Flutter Packages

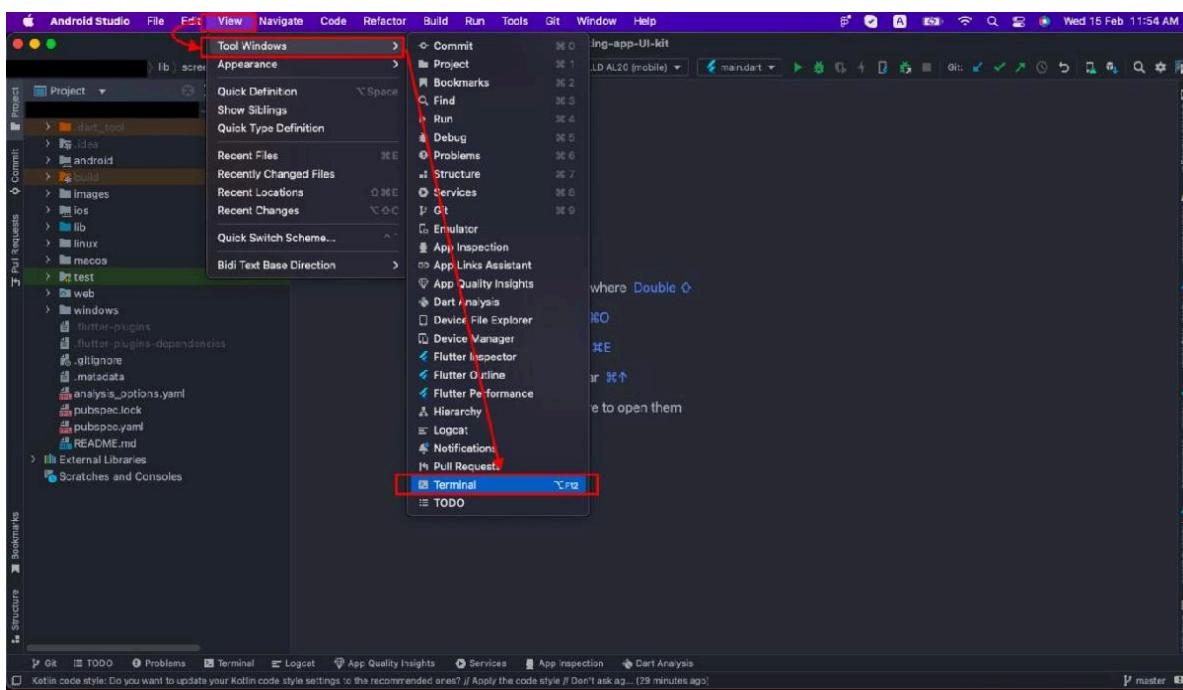
Now you have to open tool option from the topbar and from the opening menu you have to choose Flutter and then Flutter Pub Get & press or select that.  
Also if required, you have to give the sdk path, & allow for the configuration.



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.



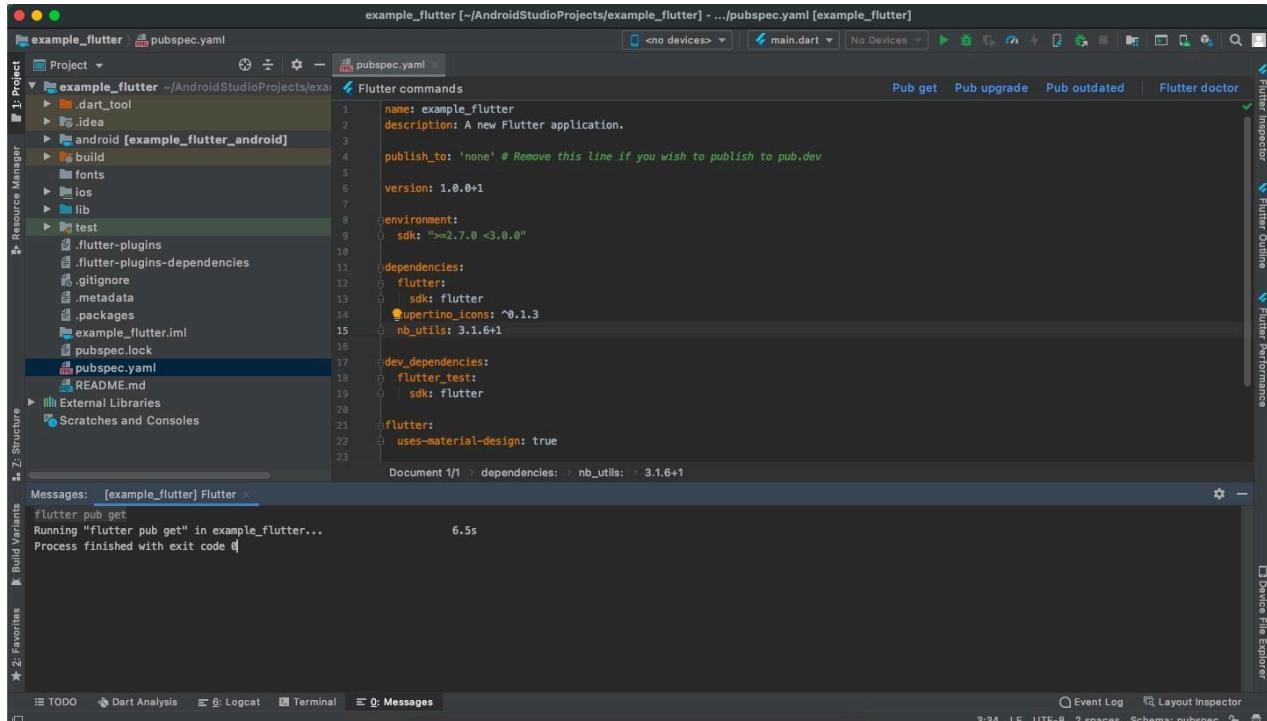
Now type flutter pub get on terminal and press enter.

This will get the dependencies for the application.

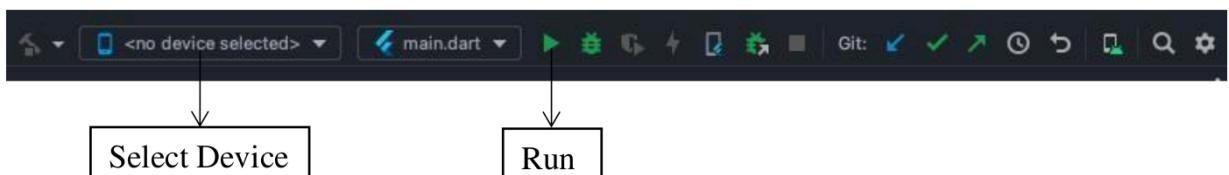
Make sure that you are connected to the internet while performing this task

## Build and Run App

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

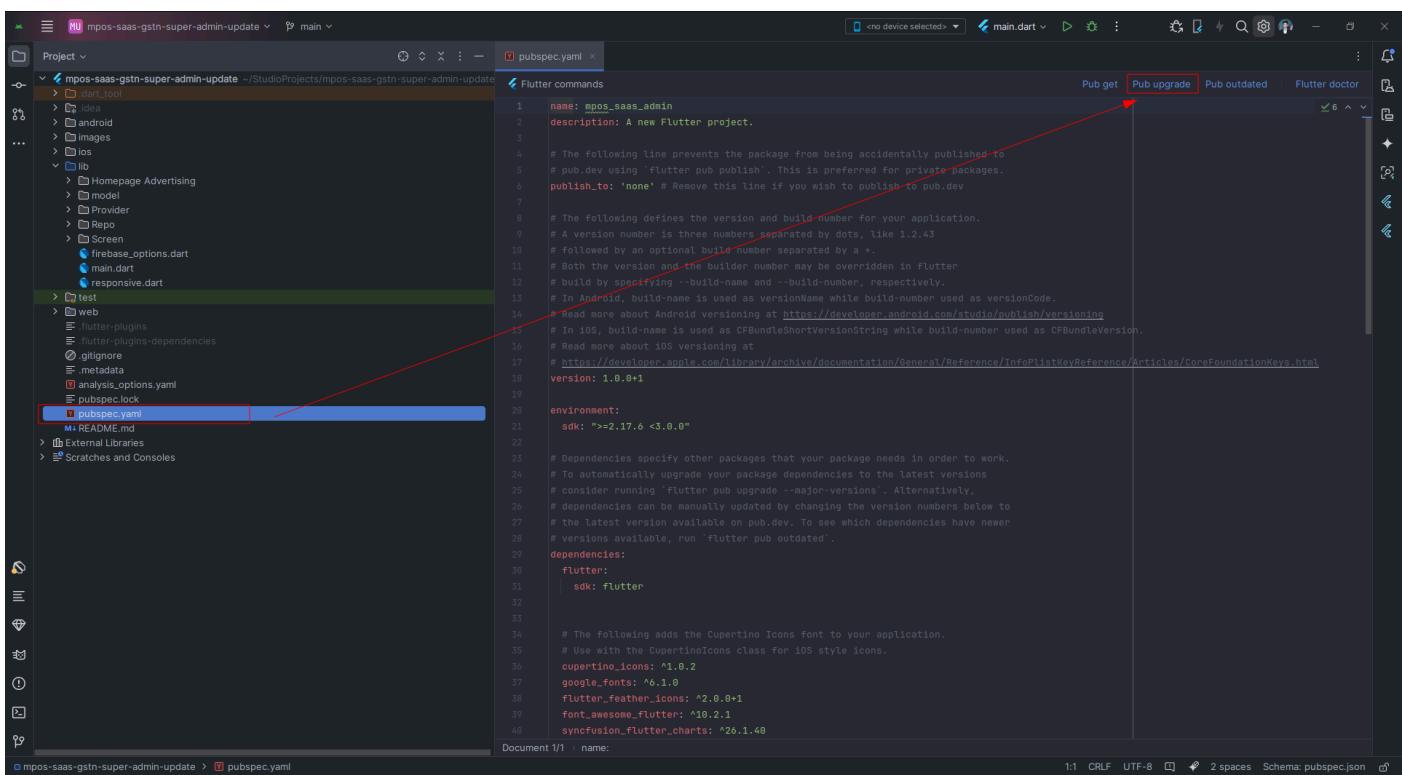


1. Locate the main Android Studio toolbar.
2. In the target selector (<no device selected>), select an Android device for running the app. Or if you want to run the web app select the available browser from the dropdown menu. If none are listed as available, select Tools > Android > AVD Manager and create 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.

