Introduction To POS Saas- Flutter POS Inventory Softwate
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 iOSand 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: 2022.3.1
- Flutter version: 3.19.4 (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

Setup 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 dothis by following the command below:

```
flutter doctor
```

```
Last login: Tue Nov 21 15:26:50 on console
tahmidtarongo@Tahmids-MacBook-Air ~ % flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.16.0, on macOS 14.1.1 23B81 darwin-arm64, locale
    en-BD)
[✓] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[✓] Xcode - develop for iOS and macOS (Xcode 15.0)
[✓] Chrome - develop for the web
[✓] Android Studio (version 2022.3)
[✓] VS Code (version 1.84.0)
[✓] Connected device (3 available)
[✓] Network resources

• No issues found!
```

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 youselected when it downloaded from the marketplace. This will open the application sourcecode 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 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 are must be followed to build and run application.

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.

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 reuseable widgets are stored here, like(Top bar, Side bar).Also the constant file located in thi 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 has been declared here.

Firebase Setup

Step 1

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

After tap 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 new one.

Here you have enter a name for your project.then click on the ckeck box and 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 for now and press continue.

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

The environment will be 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 be appear wuth a tabbar. 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 on save button.

A new page will be open like this. Here Email/Password are enable 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 Databse from the list inside the build section. And press on Create Database button.

Select any location from the dropdown button and hit 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 secon popup will be arise. And choose the cloud storage location from the dropdown. And press on the Done button.

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.

unpublished changes | Publish | Discard

```
rules_version = '2';

// Craft rules based on data in your Firestore database
// allow write: if firestore.get(
//     /databases/(default)/documents/users/$(request.auth.uid)).data.isAdmin;
service firebase.storage {
  match /b/{bucket}/o {

    // This rule allows anyone with your Storage bucket reference to view, edit,
    // and delete all data in your Storage bucket. It is useful for getting
    // started, but it is configured to expire after 30 days because it
    // leaves your app open to attackers. At that time, all client
    // requests to your Storage bucket will be denied.
    //
    // Make sure to write security rules for your app before that time, or else
    // all client requests to your Storage bucket will be denied until you Update
    // your rules
    match /{allPaths=**} {
      allow read, write: if request.time < timestamp.date(2025, 9, 16);
    }
  }
}
```

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 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 you project. Press **Add user** button.

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

Keep in mind that without this email you can’t log in to the Superadmin 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.(apikey, authDomain, databaseURL, projectID, storageBucket, messagingSenderId,appId) which you copied in last section.
Make sure that you have paste the correct code.

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 write the valid & the same email.

```
import 'dart:convert';
import 'package:firebase_database/firebase_database.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:intl/intl.dart';
import 'package:nb_utils/nb_utils.dart';

const kAdminEmail = 'example@gmail.com';

///apps name

String appsName='Pos Saas';
String appsTitle='Poss Saas Super Admin';
String appsLogo='images/pos.png';


Future<String> getSaleID({required String id}) async {
  String key = '';
  await FirebaseDatabase.instance.ref().child('Admin Panel').child('Seller List').orderByKey().get().then((value) async {
    for (var element in value.children) {
      var data = jsonDecode(jsonEncode(element.value));
      if (data['userId'].toString() == id) {
        key = element.key.toString();
      }
    }
  });
  return key;
}

// const kMainColor = Color(0xFF3F8CFF);
const kMainColor = Color(0xff8424FF);
const kDarkGreyColor = Color(0xFF2E2E3E);
const kLitGreyColor = Color(0xFFD4D4D8);
const kGreyTextColor = Color(0xFF828282);
const kBorderColorTextField = Color(0xFFE8E7E5);
const kDarkWhite = Color(0xFFF5F5F5);
```

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 open the app enter the user email and passord. Which you have given to add new user. After Login you have to follow some steps and it is required to setiing up your app successfully.

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

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

Build File For Server Upload

To build the web application to upload the server, you have to open the terminal section. With andoid studio you can open it from the bottombar 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.

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.

After doing all the task 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 fllow the previous instructions to open.

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 reuseable widgets are stored here, like(Top bar, Side bar).Also the constant file located in thi 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 has been declared here.