# Project Version Update



The screenshot shows the Android Studio interface with a Flutter project named 'mpos-saas-gstn-super-admin-update'. The 'pubspec.yaml' file is open in the code editor. In the top right toolbar, there are several buttons: 'Pub get', 'Pub upgrade' (which is highlighted with a red box and an arrow), 'Pub outdated', and 'Flutter doctor'. The code in the pubspec.yaml file includes standard configuration like package name, version, and dependencies.

```
name: mpos_saas_admin
description: A new Flutter project.

# The following line prevents the package from being accidentally published to
# pub.dev using `flutter pub publish`. This is preferred for private packages.
publish_to: 'none' # Remove this line if you wish to publish to pub.dev

# The following defines the version and build number for your application.
# A version number is three numbers separated by dots, like 1.2.43
# followed by an optional build number separated by a +
# Both the version and build number may be overridden in flutter
# by specifying --build-name and --build-number, respectively.
# In Android, build-name is used as versionName while build-number used as versionCode.
# In iOS, build-name is used as CFBundleShortVersionString while build-number used as CFBundleVersion.
# Read more about iOS versioning at
# https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/coreFoundationKeys.html
version: 1.0.0+1

environment:
  sdk: ">=2.17.6 <3.0.0"

# Dependencies specify other packages that your package needs in order to work.
# To automatically upgrade your package dependencies to the latest versions
# consider running `flutter pub upgrade --major-versions`. Alternatively,
# dependencies can be manually updated by changing the version numbers below to
# the latest version available on pub.dev. To see which dependencies have newer
# versions available, run `flutter pub outdated`.
dependencies:
  flutter:
    sdk: flutter

  # The following adds the CupertinoIcons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.2
  google_fonts: ^6.1.0
  flutter_feather_icons: ^2.0.0+1
  font_awesome_flutter: ^10.2.1
  syncfusion_flutter_charts: ^26.1.40
```

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:

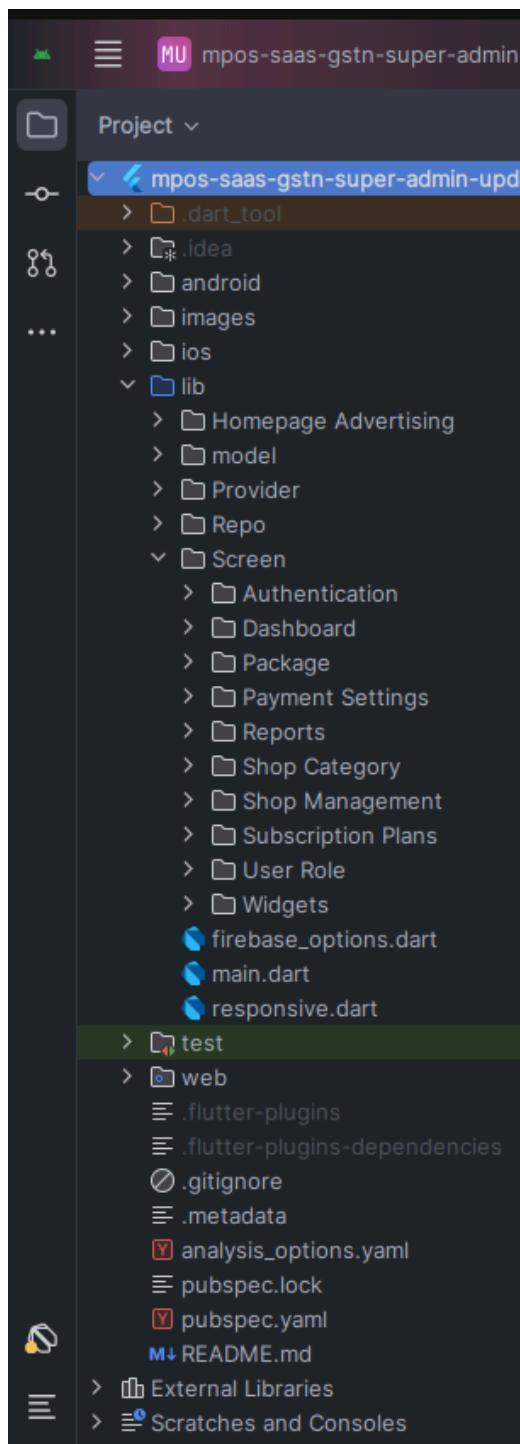
- Ensure your computer is connected to the internet.
- In your IDE, locate and click the Pub upgrade button. This button is typically found at the top of the editor or in the pubspec.yaml tab.
- Alternatively, you can run the following command in your terminal from the project directory:

## flutter pub upgrade

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

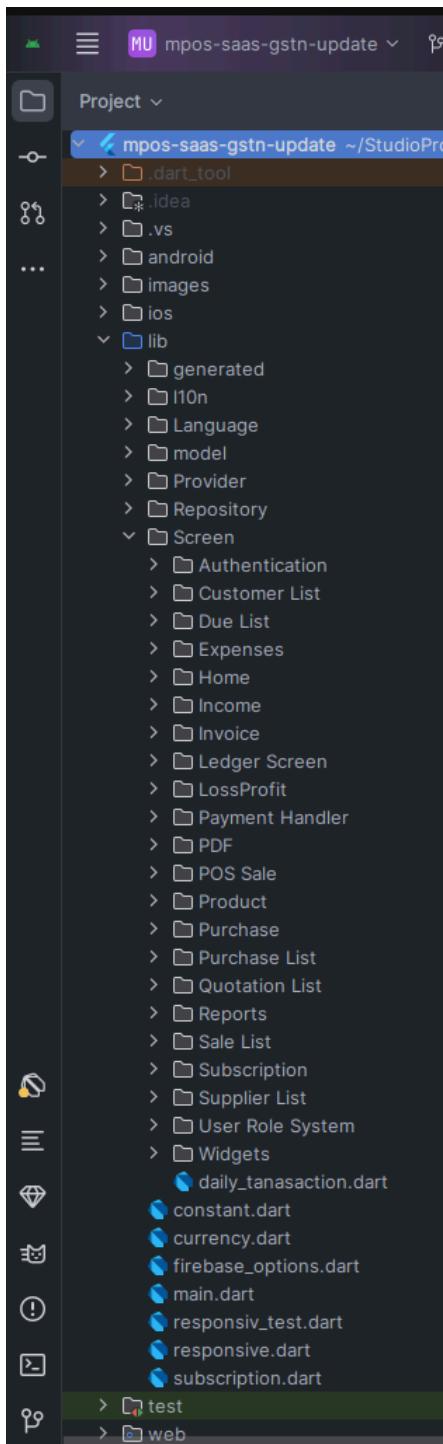
## Project Structure:

### MPOS SAAS Superadmin



- **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
  - **firebase\_option.dart:** all the firebase credentials are stored here.

## MPOS SAAS Web

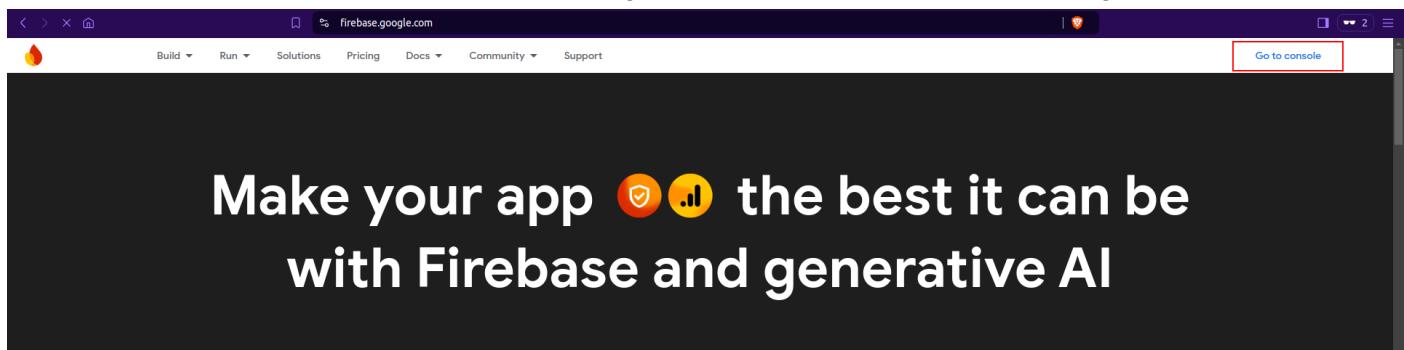


- **images**: Apps/widgets/snippets images as well as appIcon are stored here.
- **lib**: Applications main files and directories are located here.
  - **Screen**: Full app's UI codes are stored here.
  - **Widgets**: All the reusable widgets are stored here
  - **firebase\_option.dart**: all the firebase credentials are stored here.

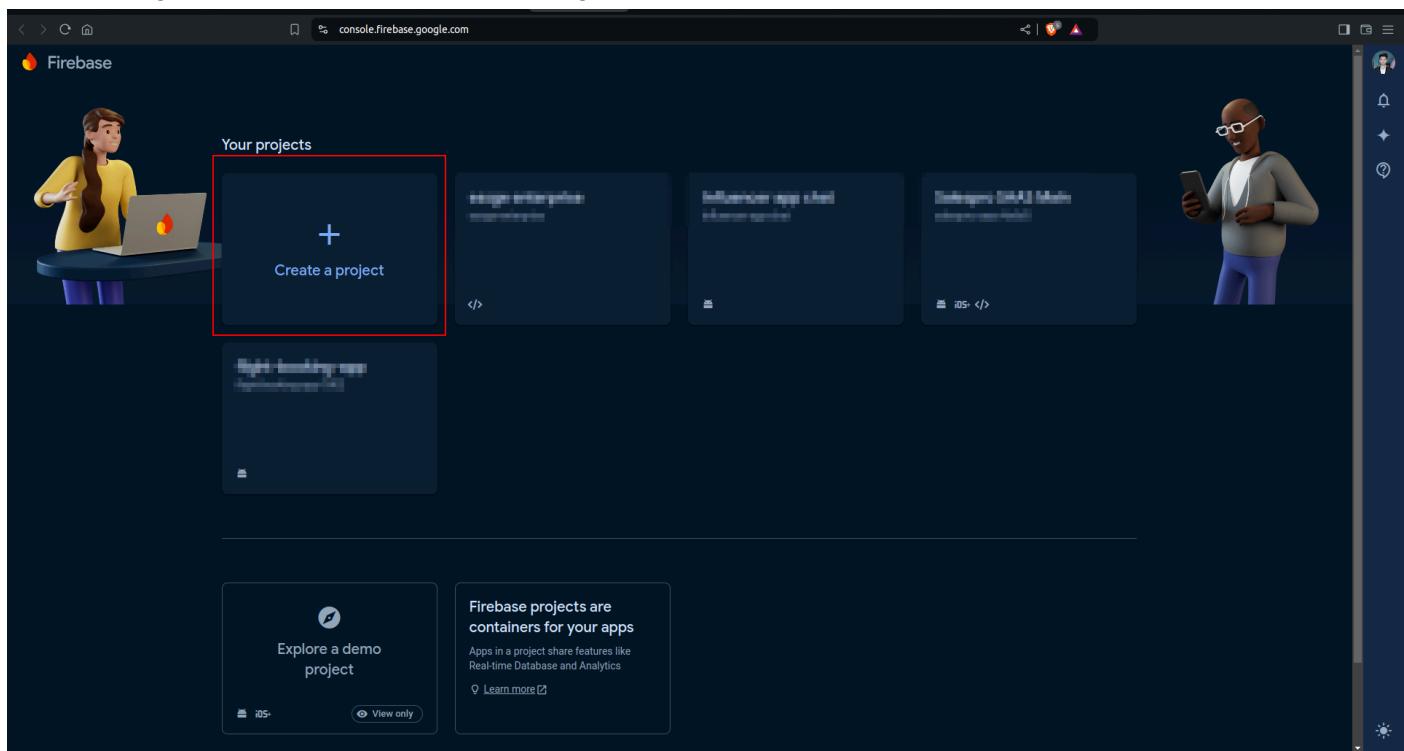
## Firebase Setup

### Step 1

First of all you have to go to <https://console.firebaseio.google.com/> and click on Go to the console which is on the right side of the top bar menu.(if not signed in)

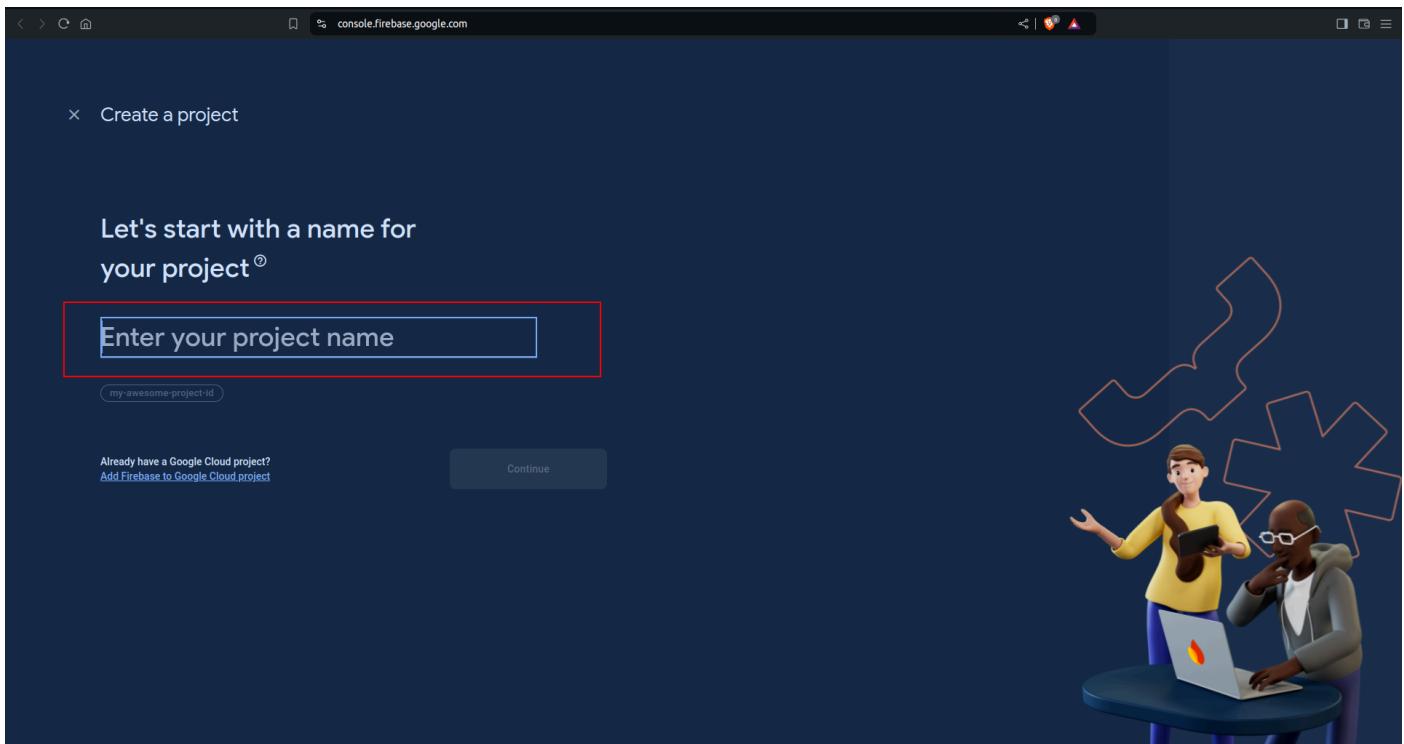


After tapping on the Go to console a new page will be open.

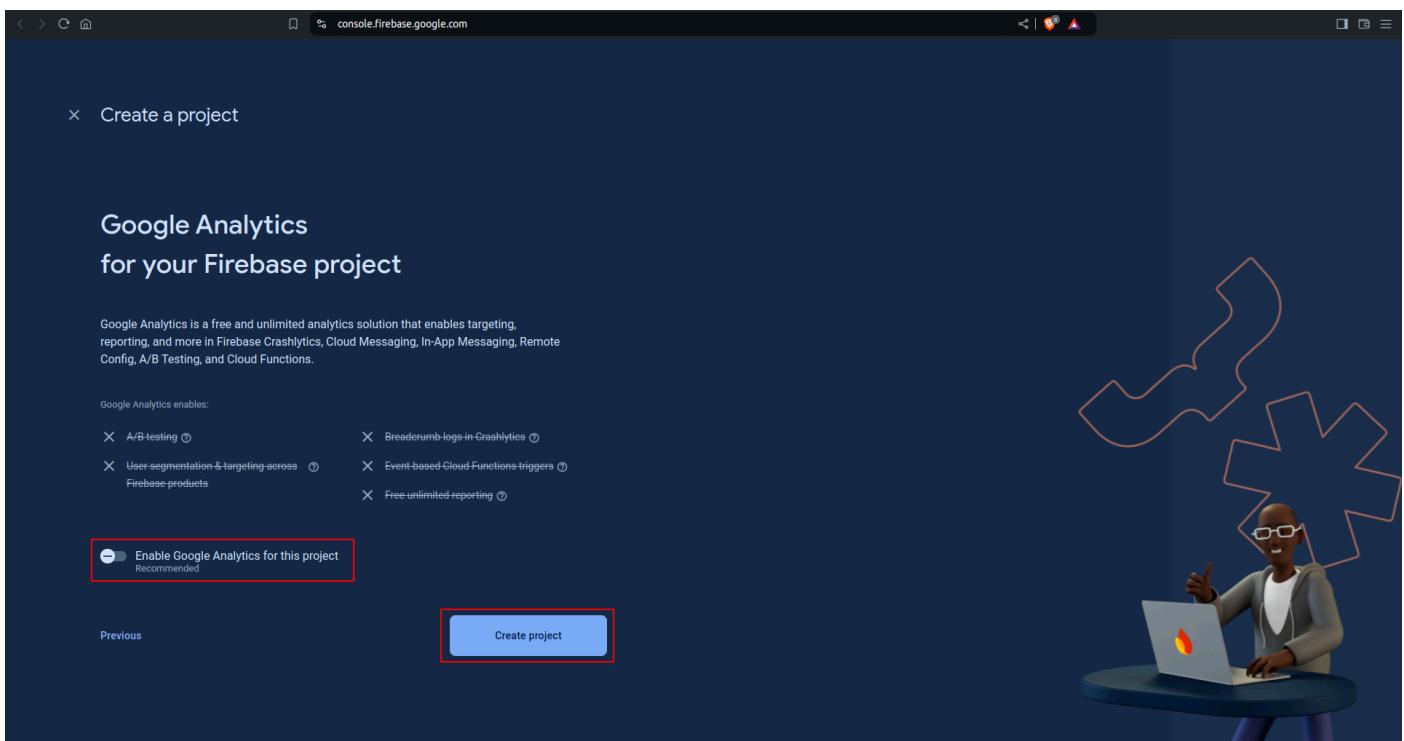


Now you have to create a new project which you want to connect with the app. Click on **Create a project** button to create a new one.

A screen will open like this.

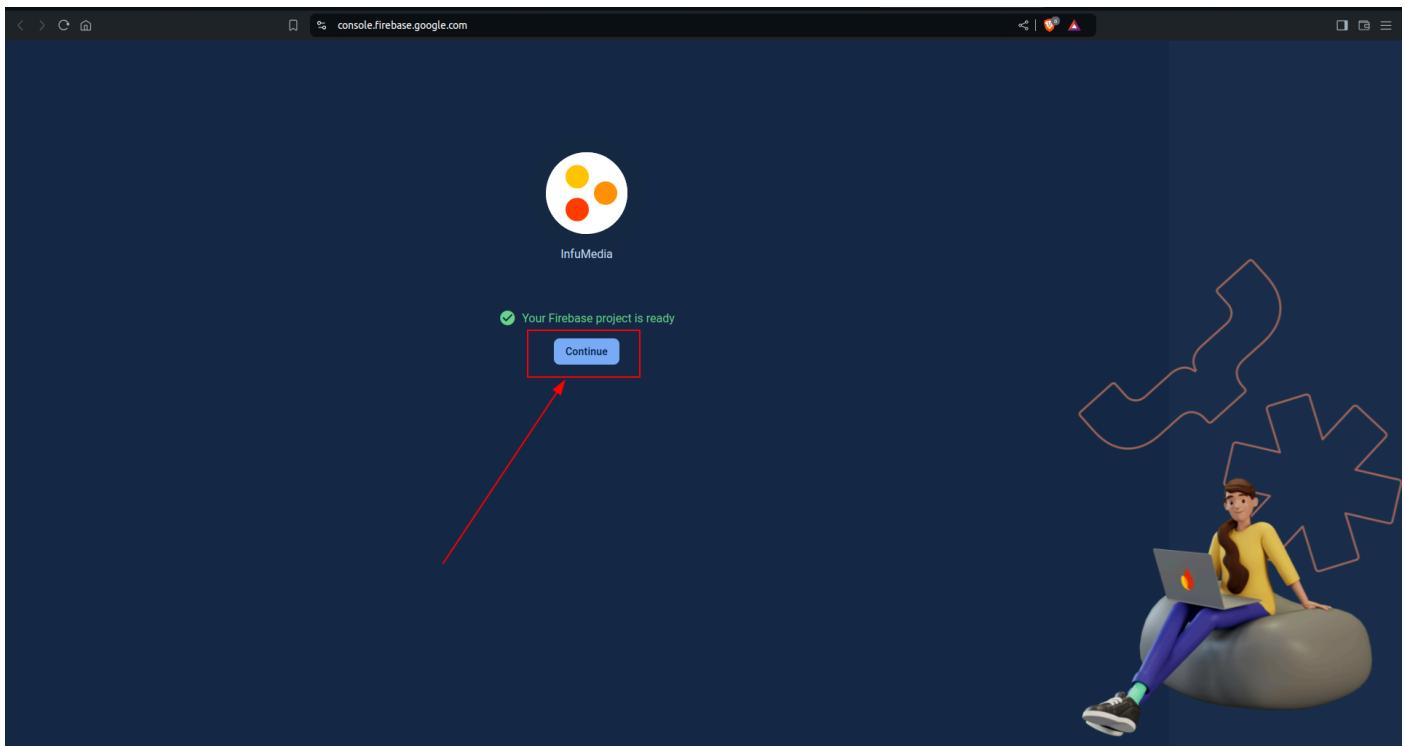


Here you have to enter a name for your project. then press continue to go to the next step.



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 **Create project**.

Here a new project is created for you. Press continue to view the project.



A screenshot of the Firebase console showing the 'Mpos' project overview. On the left, there's a sidebar with navigation links like 'Project Overview', 'Build', 'Run', 'Analytics', and 'All products'. The main area shows the project name 'Mpos' and a call-to-action: 'Get started by adding Firebase to your app', accompanied by icons for iOS, Android, and web development. Below this, there's a section titled 'Accelerate app development' with two cards: 'Authentication' (described as an end-to-end user identity solution) and 'Cloud Firestore' (described as providing realtime updates, powerful queries, and automatic scaling). A small modal window is partially visible at the bottom of the main content area.

## 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. This will open a screen like this.

The screenshot shows the Firebase console for a project named 'salespro'. The left sidebar has 'Authentication' selected under the 'Build' category. The main page title is 'Authentication' with the subtitle 'Authenticate and manage users from a variety of providers without server-side code'. Below this is a 'Get started' button, which is highlighted with a red box. To the right is a graphic of a yellow badge with a person's face, a lock, and a key. A video thumbnail for 'Introducing Firebase Authentication' is also visible.

Now press on the **Sign-in method** and select **Email/Password**.

The screenshot shows the 'Authentication' screen with the 'Sign-in method' tab selected. Under 'Sign-in providers', the 'Email/Password' option is highlighted with a red box. Other options shown include 'Phone', 'Anonymous', 'Google', 'Facebook', 'Play Games', 'Game Center', 'Apple', 'GitHub', 'Microsoft', 'Twitter', and 'Yahoo'. Below this is an 'Advanced' section with a 'SMS Multi-factor Authentication' card. A red arrow points from the text above to the 'Email/Password' button.