Now open the URL https:/console.firebase.google.com/
and click on Go to 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 the you web app. No need to check the firebase hosting for this project and press on 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

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.(apikey, authDomain, databaseURL, projectID, storageBucket, messagingSenderId,appId) which you copied in last section.

Make sure that you have paste the correct code.

```
      throw UnsupportedError(
        'DefaultFirebaseOptions have not been configured for macos - '
        'you can reconfigure this by running the FlutterFire CLI again.',
      );
    case TargetPlatform.windows:
      throw UnsupportedError(
        'DefaultFirebaseOptions have not been configured for windows - '
        'you can reconfigure this by running the FlutterFire CLI again.',
      );
    case TargetPlatform.linux:
      throw UnsupportedError(
        'DefaultFirebaseOptions have not been configured for linux - '
        'you can reconfigure this by running the FlutterFire CLI again.',
      );
    default:
      throw UnsupportedError(
        'DefaultFirebaseOptions are not supported for this platform.',
      );
  }
}

static const FirebaseOptions web = FirebaseOptions(
    apiKey: "AIzaSyBWWnfsKSnQEUaJ1TIVAxb6tU8DIf5lQ",
    authDomain: "pos-saas-a7b6c.firebaseapp.com",
    databaseURL: "https://pos-saas-a7b6c-default-rtdb.firebaseio.com",
    projectId: "pos-saas-a7b6c",
    storageBucket: "pos-saas-a7b6c.appspot.com",
    messagingSenderId: "606700156156",
    appId: "1:606700156156:web:d10d411b843f218a3dd379");
}
```

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 open the app register an account. If the registration is complete successfuly, 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 andoid studio you can open it from the bottombar 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.

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 constant.dart file and change the name as you want.

```dart
import 'dart:convert';
import 'package:firebase_database/firebase_database.dart';
import 'package:flutter/material.dart';
import 'package:flutter_easyloading/flutter_easyloading.dart';
import 'package:mailer/mailer.dart';
import 'package:mailer/smtp_server.dart';
import 'package:nb_utils/nb_utils.dart';
import 'Screen/Due List/due_list_screen.dart';
import 'Screen/Expenses/expenses_list.dart';
import 'Screen/LossProfit/lossProfit_screen.dart';
import 'Screen/Reports/report_screen.dart';
import 'Screen/Stock List/stock_list_screen.dart';
import 'model/user_role_model.dart';

final GlobalKey<RefreshIndicatorState> _refreshIndicatorKey = GlobalKey<RefreshIndicatorState>();

String appsName = 'Pos Saas';
String appsTitle = 'Pos Saas Web';

List<String> selectedNumbers = [];
Future<String> getSaleID({required String id}) async {
  String key = '';
  await FirebaseDatabase.instance.ref().child('Admin Panel').child('Seller List').orderByKey().get().then((value) async {
    for (var element in value.children) {
      var data = jsonDecode(jsonEncode(element.value));
      if (data['userId'].toString() == id) {
        key = element.key.toString();
      }
    }
  });
  return key;
}

String constUserId = '';
bool isSubUser = false;
String constSubUserTitle = '';
```

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

POS Saas Mobile App Setting

Run the app like the web and superadmin 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 andoid 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

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.

Generate SHA code for the app

For generate the SHA code, go to android, select gradlew & right click then open with Terminal. It will open the terminal automatically. Then enter the command

```
./gradlew signinReport
```

and press enter. It will generate the SHA code.

After generate the SHA code copy two SHA-256 from the varient: debug & release.

Then go to the firebase console and open the project and click on the android icon.

And then click on the setting button on the popup dialog. It will open a new page.

Now scroll to the down section and here add one SHA code on the form field and press Save button.

Again press on add fingerprint and add the second SHA code and press Save button.

Now go to the environment and run the project.

Change the app name

To chnage the app name you have to got to the AndroidManifest.xml file.
path: Project/android/app/src/main

After opening the file chnage the **android:label =”AppName”**

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

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 folder. 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.

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**. Thats all. After sometimes it will build a release APK file.

After completeing 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 you app in google play store you can follow this instruction:
https://docs.flutter.dev/deployment/android

For 24/7 support email us here: maantheme@gmail.com

Skype: https://join.skype.com/invite/kEPqlmF1Vfqk