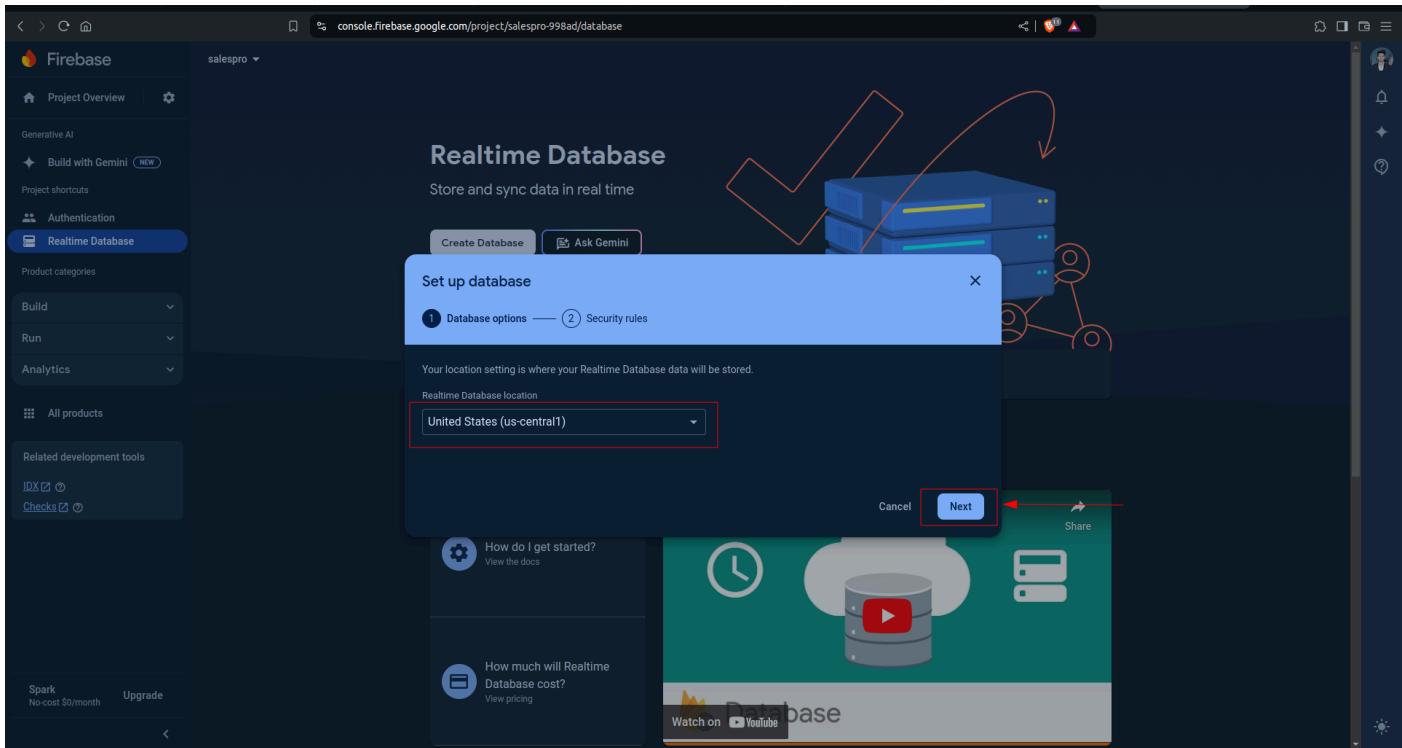
And here enable the **switch icon** and hit the **Save** button.

The screenshot shows the Firebase console's Authentication section. In the 'Sign-in providers' section, the 'Email/Password' provider is selected and has its 'Enable' switch turned on. A red box highlights the 'Enable' switch, and another red box highlights the 'Save' button at the bottom right of the modal. Below this, the 'Advanced' section contains information about SMS Multi-factor Authentication, with a note that it requires an upgrade. A red arrow points from the sidebar's 'Realtime Database' link to the 'Create Database' button on the main page.

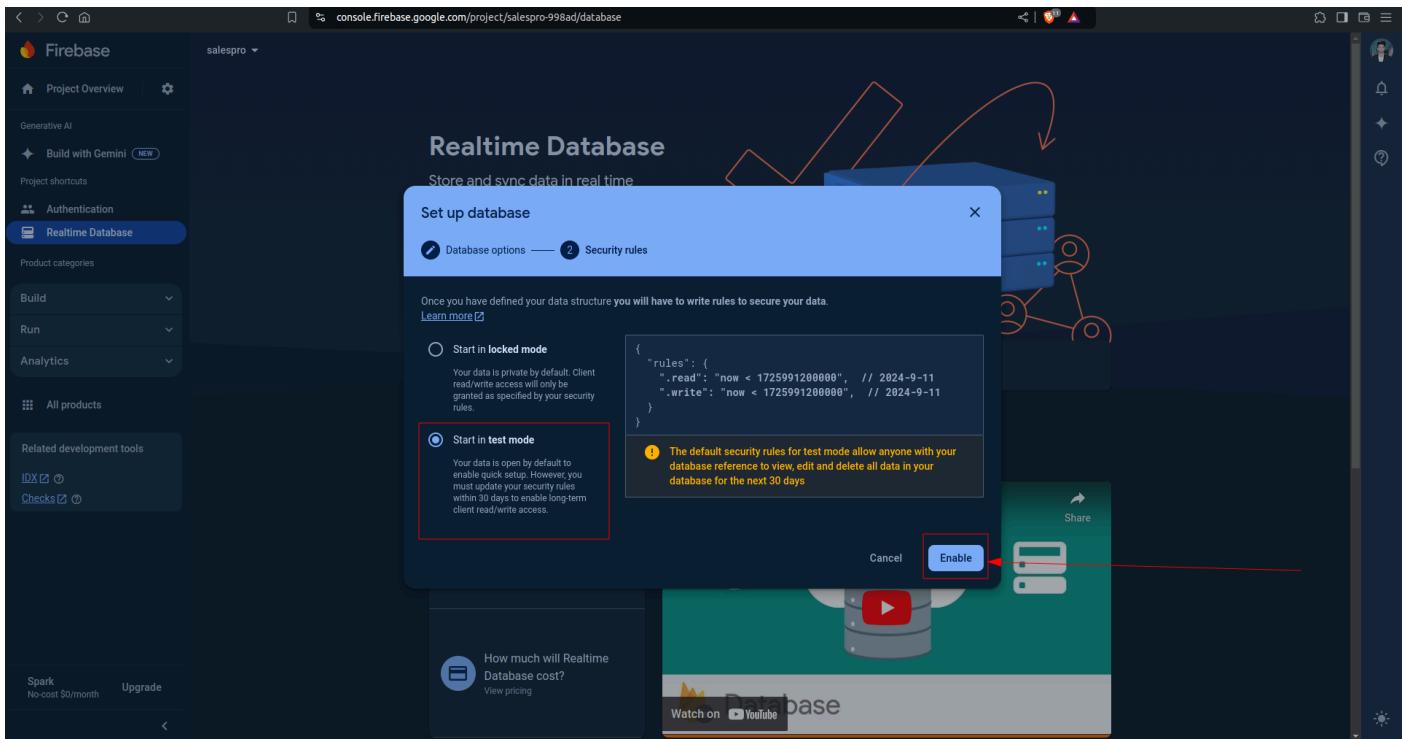
Now again from the **sidebar** click on the **Realtime Database** section, then press the **Create Database** button.

The screenshot shows the Firebase console's Realtime Database section. The 'Create Database' button is highlighted with a red box and a red arrow pointing from the sidebar. The main page features a large blue server icon with a checkmark, indicating readiness. Below the server icon, there's a 'Is Realtime Database right for you?' section and a 'Learn more' section with links to get started and view pricing. A red arrow points from the sidebar's 'Realtime Database' link to the 'Create Database' button.

Here a popup will arise, and **select any location** from the dropdown and press **Next** button.



And select **test mode** and press **Enable**.



Now come to the **Rules** tab of that screen and make the **read/write rules true** like the image. Or remove all the code and just paste this

```
{  
  "rules": {  
    ".read": true,  
    ".write": true,  
  }  
}
```

The screenshot shows the Firebase Realtime Database Rules playground interface. On the left, there's a sidebar with project settings and a 'Realtime Database' section selected. The main area has tabs for 'Data', 'Rules' (which is active), 'Backups', and 'Usage'. A red box highlights the code editor where the following rules are written:

```
1 *  
2 *  {  
3 *   "rules": {  
4 *     ".read": true,  
5 *     ".write": true,  
6 *   }  
}
```

Below the code editor are buttons for 'Publish' and 'Discard'. To the right, there's a 'Rules playground' button. At the bottom, it says 'Database location: United States (us-central1)'.

Make sure you copy and paste the same code as it is. And then press the **Publish** button.

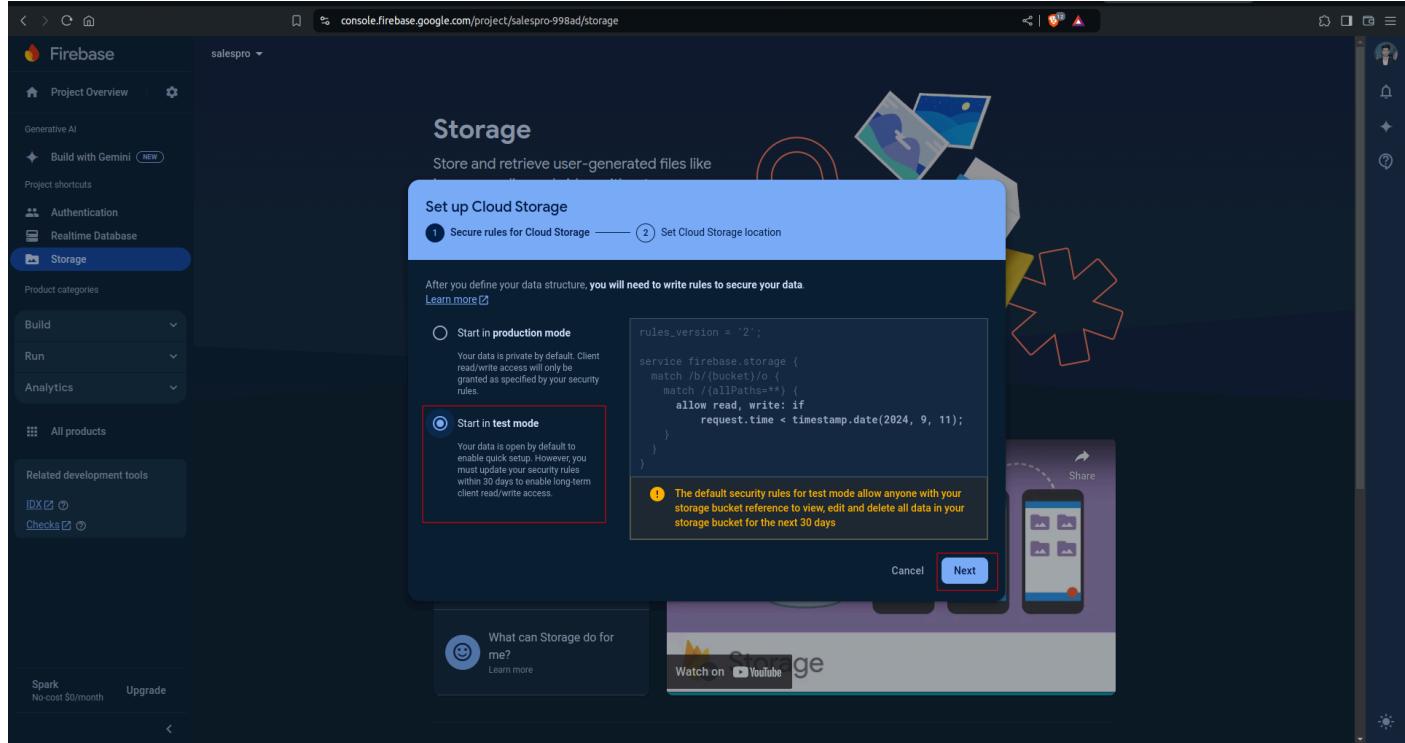
Now again from the **sidebar** click on the **Storage** section, then press the **Get Started** button.

The screenshot shows the Firebase Storage get started page. On the left, there's a sidebar with a 'Storage' section highlighted by a red arrow. The main area features a 'Get started' button with a red arrow pointing to it. To the right, there's a large illustration of a folder containing documents and a lock icon. Below the illustration, there's a 'Learn more' section with three cards:

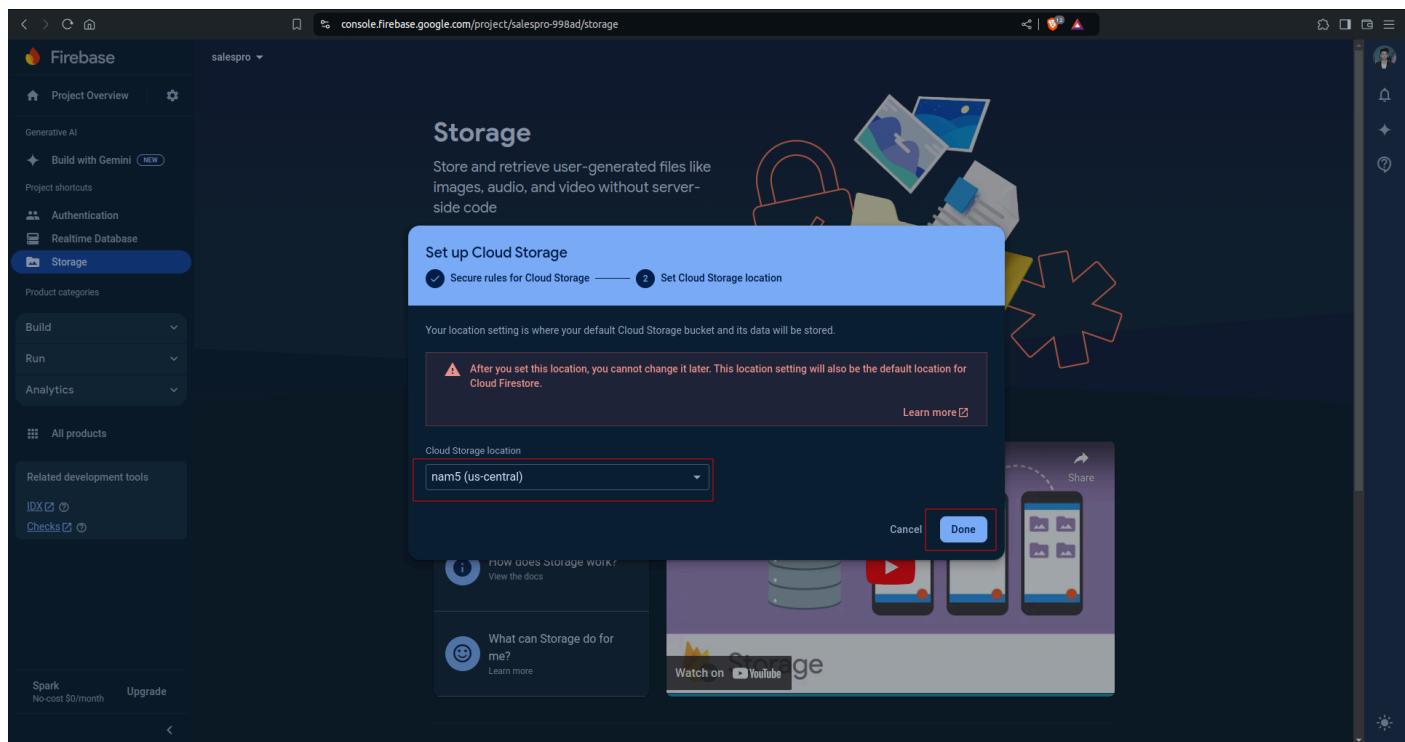
- How do I get started? View the docs
- How does Storage work? View the docs
- What can Storage do for me? Learn more

At the bottom, there's a 'Watch on YouTube' button and a 'Share' icon. The page title is 'Storage' and it describes storing and retrieving user-generated files like images, audio, and video without server-side code.

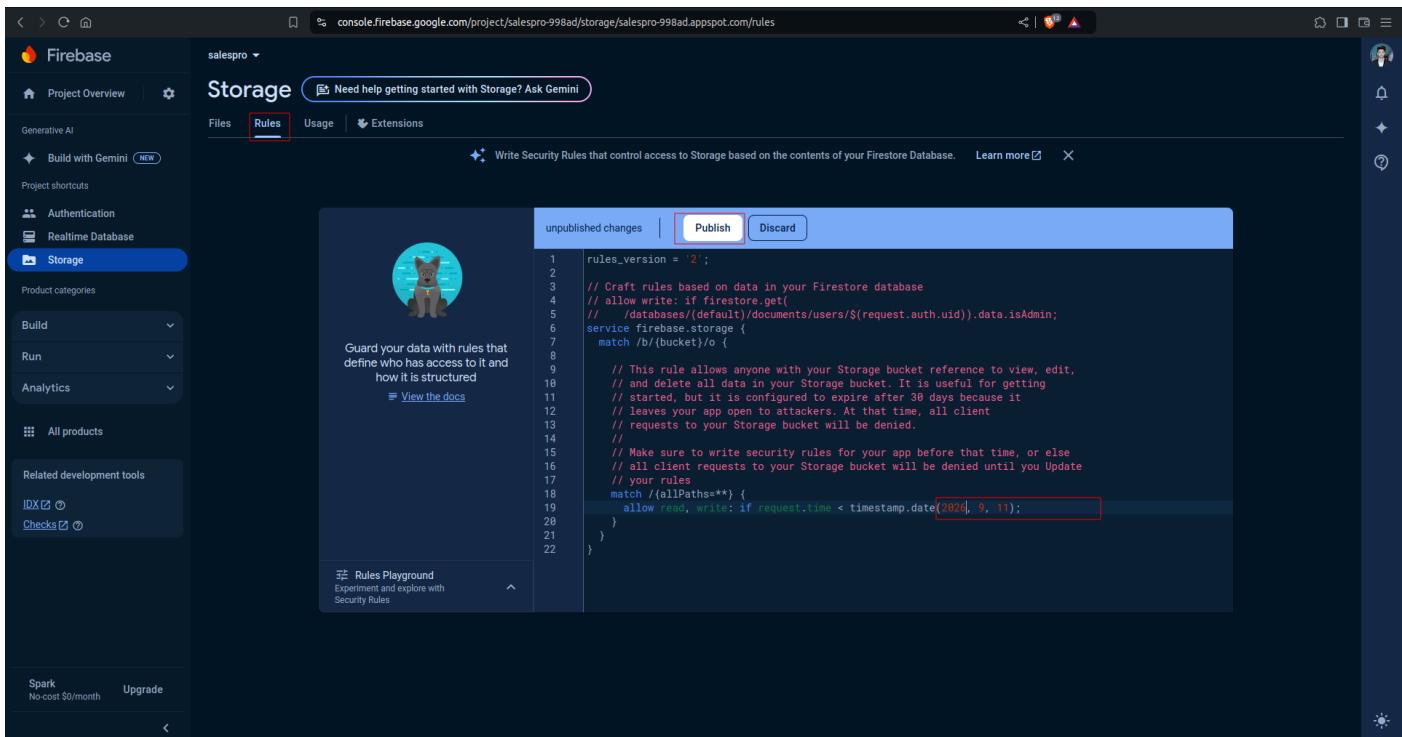
Here a popup will arise, and **Start in test mode** and press **Next** button.



Here **select any location** from the dropdown and press the **Done** button.



Now come to the **Rules** tab on that screen (Storage).



The screenshot shows the Firebase Storage Rules editor. On the left, there's a sidebar with project navigation and a dark theme header. The main area is titled "Storage" and has tabs for "Files", "Rules" (which is selected), "Usage", and "Extensions". A sub-header says "Write Security Rules that control access to Storage based on the contents of your Firestore Database." Below this is a code editor with syntax highlighting for JSON. The code defines rules for a storage bucket:

```
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(2026, 9, 11);
20     }
21   }
22 }
```

At the top of the code editor, there are buttons for "unpublished changes", "Publish" (which is highlighted with a red box), and "Discard". Below the code editor, there's a "Rules Playground" section with a link to "View the docs".

Here increase the time stamp as you want and press the publish button.

## Add Superadmin to Firebase

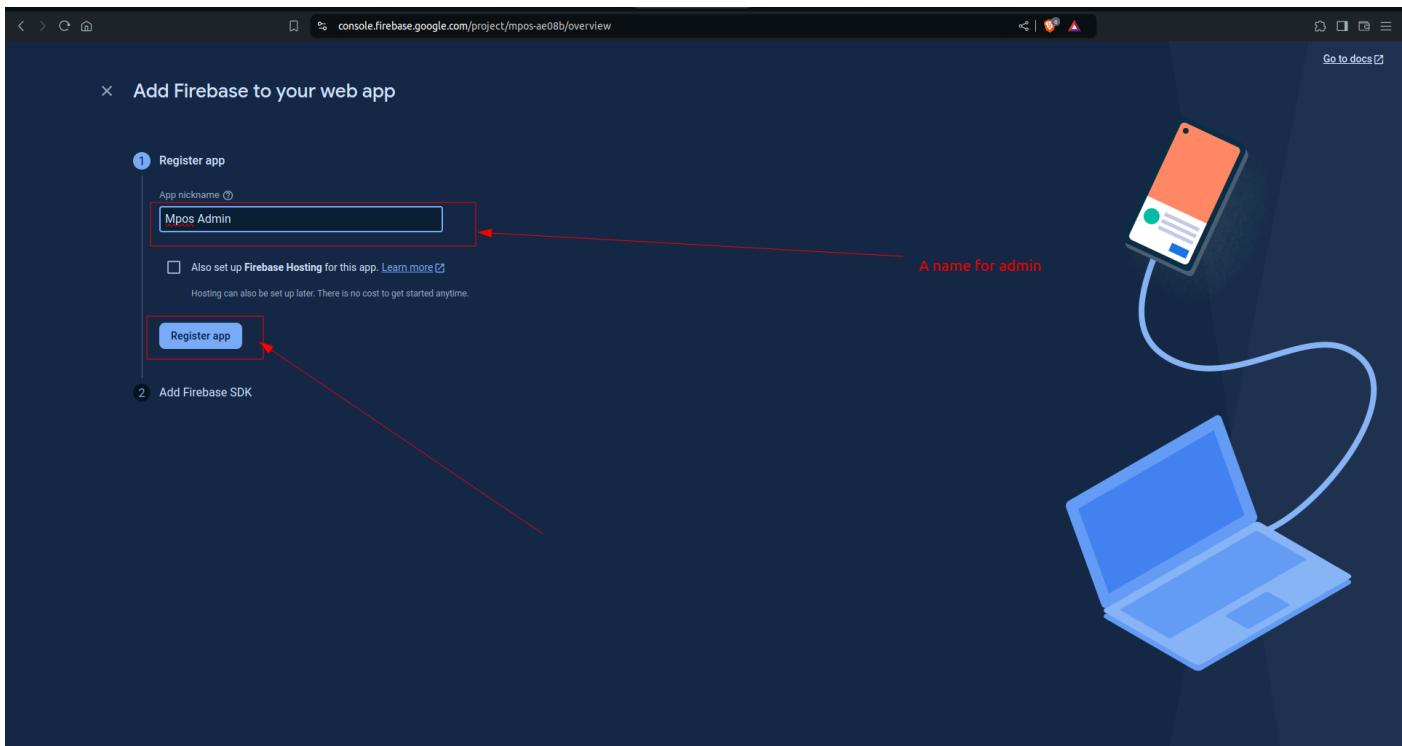
Go back to the **Project overview** screen and here you have to add the web app .

The screenshot shows the Firebase Project Overview page for a project named 'Mpos'. On the left, there's a sidebar with navigation links like 'Project Overview', 'Generative AI', 'Build with Gemini', 'Product categories', 'Build', 'Run', 'Analytics', 'All products', and 'Related development tools'. The main area has a dark background with two cartoon characters. One character is pointing towards a button labeled 'Add Web'. Below this, there's a section titled 'Get started by adding Firebase to your app' with icons for iOS, Android, and Web. A callout arrow points from the text 'Add an app to get started' to the 'Web' icon. Below this, there's a section titled 'Accelerate app development' with cards for 'Authentication', 'Cloud Firestore', 'Storage', and 'Hosting'. At the bottom, there's a footer with 'Spark' and 'Upgrade' buttons.

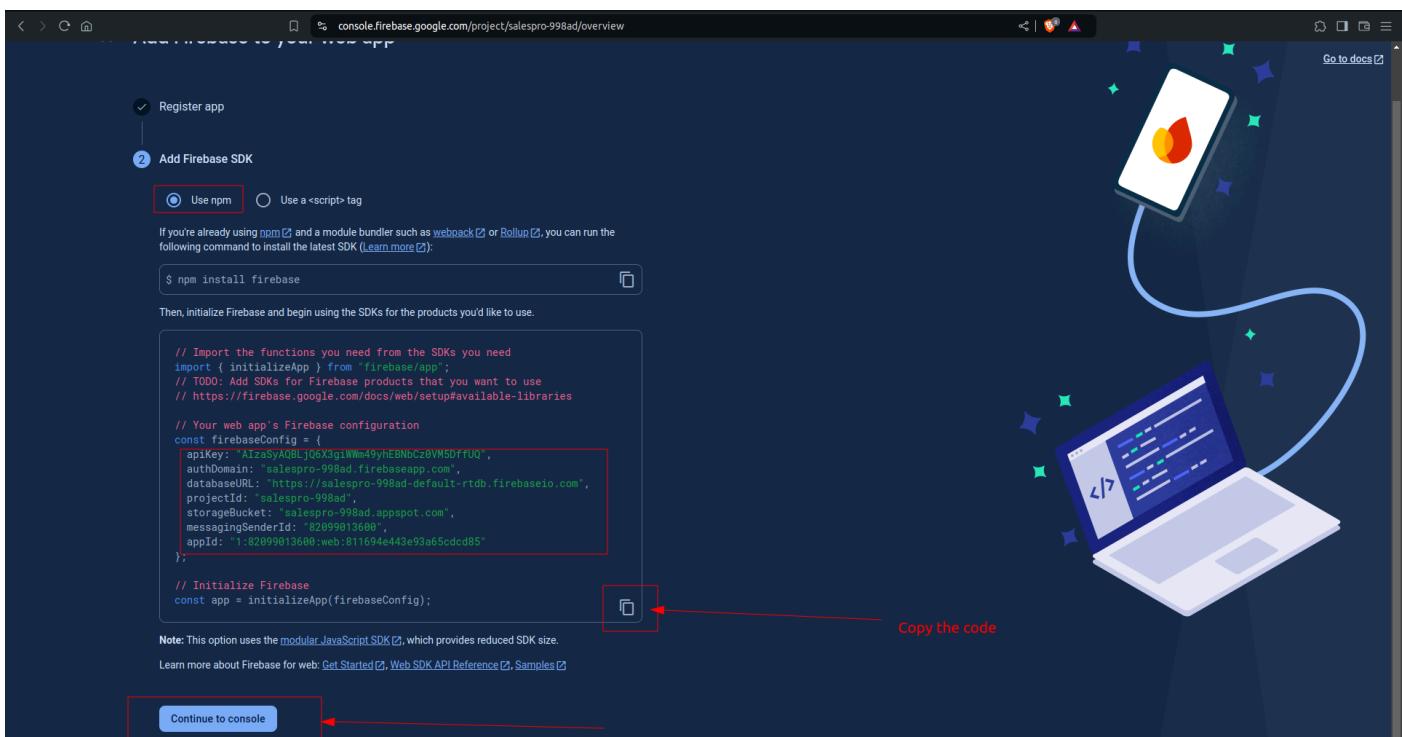
Press on the **Web** button to connect the admin with firebase.

The screenshot shows the Firebase Project Overview page for a project named 'salespro'. The sidebar is similar to the previous one. In the main area, there's a 'Select a platform' dropdown menu with four options: iOS, Android, Web, and Select a platform. A red arrow points from the text 'Press on web' to the 'Web' option in the dropdown. Below this, there's a section titled 'Choose a product to add to your app' with cards for 'Authentication', 'Cloud Firestore', 'Storage', and 'Hosting'. At the bottom, there's a footer with 'See all Build features' and 'Run and optimize your app with confidence'.

Give a name for your salespro admin panel and press the **Register app** button.



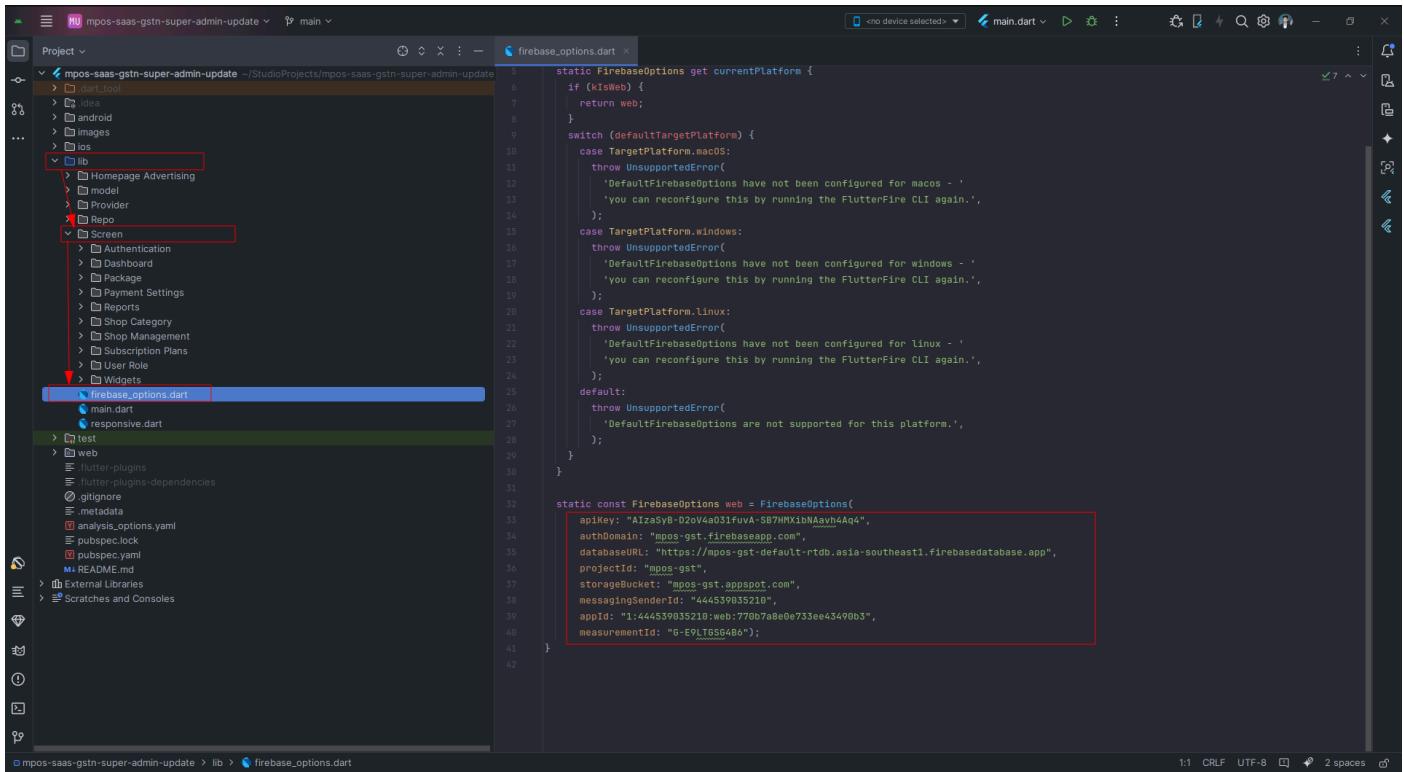
Here make sure you are on the **Use npm** radio button. Then copy the code inside the **firebaseConfig**. Or you can copy them all and keep them in a safe place. And press **Continue the console** button.



Now go back to the android studio and open the salepro superadmin and the **firebase\_options.dart** folder.

Path: /**mpos-saas-gstn-update/lib/Screen/firebase\_options.dart**

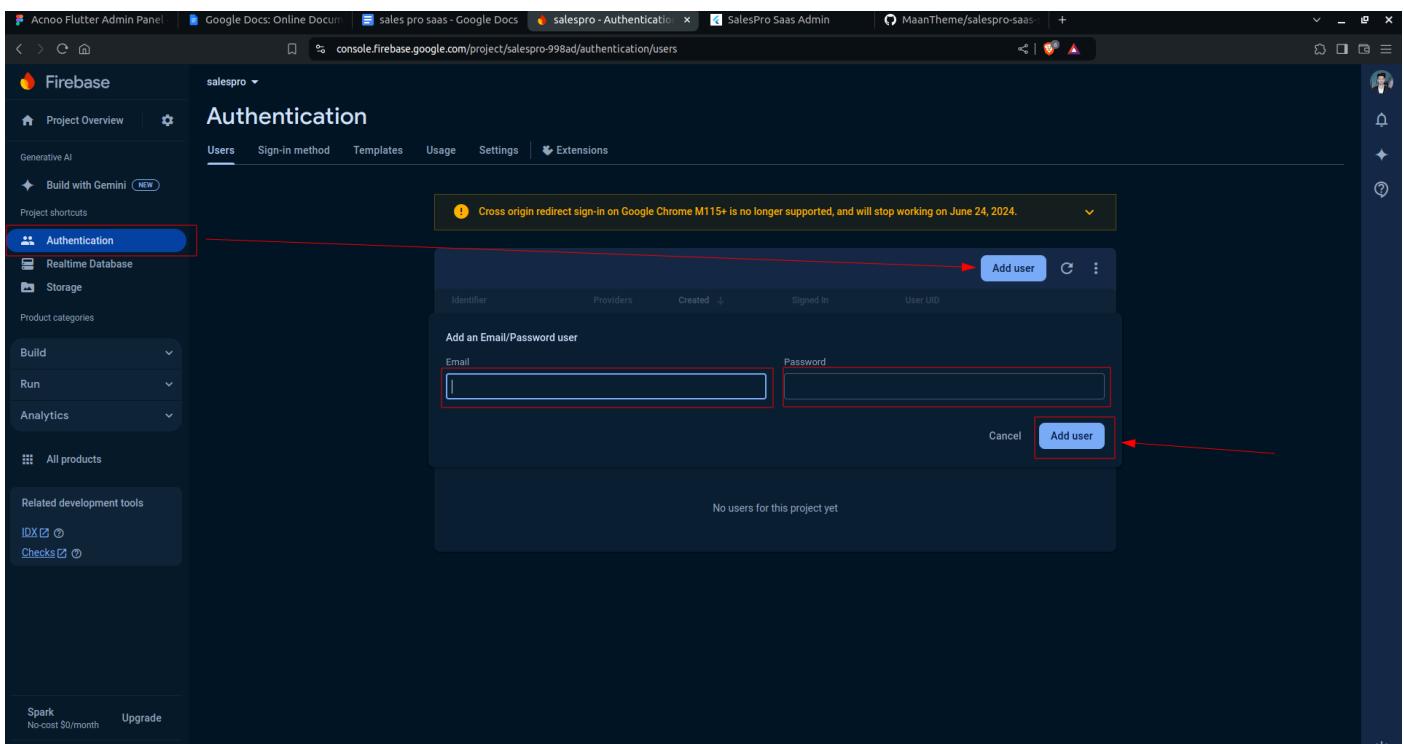
Now replace the credentials which you copied from the **firebase console**. And paste them, and make sure you have pasted all the credentials properly and valid.



```
static const FirebaseOptions web = FirebaseOptions(  
    apiKey: "AIzaSyB-Dz0V4a03JfuVA-S87HMX10NAavh4Au4",  
    authDomain: "mpos-gst.firebaseioapp.com",  
    databaseURL: "https://mpos-gst-default-rtdb.firebaseio.com",  
    projectId: "mpos-gst",  
    storageBucket: "mpos-gst.appspot.com",  
    messagingSenderId: "444539035210",  
    appId: "1:444539035210:web:770b7a0e0e733ee43490b3",  
    measurementId: "6-E9LTSG4B6";
```

Again go back to the **Firebase console**, and Select the **Authentication** from the sidebar and press **Add User**. Here give an email and a password for your **admins** login credentials. And Press **Add User** button under the text box.

Note that this will need every time you login to your admin panel.



The screenshot shows the Firebase Admin Console interface. The left sidebar has 'Authentication' selected. The main area is titled 'Authentication' and shows a table with one row: 'No users for this project yet'. Below the table, there's a form to 'Add an Email/Password user'. It has two input fields: 'Email' and 'Password', both highlighted with a red box. To the right of the 'Email' field is a 'Cancel' button, and to the right of the 'Password' field is an 'Add user' button, also highlighted with a red box. A yellow warning banner at the top right says: 'Cross origin redirect sign-in on Google Chrome M115+ is no longer supported, and will stop working on June 24, 2024.'

Now go back to the android studio and open the salepro superadmin and the **constant.dart** folder.  
Path: /mpos-saas-gstn-super-admin-update/lib/Screen/Widgets/Constant Data/constant.dart

The screenshot shows the Android Studio interface with the project tree on the left and the code editor on the right. The code editor displays the `constant.dart` file from the `mpos-saas-gstn-super-admin-update` project. The file contains various color definitions and text styles. A red arrow points from the text `const kAdminEmail = 'acnooteam@gmail.com';` to the comment `Your email for admin`. The code editor status bar at the bottom indicates the file is 443 lines long, uses CRLF line endings, and is in UTF-8 encoding.

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';

const kAdminEmail = 'acnooteam@gmail.com';

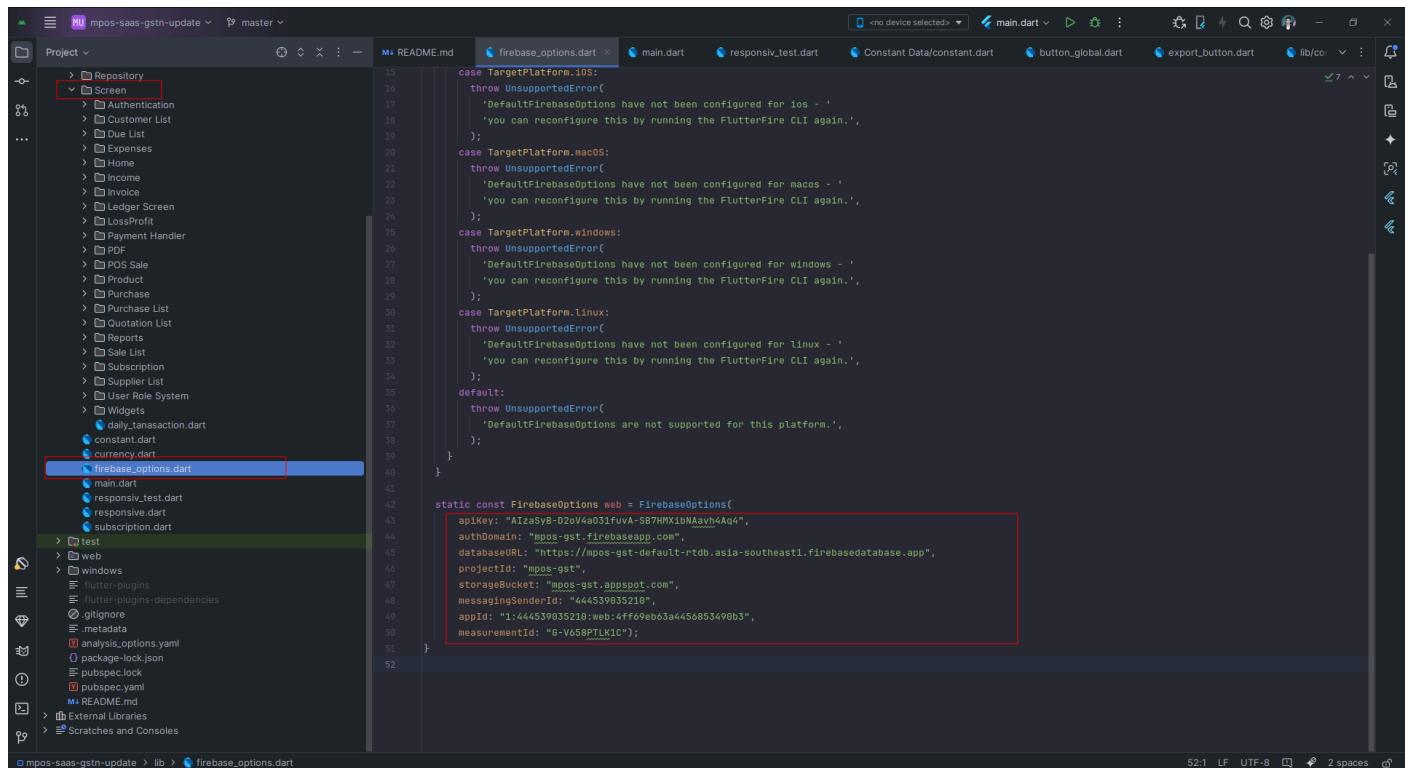
// const kMainColor = Color(0xFF3F80FF);
const kMainColor = Color(0xFF5B26E);
const kDarkGreyColor = Color(0xFF2E2E3E);
const kLightGreyColor = Color(0xFFD0D0B8);
const kGreyTextColor = Color(0xFF828282);
const kBordersColorTextField = Color(0xFFFFE8E7E5);
const kDarkWhite = Color(0xFFFFFFFF);
const kWhiteTextColor = Color(0xFFFFFFFF);
const kRedTextColor = Color(0xFFFFE2525);
const kBlueTextColor = Color(0xFFFF2080F6);
const kYellowColor = Color(0xFFFF8C00);
const kGreenTextColor = Color(0xFF15CD75);
const kTitleColor = Color(0xFF2AE2E3E);
final kTextStyle = GoogleFonts.manrope(
  color: Colors.white,
);

const kButtonDecoration = BoxDecoration(
  color: kMainColor,
  borderRadius: BorderRadius.all(
    Radius.circular(40.0),
  ), // BorderRadius.all
); // BoxDecoration

const kInputDecoration = InputDecoration(
  hintStyle: TextStyle(color: kBordersColorTextField),
  filled: true,
  fillColor: Colors.white,
  enabledBorder: OutlineInputBorder(
    borderRadius: BorderRadius.all(Radius.circular(8.0)),
    borderSide: BorderSide(color: kBordersColorTextField, width: 2),
  ), // OutlineInputBorder
  focusedBorder: OutlineInputBorder(
    borderRadius: BorderRadius.all(Radius.circular(6.0)),
    borderSide: BorderSide(color: kBordersColorTextField, width: 2),
  ); // InputDecoration
OutlineInputBorder outlineInputBorder() {
```

**Firebase setup is complete for Mpos SAAS Superadmin.**

Now replace the same credentials to the **Mpos Web App** which you copied from the **firebase console**.



The screenshot shows a Flutter project named 'mpos-saas-gstn-update' in an IDE. The project structure on the left includes 'Repository', 'Screen', 'Authentication', 'Customer List', 'Due List', 'Expenses', 'Home', 'Income', 'Invoice', 'Ledger Screen', 'LossProfit', 'Payment Handler', 'PDF', 'POS Sale', 'Product', 'Purchase', 'Purchase List', 'Quotation List', 'Reports', 'Sale List', 'Subscription', 'Supplier List', 'User Role System', 'Widgets', 'daily.transaction.dart', 'constant.dart', 'currency.dart', 'firebase\_options.dart', 'main.dart', 'responsive\_test.dart', 'responsive.dart', and 'subscription.dart'. The 'lib' folder contains 'test', 'web', 'windows', 'flutter-plugins', 'flutter-plugins-dependencies', '.gitignore', '.metadata', 'analysis.options.yaml', 'package-lock.json', 'pubspec.lock', 'pubspec.yaml', and 'README.md'. The 'Scratches and Consoles' folder is also visible.

The code editor displays the 'firebase\_options.dart' file. The code defines a static const variable 'FirebaseOptions web' containing the following configuration:

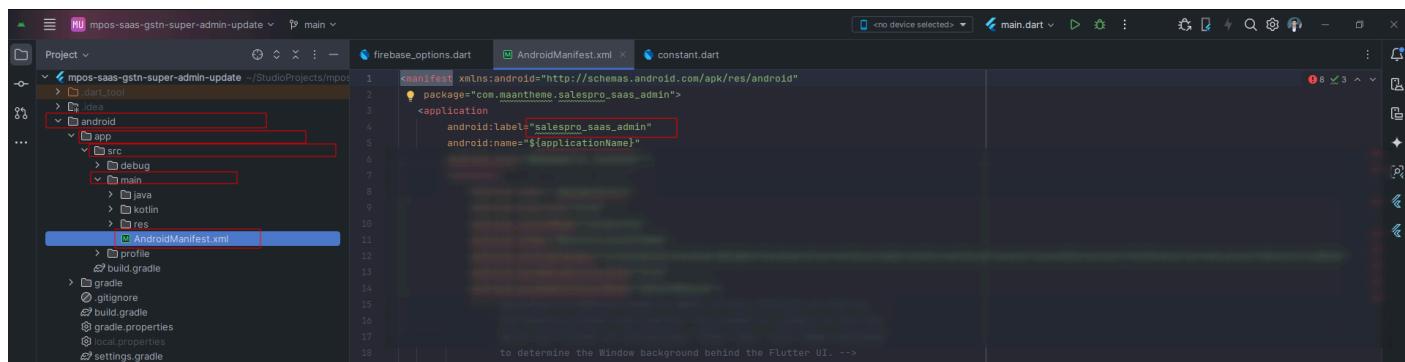
```
static const FirebaseOptions web = FirebaseOptions(  
    apiKey: "AIzaSyB-D20V4a031fvA-SB7RMX1bMaavh4Ag4",  
    authDomain: "mpos-gst.firebaseioapp.com",  
    databaseURL: "https://mpos-gst-default-rtdb.firebaseio.com",  
    projectId: "mpos-gst",  
    storageBucket: "mpos-gst.appspot.com",  
    messagingSenderId: "444539835210",  
    appId: "1:444539835210:web:4fffb9eb03a4456853490b3",  
    measurementId: "G-V658PTLK1C");
```

## Change the app name

### MPOS Super Admin

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

path: Project/android/app/src/main/AndroidManifest.xml



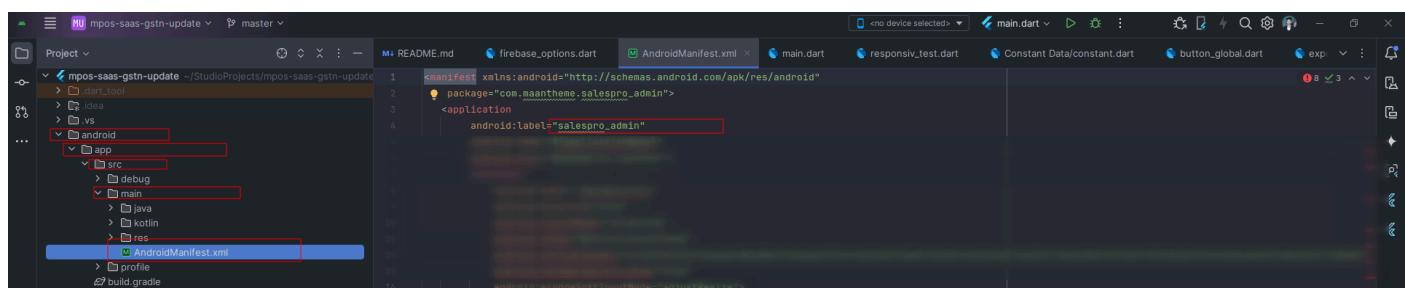
After opening the file change the **android:label = "AppName"**

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

### MPOS Web App

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

path: Project/android/app/src/main/AndroidManifest.xml



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

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

## Change Logo & logo inside admin panel



### Select Images:

- Choose the image that you want to use for the main logo.
- Ensure the image is appropriate and formatted correctly for the app's display requirements.

### Rename Images:

- Rename the selected image with the following filenames:
  - logo: For MPOS Admin “**mpos.png**” &
  - logo: For web App “**mlogo.png**”
- Make sure to keep the file extension as **.png**.

### Replace Existing Images:

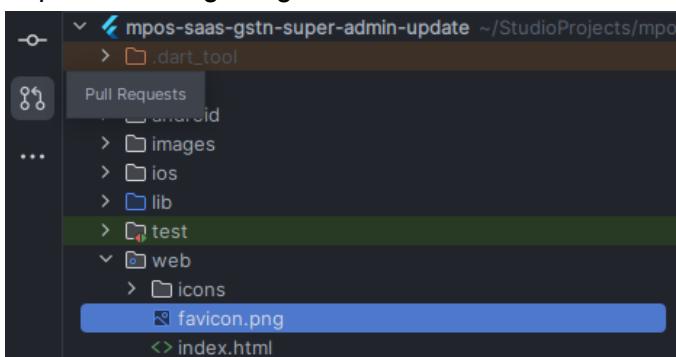
- Locate the folder where the current logo & avatar images are stored. This is typically named images
- Copy and paste your newly renamed images into this folder.
- If prompted, allow the new images to replace the existing ones.

## Change favicon:

### Rename Images:

- Rename the selected image with the following filenames:
  - Splash logo: **favicon.png**
- Make sure to keep the file extension as **.png**.

### Replace Existing Images, **icons** folder inside the **web**

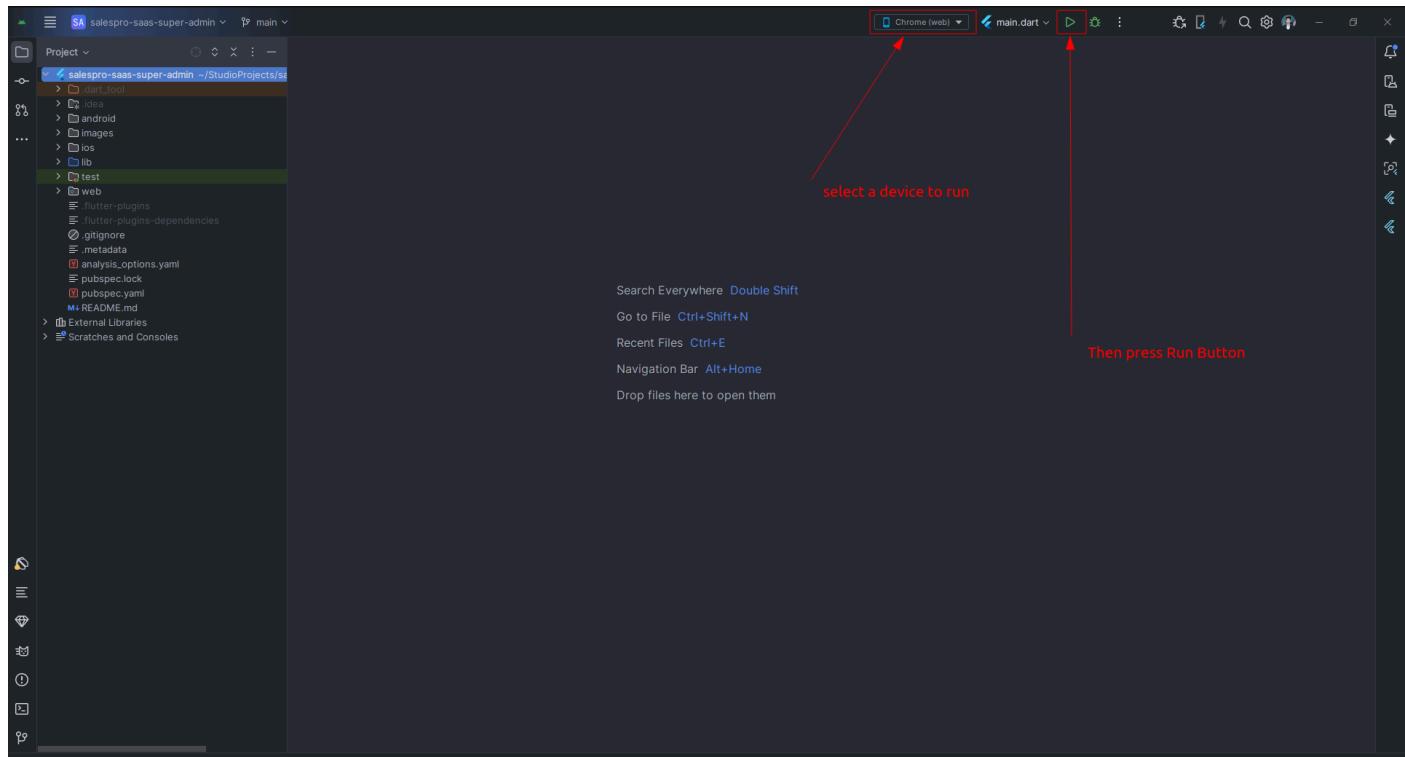


### Restart the App:

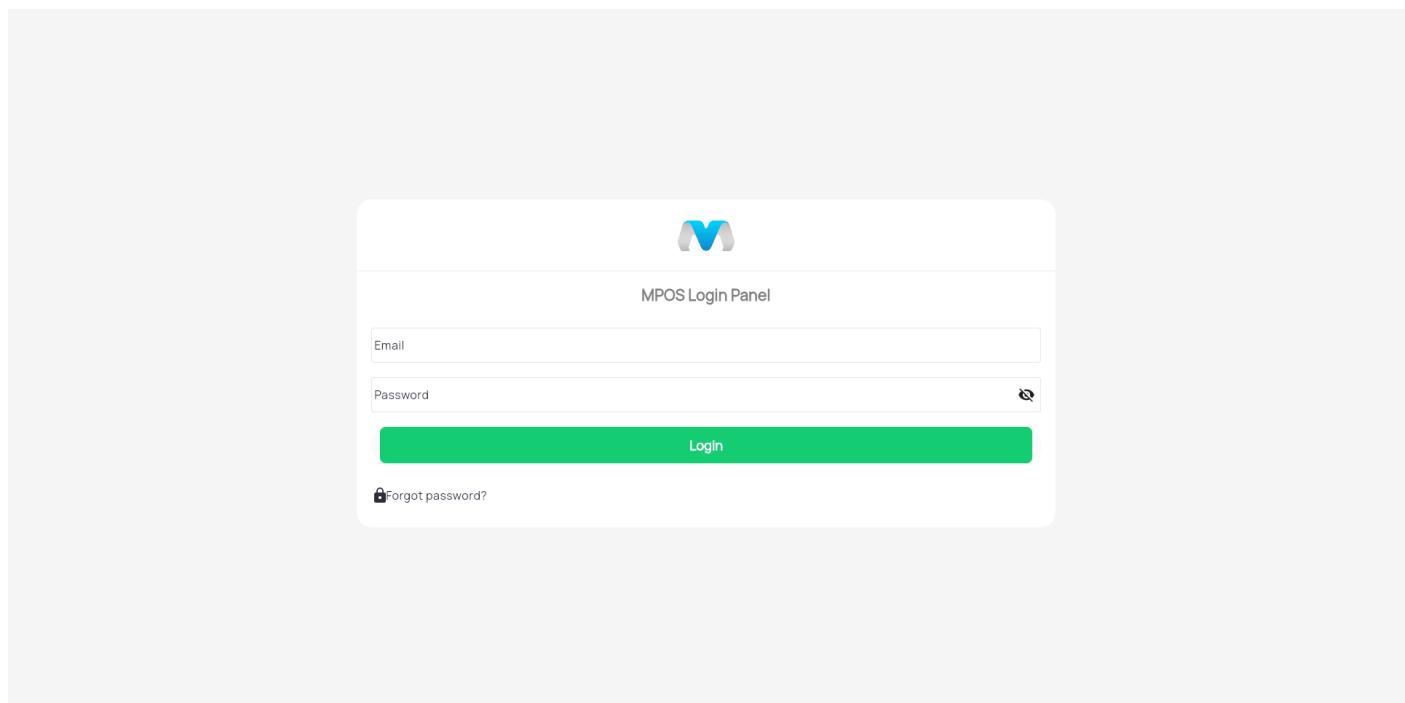
- Close the app completely to ensure all changes take effect.

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

## Now Run the Super Admin First.



After opening, sign in to the admin with your email which you have added for admin.



## Now Add some categories for the users.

S.L	CATEGORY NAME	DESCRIPTION	CREATED BY	ACTION
1	gatiarena	tienda de arena para gatos	Admin	⋮
2	Electronic	Electronic	Admin	⋮
3	Cosmetics	Cosmetics	Admin	⋮
4	clothing	Add new Category	Admin	⋮
5	ddd		Admin	⋮
6	test		Admin	⋮
7	food		Admin	⋮
8	Network		Admin	⋮
9	print		Admin	⋮
10	Spices		Admin	⋮
11	Resturant		Admin	⋮

MPOS SAAS  
Version 3.0

## Add Payment credentials

Payment Settings	
PayPal:	
Is Live	<input type="checkbox"/>
PayPal Client Id	AVzTTWaOR7tDoA5SmulkoNO4r_2ui1UcVAS-Cv_KJQg6SuHDLc
PayPal Client Secret	EEU8C7U6EMxbUYICLRjrZaKZFVc9ZAojAFMor2mqnTbulaP6cg
<input type="button" value="Cancel"/> <input type="button" value="Save"/>	
Razorpay:	
Is Live	<input type="checkbox"/>
Razorpay Key	<input type="text"/>
Razorpay Secret	<input type="text"/>
<input type="button" value="Cancel"/> <input type="button" value="Save"/>	

MPOS SAAS  
Version 3.0

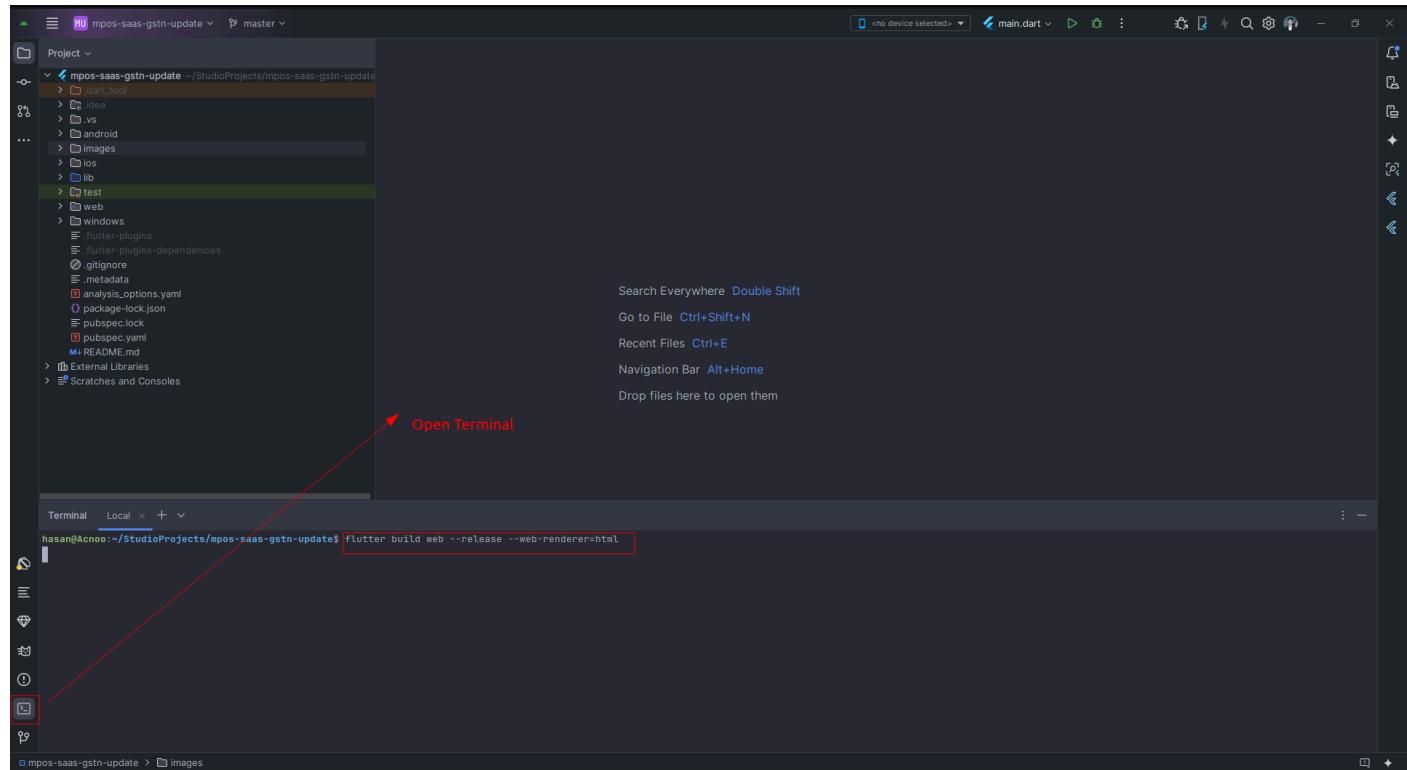
That's all for MPOS Saas Super Admin

## 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 the terminal write the command. Same for Admin and the web app.

```
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: /YourFolderName/mpos-saas-gstn-update/web

Now compress the web folder and extract it to the server

**Support skype:** <https://join.skype.com/invite/kEPqlmF1Vfqk>

**Support email:** [acnooteam@gmail.com](mailto:acnooteam@gmail.com)

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